Dynamic Programming Optimization in Line of Sight Networks[☆]

Pavan Sangha[a], Prudence W. H. Wong[a], Michele Zito[a,*]

[a]Department of Computer Science
University of Liverpool
Liverpool L69 3BX, United Kingdom

Abstract

Line of Sight (LoS) networks were designed to model wireless communication in settings which may contain obstacles restricting node visibility. For fixed positive integer d, and positive integer ω, a graph $G = (V, E)$ is a (d-dimensional) LoS network with range parameter ω if it can be embedded in a finite cube of the d-dimensional integer grid so that each pair of vertices in V are adjacent if and only if their embedding coordinates differ only in one position and such difference is less than ω.

In this paper we investigate a dynamic programming (DP) approach which can be used to obtain efficient algorithmic solutions for various combinatorial problems in LoS networks. In particular DP solves the Maximum Independent Set (MIS) problem in LoS networks optimally for any ω on *nar*-

---

*Corresponding author
  Email addresses: `pavan@liverpool.ac.uk` (Pavan Sangha),
`pwong@liverpool.ac.uk` (Prudence W. H. Wong), `michele@liverpool.ac.uk` (Michele Zito)

row LoS networks (i.e. networks which can be embedded in a $k \times k \ldots \times k \times n$ region, for some fixed k independent of n). In the unrestricted case it has been shown that the (decision version of the) problem is NP-hard when $2 < \omega = O(n^{1-\delta})$, for fixed $0 < \delta < 1$. We describe how DP can be used as a building block in the design of good approximation algorithms in this case. In particular we present a semi-online polynomial-time approximation scheme for the MIS problem in narrow d-dimensional LoS networks, as well as a polynomial-time 2-approximation algorithm and a fast polynomial time approximation scheme for the MIS problem in arbitrary d-dimensional LoS networks. Finally we comment on how the approach can be adapted to prove similar results for a number of important optimization problems in LoS networks.

Keywords: Networks, Dynamic Programming, Optimization, Approximation Algorithms

1. Introduction

A wireless network typically consists of devices that communicate using radio frequencies, bluetooth or other wireless protocols. Geometric graphs often provide a good model for such networks with vertices representing the devices, and edges associated to the communication ability between pairs of devices. A number of issues reduce the potential of wireless communication. First of all there is typically a communication range restriction: devices should be close in distance in order to be able to communicate. Also, real world wireless networks are typically prone to line of sight restrictions, often due to the presence of a large number of obstacles, like those found in urban

2

settings. A group of devices can only communicate if they are both close and also there is no obstacle between them. While the presence of obstacles can be difficult to model, it is clear that a good model of wireless network should ideally incorporate both communication range restrictions and line of sight restrictions.

Frieze et al. [2] introduced the notion of (random) 2-dimensional Line of Sight (LoS) networks and studied connectivity problems in this setting. Since then connectivity in higher dimensions, percolation and communication problems have been analyzed [3, 4, 5] in the same model. For positive integers d, k and n, let $\mathbb{Z}_n^d$ be the d-dimensional cube $\{1, \ldots, n\}^d$. From now on we assume that $d > 1$ is a fixed integer independent of n and k. We will also work with *narrow* "cubes" $\{1, \ldots, x_1\} \times \ldots \times \{1, \ldots, x_{d-1}\} \times \{1, \ldots, n\}$, where the x_i are positive integers bound by k, a positive integer constant independent of n, which we denote by $\mathbb{Z}_{n,k}^d$. We say that distinct points p_1 and p_2 in one of these cubes *share a line of sight* if their coordinates differ in a single place. In this paper we mainly work with (vertex) weighted graphs: these will be described by triples (V, E, w) where as usual V is the set of nodes, E is the set of edges, and w is a function assigning a positive weight to each element of V. An *unweighted* graph is a weighted graph whose weighting function is the constant $w(v) = 1$ for all $v \in V$. A graph $G = (V, E, w)$ is said to be a *Line of Sight (LoS) network (with parameters n, k and ω)* if there exists an embedding $f_G : V \to \mathbb{Z}_n^d$ such that $\{u, v\} \in E$ if and only if $f_G(u)$ and $f_G(v)$ share a line of sight and the (Manhattan) distance between $f_G(u)$ and $f_G(v)$ is less than ω. We refer to ω as the *range parameter* of the network. A *narrow* LoS network is a LoS network such that $f_G(V) \subseteq \mathbb{Z}_{n,k}^d$. LoS networks

3

keep the distance constraints of other geometric models [6] but also provide a simple mechanism to model communication in an environment containing obstacles.

In this work we mainly focus on the well-known Maximum Independent Set problem. In fact we work with the weighted version of it, where one is after an independent set of the largest possible totalk weight, defined as the sum of the weights of the elements of the chosen set (precise definitions of both problems are given in Section 2). (Narrow) unweighted LoS networks could be seen as simplistic models of urban environment (e.g. a portion of Manhattan, where junctions correspond to nodes and range constraints define the possible connections). In this context large independent sets could be used to assign police officers to junctions so as to maximize the police presence (but still guarantee that two officers cannot shoot each other, assuming their gun's firing range is at most $\omega - 1$ blocks). In general finding large independent sets in a graph is NP-hard [7] and even finding good approximate solutions in polynomial time is difficult [8]. On LoS networks, in the unweighted case, if $\omega \leq 2$ or $d = 2$ and $\omega = n$ the problem can be solved optimally in polynomial time: in the former case the graph is bipartite and MIS can be reduced to the maximum matching problem [9], in the latter the particular LoS networks are also known as *gridline graphs* and the result can be found in [10]. However, Sangha and Zito [11] showed that the (decision version of the) general problem is NP-hard for $d \geq 2$ and $2 < \omega = O(n^{1-\delta})$ where $0 < \delta < 1$ is fixed, and that it admits a polynomial-time d-approximation and an efficient polynomial time approximation scheme (or EPTAS, following [12]) for constant ω.

4

In this paper we describe two algorithms that are guaranteed to output good quality solutions for the MIS in LoS networks when ω is a constant independent of n. The first one is a polynomial-time approximation algorithm that returns a solution whose total weight is at least half the weight of an optimal solution on any LoS network admitting a d-dimensional embedding. For $d > 2$ no such algorithm was known. The second one is a new efficient polynomial-time approximation scheme for the same problem that is faster than the one in [11]. The two results hinge on a dynamic programming strategy that can be used to solve optimally the MIS problem on narrow instances, for any ω. A similar approach was recently proposed by Jallu and Das [13] in the context of unit disk graphs (which in many ways represent a continuous version of the LoS network model). But they only consider a 2-dimensional model, we deal with a general range parameter, and the polynomial-time approximation scheme that they present is slower than ours.

The technique described in this paper, also finds application [14] in the following scheduling problems. Suppose that a company manages advertisements from some k clients over a long period of n discrete time points. At any time advertisements of some subset of clients are available to be aired but the company can only select a certain number l of them to advertise due to resource limitation. In addition some "advertisement diversity" policy requires that advertisements from the same client cannot be aired more than once in a given period of $\omega - 1$ time instants. The goal of the company is to schedule the airing of these advertisements satisfying the constraints and maximising the number of advertisements aired. This problem (which from now on will be referred to as ADSSCHED) has one slight difference from the

5

MIS problem on narrow 2-dimensional LoS networks, in the sense that the "proximity" restriction only applies to one dimension (the time dimension) but not the other (the client dimension). Nevertheless, as we show later, the solution we develop can be adapted to solve this problem.

It is important to stress that, in what follows, we always assume that the various LoS networks are provided along with an embedding in $\mathbb{Z}_n^d$ (and in fact this is a crucial feature exploited by the algorithms in this paper). Therefore, the results on general LoS networks presented here should be compared with the ones in [11] where the knowledge of such embedding was not assumed. For completeness we mention that, since we started our work, others [15] have investigated the problem of deciding whether a graph admits a particular line of sight embedding.

Finally, we remark that the approximation strategies described in the context of the MIS apply to a number of other optimization problems in LoS networks. These include Vertex Cover, Min Dominating Set, Min Edge Dominating Set, Max Triangle Packing, Max H-matching, Max Tile Salvage.

The rest of the paper is organized as follows. After a section containing some useful definitions, in Section 3, we describe our main technical tool: a dynamic programming approach that solves optimally the MIS problem in narrow (LoS network) instances. We present the algorithm, a proof of correctness and a simple application to the AdsSched problem defined above. The remaining sections present further applications of this idea. Section 4 describes how the dynamic programming algorithm can be incorporated in a semi-online [16] algorithm which always returns a good quality feasible solution to the MIS problem on narrow instances. Section 5 presents the

6

approximation algorithm for the MIS problem in general d-dimensional LoS networks, whereas Section 6 focuses on the EPTAS for the 2-dimensional case, and some additional applications. Section 7 wraps up the paper with a summary of the results presented and some directions for future work.

2. Problem Definitions and Preliminaries

Although all the techniques described in this work can be applied, with minor changes, to a number of graph-theoretic optimization problems, the paper mainly focuses on the Maximum Independent Set problem. To keep the presentation self-contained we spell-out the definition of this problem.

Maximum Independent Set. (MIS)

INSTANCE: A graph $G = (V, E)$.

FEASIBLE SOLUTIONS: All sets of vertices $U \subseteq V$ such that no two vertices in U are joined by an edge in E.

COST: The cardinality of the chosen solution U.

OPTIMIZATION: we are interested in solutions of MAXIMUM cost.

In fact our results apply to the weighted version of the same problem where one is after independent sets of the large possible weight.

Maximum Weighted Independent Set. (MWIS)

INSTANCE: A weighted graph $G = (V, E, w)$.

FEASIBLE SOLUTIONS: All sets of vertices $U \subseteq V$ such that no two vertices in U are joined by an edge in E.

7

COST: The sum of the weights of the elements of the chosen solution U.

OPTIMIZATION: we are interested in solutions of MAXIMUM cost.

In this paper arrays will be d-dimensional tables of non-negative numbers. In particular, for fixed $k > 0$, *narrow* arrays are tables of size $x_1 \times \ldots \times x_{d-1} \times y$ where the $x_i \in \{1, \ldots, k\}$ for all $i \in \{1, \ldots, d-1\}$ whereas y is a positive integer no larger than $n + \omega$. It will be convenient to group the first $d - 1$ indices and so we will often write $\boldsymbol{x}^{d-1} \times y$ instead of $x_1 \times \ldots \times x_{d-1} \times y$ or $A[\boldsymbol{i}, j]$ instead of $A[i_1, \ldots, i_{d-1}, j]$. In this context the jth *column* of array A will be the collection of elements $A[\boldsymbol{i}, j]$ for all possible values of $\boldsymbol{i}$.

- For any array A and $j_1, j_2 \in \{1, \ldots, n\}$ with $j_1 \leq j_2$, denote by $A[j_1 : j_2]$ the sub-array containing columns $j_1, \ldots, j_2$. When $j_1 = j_2 = j$ we use $A[j]$ instead of $A[j : j]$ unless ambiguity arises.

- For any two arrays A_1, A_2 of size $\boldsymbol{x}^{d-1} \times y$, we say that A_1 *agrees with* A_2, denoted by $A_1 \leq_a A_2$, if $A_1[\boldsymbol{i}, j] \leq A_2[\boldsymbol{i}, j]$ for all $\boldsymbol{i}$ and j.

- For any array A, we denote by $h(A)$ (resp. $t(A)$) the *head* (resp. *tail*) *subarray* of A containing all but the last column (resp. all but the first column) of A. In other words, $h(A)$ and $t(A)$ have $y - 1$ columns if A has y columns.

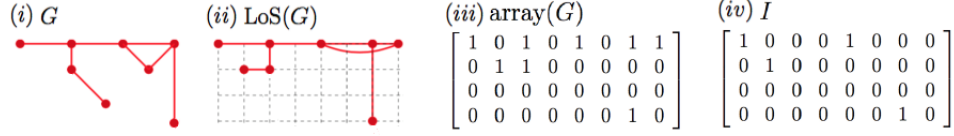- We say that A_1 is *consistent* with A_2 (in symbols $A_1 \vDash A_2$) if $t(A_1)$ is the same as $h(A_2)$.

8

Figure 1: Figure (i) is a graph G and Figure (ii) is its LoS embedding in an 8×4 rectangle in $\mathbb{Z}^2$ with $\omega = 4$. Figure (iii) represents the array layout of G (ignoring the leftmost ω columns of zeroes) and Figure(iv) is an independent array of largest array sum, corresponding to the largest independent set in the graph G.

- Let the *column sum of an array A at column j* be the quantity

$$||A||_j = \sum_i A[\boldsymbol{i}, j].$$

We refer to the quantity $\sum_j ||A||_j$ as the *array sum of A*, and we denote it by $||A||$.

Given a weighted narrow LoS network $G = (V, E, w)$, let array(G) be a $\boldsymbol{k}^{d-1} \times (n + \omega)$ array satisfying array$(G)[\boldsymbol{i}, j] = w(v)$ (resp "0") if and only if location $(\boldsymbol{i}, j) \in \mathbb{Z}_{n,\boldsymbol{k}}$, for some $j \in \{1, \ldots, n\}$, corresponds (resp. does not correspond) to a vertex $v \in V$ in the LoS embedding of G. Also, array$(G)[\boldsymbol{i}, j] = 0$ for any $\boldsymbol{i}$ and $j \in \{-(\omega - 1), \ldots, 0\}$ (the left-most ω columns of array(G) will be needed at the start of the dynamic programming process described in Section 3). Figure 1 provides an example. In this setting an *independent array I* of array(G) is any array of size $\boldsymbol{k}^{d-1} \times (n+\omega)$ satisfying

1. $I \leq_a$ array(G) and

2. for distinct columns j_1, j_2 if $I[\boldsymbol{i}, j_1] > 0$ and $I[\boldsymbol{i}, j_2] > 0$ then $|j_1 - j_2| \geq \omega$ and for distinct rows indexed by $\boldsymbol{i}_1$ and $\boldsymbol{i}_2$ if $I[\boldsymbol{i}_1, j] > 0$ and $I[\boldsymbol{i}_2, j] > 0$ then either $\boldsymbol{i}_1$ and $\boldsymbol{i}_2$ do not share a line of sight or they do but the gap

9

between the values of the differing co-ordinate is at least ω.

A *feasible array* W is an array of size $\boldsymbol{k}^{d-1} \times \omega$ only containing zeros or ones and such that there exists a $\boldsymbol{k}^{d-1} \times \omega$ independent array I_W such that $W[\boldsymbol{i}, j] = 1$ (resp $= 0$) if and only if $I_W[\boldsymbol{i}, j] > 0$ $(= 0)$. The array I_W is a *witness of* W. Note that I_W is a sub-array of array(G) and as such its elements could be arbitrary non-negative numbers. The feasible arrays, however, only contain zeroes and ones. More importantly, since any feasible array has exactly ω columns, it contains at most one non-zero entry per row. We denote by $\mathcal{F}$ the set of all feasible arrays of size $\boldsymbol{k}^{d-1} \times \omega$, and for each $j \in \{0, \ldots, n\}$, $\mathcal{F}_{G,j} \subseteq \mathcal{F}$ is the set of feasible arrays W satisfying $W \leq_a$ array$(G)[j - \omega + 1 : j]$. Note that in particular for any independent array I of array$(G)[j - \omega + 1 : j]$ for $1 \leq j \leq n$ is in fact the witness of some $W \in \mathcal{F}_{G,j}$.

We observe that $\mathcal{I}$ is an independent set of G if and only if $\mathcal{I}$ is an independent array of array(G). Thus finding a maximum total weight independent set in G is equivalent to finding the independent array of array(G) with the largest array sum (we refer to such an array as a *largest independent array*). In Section 3 we show how a simple DP algorithm finds an independent array of array(G) with the largest array sum by working with the feasible arrays of array(G). Because of this correspondence, in the next section we refer to array(G) as G and we work with arrays instead of graphs.

3. Dynamic Programming

Given the array G of size $\boldsymbol{k}^{d-1} \times (n + \omega)$, the main idea of the optimal algorithm we describe in this section is to be guided in its choices by a

10

Algorithm 1 Computing the largest independent array in G

1: /* Initialisations */
2: $\mathrm{MIS}[0, \overrightarrow{0}] = 0$, where $\overrightarrow{0}$ is the $\boldsymbol{k}^{d-1} \times \omega$ array of all 0's, the only element of $\mathcal{F}_{G,0}$.
3: **for** $j = 1, \ldots, n$ **do**
4:     **for** $W \in \mathcal{F}$ **do**
5:         $\mathrm{MIS}[j, W] = 0$
6:     **end for**
7: **end for**
8:
9: /* Array sums computation */
10: **for** $j = 1, \ldots, n$ **do**
11:     **for** $W \in \mathcal{F}_{G,j}$ **do**
12:         let W^* be the feasible array in $\mathcal{F}_{G,j-1}$, $W^* \vDash W$, maximizing $\mathrm{MIS}[j-1, W^*]$
13:         $\mathrm{MIS}[j, W] = W[\omega] \cdot G[j] + \mathrm{MIS}[j-1, W^*]$
14:         $\mathrm{pred}[j, W] = W^*$
15:     **end for**
16: **end for**
17:
18: /* Retrieving the independent set */
19: Find $W^* \in \mathcal{F}_{G,n}$ that maximizes $\mathrm{MIS}[n, W]$
20: Set I as the rightmost column of W^*
21: **for** $j = n$ **downto** 2 **do**
22:     $W^* = \mathrm{pred}[j, W^*]$
23:     Redefine I as the rightmost column of $\mathrm{pred}[j, W^*]$ concatenated with I
24: **end for**
25: **return** I

---

table containing array sums of independent arrays of portions of G. For each
$j \in \{0, \ldots, n\}$, the process manages a table $\mathrm{MIS}[j, W]$, indexed by j as well as
all possible $\boldsymbol{k}^{d-1} \times \omega$ feasible arrays W. At the of each execution of Algorithm
1, $\mathrm{MIS}[j, W]$, for $j \in \{1, \ldots, n\}$ will contain the size of the maximum weight
independent sets in the portion of graph G that is embedded in columns
one through to j in $\mathbb{Z}_{n,k}^d$. During each iteration of the main loop in the
Algorithm (lines 10 to 16) the process runs through all $W \in \mathcal{F}_{G,j}$, and tries
to extend the independent arrays in $G[-(\omega - 1) : j - 1]$ to independent
arrays in $G[-(\omega - 1) : j]$ witnessing W. Let I' be an independent array in
$G[-(\omega-1) : j-1]$ such that $I'[j-\omega : j-1]$ is a witness for some $W' \in \mathcal{F}_{G,j-1}$

$$
\text{(i)} \quad G[1,\ldots,8] = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad I' = \left[\begin{array}{ccccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \quad W' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
\text{(ii)} \quad G[1,\ldots,9] = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \quad I = \left[\begin{array}{cccccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array} \right] \quad W = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
$$

Figure 2: Figure (i) shows 8 columns of an array G and the independent array I' of $G[1:8]$ with the largest array sum satisfying $I'[6:8] \equiv W'$. In Figure (ii) the independent array I is the independent array of $G[1:9]$ which has the largest array sum satisfying $I[7:9] \equiv W$. Note $W' \vDash W$ and that I can be obtained from I' by appending the last column of W to I'.

and assume that W' is consistent with W. By considering the next column of G, we extend I' to an independent array I of $G[-(\omega - 1) : j]$ which is a witness to W. $\mathrm{MIS}[j, W]$ contains the array sum of an independent set whose right-most ω columns are witnessed by W. The expression $W[\omega] \cdot G[j]$ on line 13 is the *Frobenius product* of $W[\omega]$ and $G[j]$, the sum of all elements of $G[i_1, \ldots, i_{d-1}, j]$ such that $W[i_1, \ldots, i_{d-1}, \omega] = 1$. Figure 2 shows an example in the two dimensional case. Array $\mathrm{pred}[j, W]$ keeps track of the extension that maximizes the size of I'.

Once this is completed for all j's the information in the array pred can be used to retrieve an actual independent set. The following result summarizes the computational properties of Algorithm 1.

Theorem 1. *Algorithm 1 computes a maximum independent set of a weighted narrow LoS network G in time*

$$
O\left(n \, \left((\omega + 1) \, k^{\frac{d-1}{\omega}} \right)^{k^{d-1}} \, k^{(d-1) \cdot k^{d-2}} \right).
$$

PROOF. The proof that the Algorithm 1 is correct, i.e. it returns an optimum independent set, is by a simple reductio ad absurdum for each j (similar,

12

say, to the one described in [17, Theorem 15.1] in the context of the longest common subsequence problem). Denote by I an independent array of G of maximum array sum. There must be an independent array I' of $G[-(\omega-1) : n-1]$ such that:

$$||I|| = ||I||_{\omega+n} + ||I'||.$$

Furthermore $t(I'[n-\omega : n-1]) = h(I[n-\omega+1 : n])$, in other words the two independent arrays must be consistent. But then we are safe to assume that

$$||I'|| = \max_{W' \in \mathcal{F}_{G,n-1} : W' \vDash W} \text{MIS}[n-1, W'])$$

(where W is witnessed by $I[n-\omega+1 : n]$) for otherwise replacing I' by the independent set on the right hand side would give us a larger set for I contradicting its optimality. By the same token, $I[n-\omega+1 : n]$ must be a witness of the feasible array W maximizing $W[\omega] \cdot G[j] + \max_{W' \in \mathcal{F}_{G,n-1} : W' \vDash W} \text{MIS}(n-1, W')$. Therefore the independent set returned by Algorithm 1 is at least as large as I.

Moving to the complexity analysis, for each j, $|\mathcal{F}_{G,j}| = O((\omega+1)^{k^{d-1}})$ as the elements of this set are $\boldsymbol{k}^{d-1} \times \omega$ tables with at most a single non-zero entry in each row: each row can be filled in at most $\omega+1$ ways, and there are k^{d-1} rows: at most $(w+1)^{k^{d-1}}$ possibilities. Hence the double loop between line 10 and 16 is executed $O(n \, (\omega+1)^{k^{d-1}})$ times. The most expensive step in the aforementioned loop is in line 12. Let $t = \lceil \frac{k}{\omega} \rceil$. Each column of a feasible array W is a $d-1$ dimensional cube of side length k. The maximum number of non-zero elements in dimension one, say, is t. Furthermore the cube is a $d-1$ dimensional object, hence an obvious upper bound on the number of

13

non-zero elements of I_W is

$$t \times k^{d-2}.$$

From this, we claim, the maximum number of elements of $\mathcal{F}_{G,j}$ that are consistent with a given $W \in \mathcal{F}_{G,j-1}$, for each j, is at most

$$\binom{k^{d-1}}{t \cdot k^{d-2}}.$$

To see this notice that, starting from an arbitrary W, we get an element of $\mathcal{F}_{G,j-1}$ by chopping off the first column of W and adding an extra column at the other end. A column is a $d-1$ dimensional cube with k^{d-1} positions and there's at most $t \cdot k^{d-2}$ non-zero positions to be placed in that. Finally note that

$$\binom{k^{d-1}}{t \cdot k^{d-2}} \leq \left(k^{d-1}\right)^{t \cdot k^{d-2}}$$

and $t \leq \frac{k}{\omega} + 1$. Hence

$$\binom{k^{d-1}}{t \cdot k^{d-2}} \leq \left(k^{d-1}\right)^{\frac{k}{\omega}^{d-1}} \left(k^{d-1}\right)^{k^{d-2}} = \left(k^{\frac{d-1}{\omega}}\right)^{k^{d-1}} \left(k^{d-1}\right)^{k^{d-2}}.$$

By the counting argument above, the loop between line 10 and 16 can be completed in time

$$O\left(n \left((\omega + 1) \, k^{\frac{d-1}{\omega}}\right)^{k^{d-1}} \left(k^{d-1}\right)^{k^{d-2}}\right)$$

and the result follows. □

Extensions. The DP algorithm described in this section can be adapted to solve optimally a host of other optimization problems in narrow LoS net-

14

works. The smallest vertex covers or dominating sets, the largest triangle packings, or H-matchings and many other "hard" combinatorial structures can all be found in polynomial time by using obvious modifications of the strategy described above. Here we show that even problems of a slightly different nature can be solved optimally by our DP approach. An instance of the ADSSCHED problem defined in Section 1 can be encoded by an array G exactly like the MIS in LoS networks. The only difference is in the definition of feasible solution. Therefore Algorithm 1 can also be used to solve the ADSSCHED problems, provided the definition of $\mathcal{F}_{G,j}$ is slightly modified. In this case the elements of this set are $k \times \omega$ arrays W satisfying the following conditions

(i) $W \leq_a G[j - \omega + 1 : j]$

(ii) W contains at most one non-zero element in each row.

(iii) W contains at most l non-zero elements in each column.

Theorem 2. *Algorithm 1 solves* ADSSCHED *optimally in time* $O(n \, k^l \, (\omega + 1)^k)$.

PROOF. The correctness of the process follows from that of Algorithm 1 as proved in Theorem 1. As to the running time, the bound is derived in the same way as the running time bound in Theorem 1. Here $d = 2$, and the only other difference is in the maximum number of elements of $\mathcal{F}_{G,j}$ that are consistent with a given $W \in \mathcal{F}_{G,j-1}$, for each j: there are at most $\binom{k}{l}$ of them. The result follows. □

4. Semi-online Approximation Algorithms

The DP algorithm in Section 3 solves optimally the offline version of the MIS problem in narrow LoS networks and several other related problems, where the entire input is known in advance. This is unrealistic in various practical settings. For example if the time parameter n in the scheduling problem is large, possibly spanning a year or more, then it is likely the input evolves over time. In such sitations it may be desirable to take a different approach, aiming for *online* algorithmic solutions with good performance guarantees. In this section we study online algorithms [18] which are shown the input graph and corresponding embedding column by column. In particular, we consider semi-online algorithms which are given further partial information and in this case we allow a semi-online algorithm to observe the input up to a certain look-ahead distance denoted by Δ, i.e., to observe up to Δ columns. Semi-online algorithms and the ability to look-ahead have been considered in other problems [19, 20]. We show that we can achieve $(1 + \epsilon)$-approximation with a look-ahead distance dependent on $\epsilon > 0$. We state our main result in terms of the MIS problem, but the strategy can be applied to any of the optimization problems described at the end of Section 3.

Theorem 3. *There is a semi-online algorithm,* ONLINE_AS, *that for any $\epsilon > 0$ computes a feasible solution for the* MIS *problem in a narrow LoS network in dimension d that is a $(1 + \epsilon)$-approximation of the optimum, in time*

$$O\left(n\ \left((\omega + 1)\ k^{\frac{d-1}{\omega}}\right)^{k^{d-1}}\ k^{(d-1)\cdot k^{d-2}}\right)$$

16

and look-ahead distance at most $\left(1 + \frac{2k}{\epsilon^2}^{d-1}\right) \omega$.

The main idea of algorithm ONLINE_AS is similar to that of the EPTAS described in [11] for general LoS networks. The algorithm builds an independent set I of G in stages. Initially I is empty. Let $G[j_1 : j_2]$ where $j_1 < j_2$ describe the subgraph of the narrow LoS network G consisting of vertices which are embedded in the region

$$\underbrace{\{1, \ldots, k\} \times \ldots \times \{1, \ldots, k\}}_{d-1 \text{ times}} \times \{j_1, \ldots, j_2\}$$

and their induced edges. A *phase* in the algorithm starts by computing a maximum independent set I_0 in a subgraph of G consisting of some column j_0 and proceeds to compute a maximum independent set I_r, for $r \geq 1$, in $G[j_0 : j_0 + r(\omega - 1)]$ provided $|I_r| \geq (1+\epsilon)|I_{r-1}|$. Thus each I_r in the sequence satisfies $|I_r| \geq (1 + \epsilon)^r |I_0|$. In addition, using the structural properties of a LoS network embedding, we may infer that $|I_r| \leq k^{d-1} r$: at most r vertices can be added to I_r in every row, and there are k^{d-1} rows altogether. Let r^* be the least r for which

$$|I_{r^*+1}| < (1 + \epsilon)|I_{r^*}| \tag{1}$$

We refer to this as the *stopping point* of the current phase. When condition (1) is reached the process adds I_r to I, removes $G[j_0 : j_0 + (r^* + 1)(\omega - 1)]$ from G, and starts a new phase from column $j_0' = j_0 + (r^* + 1)(\omega - 1) + 1$ of the original graph. Note that, if $\Delta_i = 1 + (r_i^* + 1)(\omega - 1)$ is the number of columns of G processed during phase i, it follows from Theorem 1 that

17

phase i runs in time

$$O\left(\Delta_i \left((\omega + 1) k^{\frac{d-1}{\omega}}\right)^{k^{d-1}} k^{(d-1)\cdot k^{d-2}}\right).$$

In fact, for $\epsilon = 0$ the ONLINE_AS reduces to Algorithm 1 and the complexity bound above to the bound stated in Theorem 1. For $\epsilon > 0$ there may be different phases, and different phases work on vertex disjoint portions of G. Therefore $\sum_i \Delta_i = n$. The complexity bound stated Theorem 3 follows. The proof of the Theorem will be completed once we have argued about the performance guarantees of algorithm ONLINE_AS and given some upper bound on the look-ahead distance.

Claim 1. *For any $\epsilon > 0$, algorithm ONLINE_AS computes a feasible solution for the* MIS *problem in a narrow LoS network in dimension d that is a $(1+\epsilon)$- approximation of the optimum.*

PROOF. As in [11], the performance guarantees for algorithm ONLINE_AS can be proved by a simple inductive argument. Let G_{r^*+1} be the graph removed from G at the end of the first phase of the process and denote by $\overline{G_{r^*+1}}$ the remaining graph. If I' is the independent set obtained from applying the procedure to $\overline{G_{r^*+1}}$ then, by the inductive hypothesis, the size of the largest independent set in $\overline{G_{r^*+1}}$ is at most $(1 + \epsilon)|I'|$. Furthermore I_{r^*+1} is a maximum independent set in G_{r^*+1} and, by construction,

$$|I_{r^*+1}| \leq (1 + \epsilon)|I_{r^*}|.$$

But it is also true that the elements of I_{r^*} have no neighbour in $\overline{G_{r^*+1}}$ as, in

18

fact I_{r^*} is a subgraph of G_{r^*}. Therefore $I_{r^*} \cup I'$ is an independent set of G and a $(1 + \epsilon)$-approximation for the largest ones. $\square$

We complete our analysis of algorithm ONLINE_AS by proving a simple upper bound on r^* which depends on the inverse of ϵ. Inequality (2) below can then be used to quantify the trade-off between approximation quality and look-ahead window width.

Claim 2. *In each phase of the algorithm* ONLINE_AS

$$r^* \leq 1 + \frac{2k^{d-1}}{\epsilon^2}. \tag{2}$$

PROOF. Throughout a phase we have $|I_r| \geq (1 + \epsilon)^r |I_0|$. Thus r^* is bounded above by the smallest positive integer r for which

$$k^{d-1}r < (1 + \epsilon)^r. \tag{3}$$

By the binomial theorem

$$(1 + \epsilon)^r > \binom{r}{2}\epsilon^2 = \frac{r^2 - r}{2}\epsilon^2.$$

Hence inequality (3) is satisfied if

$$k^{d-1}r < r^2\frac{\epsilon^2}{2} - r\frac{\epsilon^2}{2}$$

which is equivalent to

$$k^{d-1} + \frac{\epsilon^2}{2} < r\frac{\epsilon^2}{2}$$

and the result follows. $\square$

19

Additional Remarks. The same framework can be used to devise semi-online $(1 + \epsilon)$-approximation heuristics for finding smallest vertex covers or dominating sets, the largest triangle packings, or H-matchings. The analysis of the approximation performance is largely unchanged. The run-time in each case is affected by the run time of the specific DP algorithm used to complete each phase. The approximation heuristics are semi-online since at any moment in time we never work on more than $r^*\omega$ columns of the input data.

5. Approximation Algorithms for Unrestricted LoS Networks

In this section we show how the DP approach described in Section 3 can be exploited to define an effective approximation strategy for the MIS in arbitrary d-dimensional LoS networks, for $d \geq 2$. To avoid cluttering the presentation we first describe and analyze the algorithm for the special case $d = 2$. Then we outline the modifications necessary to extend the algorithm to the general d-dimensional case. Definitions and notations related to approximation algorithms can be found in standard textbooks like [21] or the more recent [22]. In particular all algorithms described in this section run in time that is polynomial in the input size.

In 2-dimensions, we are given a LoS network (embedded in $\mathbb{Z}_n^2$) with range parameter $\omega > 2$. The main idea is to split the input data into *strips*, each being a narrow LoS network, apply the DP algorithm to each strip, and then combine the solutions obtained for the strips into a solution for the whole instance.

In what follow let $k = \omega - 1$ and let $G_{[i]}$ be the *strip* formed by the vertices of G embedded in rows $ki + 1, \ldots, k(i + 1)$ of $\mathbb{Z}_n^2$, for $i \in \{0, \ldots, n/k - 1\}$

(assume k divides n for simplicity). Parameter i in this context is the *strip index.*
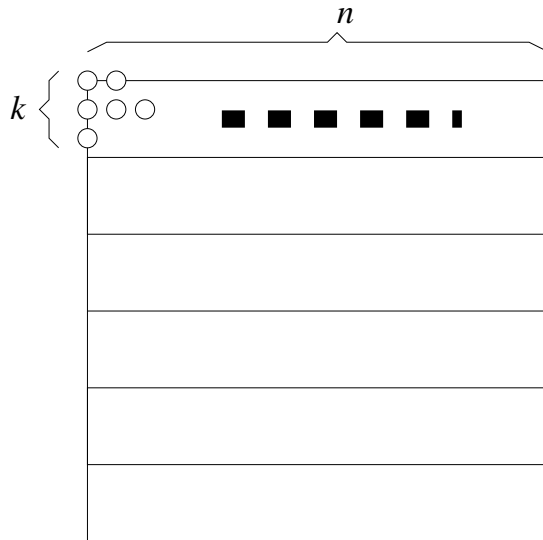

Figure 3: Splitting G into n/k strips ($k = 3$, in the given example, array cells represented as small circles).

Clearly two nodes from different strips with indices having the same parity cannot be adjacent as even if they share a line of sight they are far from each other. Therefore the union of a collection of independent sets found in all odd (resp. even) indexed strips is an independent set of the whole network. The sought approximation algorithm, which we call StripIndependentSet, returns H, the largest of these two sets.

Theorem 4. *For any fixed ω independent of n,* StripIndependentSet *is a 2-approximation algorithm for the* MIS *in a 2-dimensional LoS network.*

PROOF. Let's call Odd (Even) the collection of all odd (even) indexed strips. We can use dynamic programming to find an optimal independent set in each

strip. Let DP(Odd) (resp. DP(Even)) be the independent set found using
Algorithm 1 on each Odd (resp Even) strip. Let $\mathcal{I}$ be an independent set of
maximum size in the whole graph. $\mathcal{I} \cap$ Odd is an independent set of Odd so
it must be

$$|\text{DP(Odd)}| \geq |\mathcal{I} \cap \text{Odd}|$$

and

$$|\text{DP(Even)}| \geq |\mathcal{I} \cap \text{Even}|$$

Hence

$$|\mathcal{I}| = (|\mathcal{I} \cap \text{Odd}|) + (\mathcal{I} \cap \text{Even}|) \leq |\text{DP(Odd)}| + |\text{DP(Even)}| \leq 2 \cdot |H|.$$

The process requires $O(n/k)$ DP computations, each running in time

$$O\left(n \ \left((\omega + 1) \ k^{\frac{1}{\omega}} \right)^k \ k \right).$$

(this comes from Theorem 1 substituting $d = 2$). The overall run time is
therefore

$$O\left(n^2 \ (\omega + 1)^k \ k^{\frac{k}{\omega}} \right).$$

$\square$

Generalization to d dimensions. Two of the authors [11] already provides a 2-
approximation algorithm for the MIS problem in 2-dimensional LoS networks.
The main advantage of the approach described above lies in the fact that
algorithm StripIndependentSet can be generalized to arbitrary dimension
$d > 2$. The general strategy is unchanged but the notion of Odd (resp.

Even) strip is slightly more elaborate. As in Section 3, it is convenient to think of the network nodes as the elements of a d-dimensional table. In this context a *strip* is a collection of elements

$$G[k(i_1 - 1) + j_1, \ldots, k(i_{d-1} - 1) + j_d, i_d]$$

where $j_h \in \{1, \ldots, k\}$ and the vector $(i_1, \ldots, i_{d-1})$ satisfies $i_h \in \{1, \ldots, n/k\}$, for $h \in \{1, \ldots, d-1\}$ (whereas $i_d \in \{1, \ldots, n\}$). The vector $\boldsymbol{i} = (i_1, \ldots, i_{d-1})$ is the strip *index*. A vertex belongs to an *odd* (resp. *even*) strip if its strip index satisfies:

$$\sum_{h=1}^{d-1} i_h \mod 2 = 1 \qquad (\text{resp. } 0).$$

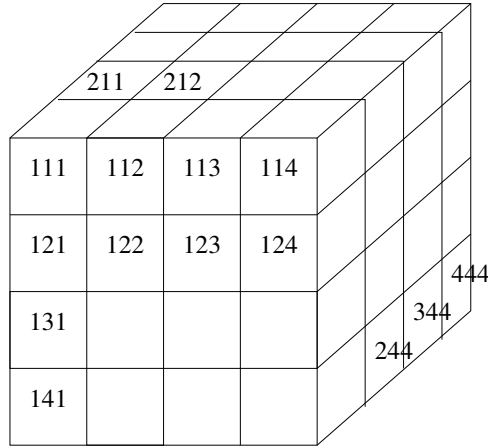Figure 4 attempts to give an idea of the partitioning for $d = 4$.

Figure 4: Splitting G into strips. Here is one of the n 3-dimensional "bases" (each small cube is labelled by the corresponding triple (i_1, i_2, i_3)). For $d = 2$, one dimension is split into n/k intervals. For $d = 4$, three dimensions are split into $(n/k)^3$ cubes of side size k. DP gives an optimal solution in each $k^3 \times n$ strip.

330 **Claim 3.** *Vertices belonging to different strips whose indices have the same*

23

parity are not connected by an edge in G.

It follows from the claim above that algorithm StripIndependentSet returns an independent set in any LoS network embedded in d dimensions and the following result complete our argument.

Theorem 5. *For any fixed ω and d independent of n,* StripIndependentSet *is a 2-approximation algorithm for the* MIS *in a d-dimensional LoS network.*

PROOF. The same argument used to prove Theorem 4 applies. This time $O((n/k)^{d-1})$ DP computations are needed and each of them requires time

$$O\left(n \ \left((\omega+1) \ k^{\frac{d-1}{\omega}}\right)^{k^{d-1}} \ k^{(d-1)\cdot k^{d-2}}\right).$$

Therefore the running time of the process is:

$$O\left(n^d \ (\omega+1)^{k^{d-1}} \ k^{\frac{(d-1)k^{d-1}}{\omega}} \ k^{(d-1)\cdot(k^{d-2}-1)}\right).$$

$\square$

6. Polynomial Time Approximation Schemes

The DP approach in Section 3 can also be exploited to obtain an EPTAS for the MIS problem in general LoS networks, for any $d \geq 2$. As in the previous section we first present the idea for the case $d = 2$. In this section again $k = \omega - 1$.

The algorithm works on the given network (which, as usual in this paper, is provided with its embedding in $\mathbb{Z}_n^2$) decomposed into strips, as in Figure

24

3, however we need one additional concept. Let h be a positive integer. Its value will be fixed later on in our analysis, but for now we require that h be a fixed constant independent of n. A *block* is a collection of contiguous strips (note that the number of rows in each block is a multiple of k). For each $i \in \{0, \ldots, h\}$, let $\mathcal{B}_{h,i}$ be the partition of G into blocks such that the top one contains $i \times k$ rows, and all the others (except perhaps the last one) contain $h \times k$ rows. Successive blocks are separated by a single strip. Let $\partial \mathcal{B}_{h,i}$ be the union of these "excluded" strips. Let B be an arbitrary block of $\mathcal{B}_{h,i}$. Since the product $h \times k$ is independent of n, a maximum independent set $\mathcal{I}_B$ in B can be found in polynomial time using Algorithm 1. The set

$$\mathcal{I}_i = \bigcup_{B \in \mathcal{B}_{h,i}} \mathcal{I}_B$$

is a maximum independent set of $\mathcal{B}_{h,i}$. The algorithm returns the largest among $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_h$. Let's call $\mathcal{U}$ such set. Let $\mathcal{I}$ be a maximum independent set of the whole network. A key property of independent sets is that a maximum independent set in any strip S of G cannot have less than $|\mathcal{I} \cap V(S)|$ vertices (as the vertices in each strip in isolation are less constrained than when they are considered as part of the whole graph). Also, we can write

$$|\mathcal{I}| = \sum_{i=0}^{h} |\mathcal{I} \cap V(\partial \mathcal{B}_{h,i})|.$$

But then by a simple counting argument there must be $\hat{\imath} \in \{0, \ldots, h\}$ such that $|\mathcal{I} \cap V(\partial \mathcal{B}_{h,\hat{\imath}})| \leq |\mathcal{I}|/(1+h)$. This implies that a maximum independent
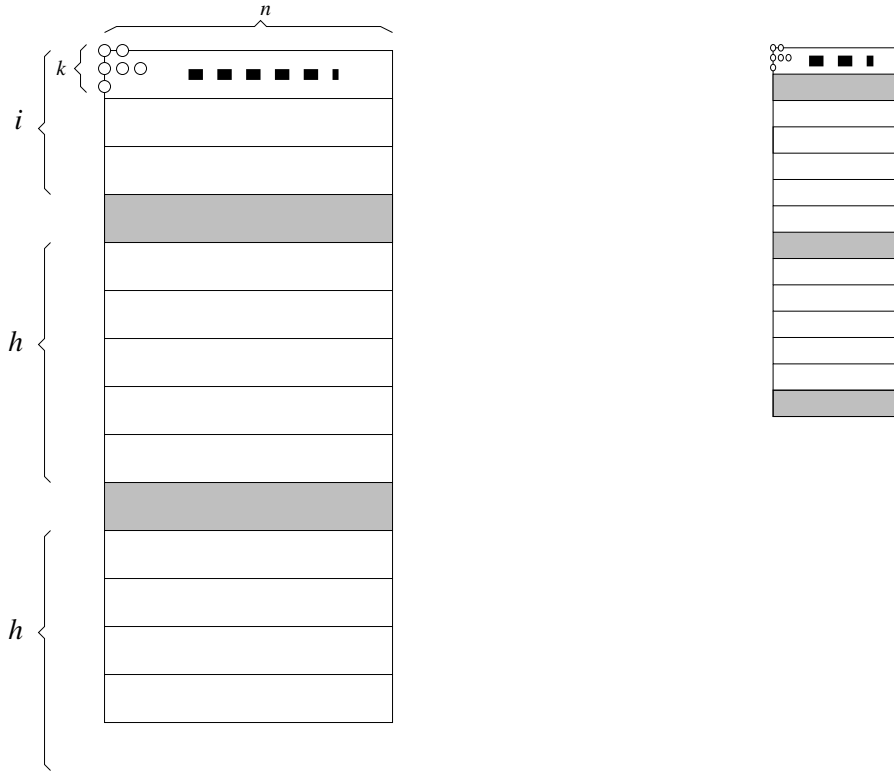
Figure 5: Splitting G into blocks. The larger picture on the left hand side describes the top part of $\mathcal{B}_{5,3}$. The smaller picture on the right presents a similar schematic representation of $\mathcal{B}_{5,1}$. In both cases the greyed strips belong to the union of the excluded strips (i.e. $\partial\mathcal{B}_{5,3}$ and $\partial\mathcal{B}_{5,1}$, respectively).

set $\mathcal{I}_{\hat{\mathbf{1}}}$ in $\mathcal{B}_{h,\hat{\mathbf{1}}}$ (which will be eventually found by the algorithm) mu1¡st satisfy

$$|\mathcal{I}_{\hat{\mathbf{1}}}| \geq |\mathcal{I} \cap V(\mathcal{B}_{h,\hat{\mathbf{1}}})| = |\mathcal{I}| - |\mathcal{I} \cap V(\partial\mathcal{B}_{h,\hat{\mathbf{1}}})| > \frac{h}{1+h}|\mathcal{I}|.$$

Thus we have

$$\frac{|\mathcal{I}|}{|\mathcal{U}|} \leq \frac{|\mathcal{I}|}{|\mathcal{I}_{\hat{\mathbf{1}}}|} \leq 1 + \frac{1}{h}$$

and the $(1 + \epsilon)$-approximation is obtained setting $h = \lceil 1/\epsilon \rceil$.

For each $i \in \{0, \ldots, h\}$, the MIS can be solved exactly in each block of

the given partition in time

$$O\left(n\ \left((\omega+1)\ (hk)^{\frac{1}{\omega}}\right)^{hk}\ (hk)\right)$$

and there are $O(n/(hk))$ blocks. The overall running time is therefore

$$O\left((h+1)\ n^2\ (\omega+1)^{hk}\ (hk)^{\frac{hk}{\omega}}\right)$$

as the process will need to try all $h+1$ partitions $\mathcal{B}_{h,\hat{1}}$.

We have proved the following:

Theorem 6. *There is a polynomial time approximation scheme for the* MIS *problem in 2-dimensional LoS networks.*

6.1. Arbitrary dimension $d > 2$

A key feature of the approximation scheme for $d = 2$ is that the collection of strips $\cup_{i=0}^{h}V(\partial\mathcal{B}_{h,i})$ is a partition of the given graph vertex set. For $d > 2$ the construction needs to be a bit careful. Figure 6 provides a diagrammatic picture of a possible construction for $d = 3$.

Let b be an integer less than d. We say that a cube isomorphic to

$$\underbrace{\{1,\ldots,x_1\} \times \ldots \times \{1,\ldots,x_b\}}_{b \text{ terms}} \times \{1,\ldots,n\} \times \ldots \times \{1,\ldots,n\}$$

(where each of the $x_i \in \{1,\ldots,k\}$) is *a size n cube narrow with respect to b of its dimensions.* In what follows a *size n, b-narrow, d-dimensional* LoS network is a LoS network whose nodes can be embedded in a size n cube that is narrow with respect to b of its dimensions.
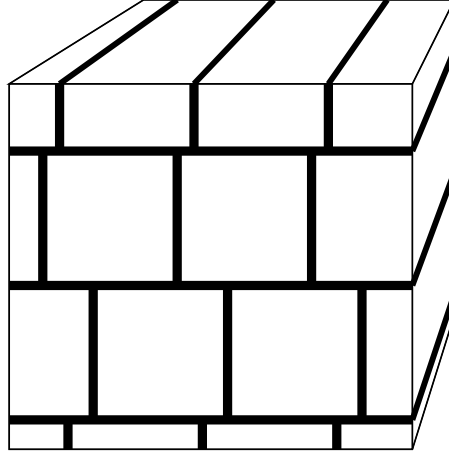
27

Figure 6: The whole LoS network partitioned into narrow LoS networks, for $d = 3$. Note that the sizes of the narrow cubes may vary. The black portions represent the vertices that are part of the excluded strips.

Claim 4. *Let d and b be fixed positive integers with $b < d$, and n be an arbitrary integer. For any $\epsilon > 0$, a $(1 + \epsilon)^{d-b-1}$-approximation for the* MIS *in a size n, b-narrow, d-dimensional LoS network can be found in time polynomial in n but exponential in ϵ^{-1}.*

PROOF. The claim can be proved by induction on $d-b$. Let G be an arbitrary size n, b-narrow, d-dimensional LoS network. If $d - b \leq 1$ we can use the dynamic programming strategy described in Section 3 to solve the problem exactly. For arbitrary $d - b$, without loss of generality assume that G is NOT narrow with respect to dimension one. Any part of G that spans the full length of the unrestricted dimensions, and is also narrow with respect to dimension 1 is a size n, $b + 1$ narrow, d dimensional LoS network. By the inductive hypothesis, the largest independent set of such network can be approximated within $(1 + \epsilon)^{d-b-2}$. For any given ϵ let $h = \lceil 1/\epsilon \rceil$. Let $i \in \{0, \ldots, h\}$. Given G, we partition it into blocks containing all nodes

28

whose co-ordinates in $\mathbb{Z}_n^d$ have the first element in the set $\{1, \ldots, ki\}$ or of the form

$$ki + (k + kh)(j - 2) + l$$

where $j \in \{2, \ldots, 2 + \lfloor (n - ki)/(k + kh) \rfloor\}$ and $l \in \{1, \ldots, kh\}$, except when j takes its largest value (in that case the largest value for l is the remainder of the integer division between $n - ki$ and $k + kh$). This defines a partition $\mathcal{B}_{h,i}$ whose blocks are size n, $b + 1$ narrow, d-dimensional LoS networks. For one choice of i, which we denote again by $\hat{\imath}$, we must have

$$|\mathcal{I} \cap V(\partial \mathcal{B}_{h,\hat{\imath}})| \leq \frac{|\mathcal{I}|}{h + 1}$$

(where $\mathcal{I}$ is a maximum independent set of the whole network). Therefore, reasoning like in the 2-dimensional case we have

$$|\mathcal{I}| \leq \left(1 + \frac{1}{h}\right) |\mathcal{I}_{\hat{\imath}}|$$

where $\mathcal{I}_{\hat{\imath}}$ is the maximum independent set of $\mathcal{B}_{h,\hat{\imath}}$. This independent set is the union of disjoint sets $\mathcal{I}_{\hat{\imath}} \cap \mathcal{B}_{h,\hat{\imath}}(j)$. By the inductive hypothesis there is an algorithm that finds a set $\mathcal{U}_{\hat{\imath}}(j)$ in the jth block of $\mathcal{B}_{h,\hat{\imath}}$ that satisfies

$$|\mathcal{I}_{\hat{\imath}} \cap \mathcal{B}_{h,\hat{\imath}}(j)| \leq (1 + \epsilon)^{d-b-2} |\mathcal{U}_{\hat{\imath}}(j)|$$

Therefore
$$|\mathcal{I}| \leq \left(1 + \frac{1}{h}\right)(1 + \epsilon)^{d-b-2} \sum_j |\mathcal{U}_{\hat{\imath}}(j)|$$

and the claim follows. $\qquad\square$

Theorem 7. *There is a polynomial time approximation scheme for the* MIS *problem in d-dimensional LoS networks.*

PROOF. For any fixed $\epsilon > 0$, define $\epsilon' = (1 + \epsilon)^{\frac{1}{d-1}} - 1$. By Claim 4 (with $b = 0$) there is an algorithm that returns an independent set whose size is at least $(1 + \epsilon')^{1-d}$ that of a largest independent set in any given LoS network. As to the running time, argument is similar to the 2-dimensional case and based again on using the analysis in Theorem 1 as a building block. For each tuple $i_1, \ldots, i_{d-1}$, the MIS can be solved exactly in each block of the given partition in time

$$O\left(n\left((\omega + 1)(hk)^{\frac{d-1}{\omega}}\right)^{(hk)^{d-1}}(hk)^{(d-1)\cdot(hk)^{d-2}}\right)$$

and there are $O((n/(hk))^{d-1})$ blocks. $\qquad\square$

Further Comments. It is perhaps instructive to compare the algorithm presented in this section with the approximation scheme described in [11] for the MIS in general d-dimensional LoS networks. The algorithm in that paper runs in time

$$O(n^d(f_{d,\omega}(\epsilon))^{d(f_{d,\omega}(\epsilon))^d/\omega})$$

where

$$f_{d,\omega}(\epsilon) = \frac{2\cdot(d+1)!}{\omega}\left(\frac{\omega}{\epsilon - \epsilon^2/2}\right)^{d+1}.$$

Since $f_{2,\omega}(\epsilon) = \frac{12\cdot\omega^2}{(\epsilon-\epsilon^2/2)^3}$, for $d = 2$, the algorithm running time reduces to essentially

$$O(n^2\ \omega^{9(4\omega/\epsilon^2)^3}\ (\frac{12}{\epsilon - \epsilon^2/2})^{36(2\omega/\epsilon^2)^3})$$

which is much slower than the bound in Theorem 6, particularly for small ϵ and moderate ω.

The approximation scheme described in this section is quite general and can be applied to several optimization problems when the input is a 2-dimensional LoS network that is presented along with its embedding in $\mathbb{Z}_n^2$. In particular, simply browsing through [7] Vertex Cover, Min Dominating Set, Min Edge Dominating Set, Max Triangle Packing, Max H-matching, Max Tile Salvage can all be solved to within $1 + \epsilon$ of the optimum in a 2-dimensional LoS network, if ω is a fixed constant independent of n.

7. Conclusions

In this paper we study the maximum independent set problem on narrow LoS networks. We propose an approach that solves this optimazion problem exactly in polynomial time on narrow LoS network, presented with their d-dimensional embedding. We also describe how such algorithm can be used as a subroutine in a semi-online process that is guaranteed to return a heuristic solution that is guaranteed to be only at most a factor $1 + \epsilon$ away from optimality, for any $\epsilon > 0$, in a 2-approximation algorithm, and in a novel polynomial time approximation scheme, all of these for the MIS problem in arbitrary d-dimensional networks, for fixed ω independent of n.

We believe that the algorithmic ideas described here can be generalized and applied to other optimisation problems on LoS networks.

31

References

[1] P. Sangha, P. W. H. Wong, M. Zito, Independent sets in restricted line of sight networks, in: A. Fernández Anta, T. Jurdzinski, M. A. Mosteiro, Y. Zhang (Eds.), Algorithms for Sensor Systems, 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Vol. 10718 of Lecture Notes in Computer Science, Springer Verlag, 2017, pp. 211–222.

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

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

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

[5] A. Czumaj, X. Wang, Communication problems in random line-of-sight ad-hoc radio networks, in: International Symposium on Stochastic Algorithms, Springer, 2007, pp. 70–81.

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

[7] M. R. Garey, D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness (1979).

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

[9] L. Lovász, M. D. Plummer, Matching Theory, Vol. 29 of Annals of Discrete Mathematics, North Holland, 1986.

[10] D. Peterson, Gridline graphs: a review in two dimensions and an extension to higher dimensions, Discrete Applied Mathematics 126 (2–3) (2003) 223–239.

[11] P. Sangha, M. Zito, Independent sets in Line of Sight networks, Discrete Applied MathematicsIn Press. DOI: `https://doi.org/10.1016/j.dam.2019.03.029` Published on-line 30/04/2019.

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

[13] R. K. Jallu, G. K. Das, Improved algorithm for maximum independent set on unit disk graph, in: S. Govindarajan, A. Maheshwari (Eds.), Algorithms and Discrete Applied Mathematics; Third International Conference CALDAM 2016, Vol. 9602 of Lecture Notes in Computer Science, Springer Verlag, 2016, pp. 212–223.

[14] R. Bellman, A. O. Esogbue, I. Nabeshima, Mathematical Aspects of Scheduling and Applications, Elsevier, 2014.

[15] M. Milanič, P. Muršič, M. Mydlarz, Induced embeddings into hamming graphs, in: K. G. Larsen, H. L. Bodlaender, J.-F. Raskin (Eds.), 42nd

International Symposium on Mathematical Foundations of Computer Science (MFCS 2017), Leibniz International Proceedings in Informatics, Dagstuhl Publishing, Germany, 2017, pp. 28:1–28:15.

[16] S. Albers, Online algorithms: a survey, Mathematical Programming 97 (1-2) (2003) 3–26.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, M.I.T. Press, 2009.

[18] A. Borodin, R. El-Yaniv, Online computation and competitive analysis, Cambridge University Press, 2005.

[19] H. Kellerer, V. Kotov, M. G. Speranza, Z. Tuza, Semi on-line algorithms for the partition problem, Oper. Res. Lett. 21 (5) (1997) 235–242.

[20] S. Albers, M. Hellwig, Semi-online scheduling revisited, Theor. Comput. Sci. 443 (2012) 1–9.

[21] D. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, 1997.

[22] D.-Z. Du, K.-I. Ko, X. Hu, Design and Analysis of Approximation Algorithms, Springer, 2011.