Independent Sets in Restricted Line of Sight Networks

Pavan Sangha, Prudence W.H. Wong, and Michele Zito

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

Abstract. Line of Sight (LoS) networks were designed to model wireless networks in settings which may contain obstacles restricting visibility of sensors. A graph $G = (V, E)$ is a 2-dimensional LoS network if it can be embedded in an $n \times k$ rectangular point set such that a pair of vertices in E are adjacent if and only if the embedded vertices are placed on the same row or column and are at a distance less than ω. We study the Maximum Independent Set (MIS) problem in restricted LoS networks where k is a constant. It has been shown in the unrestricted case when $n = k$ and $n \to \infty$ that the MIS problem is NP-hard when $\omega \geq 2$ is fixed or when $\omega = O(n^{1-\epsilon})$ grows as a function of n for fixed $0 < \epsilon < 1$. In this paper we develop a dynamic programming (DP) algorithm which shows that in the restricted case the MIS problem is solvable in polynomial time for all ω. We then generalise the DP algorithm to solve three additional problems which involve two versions of the Maximum Weighted Independent Set (MWIS) problem and a scheduling problem which exhibits LoS properties in one dimension. We use the initial DP algorithm to develop an efficient polynomial time approximation scheme (EPTAS) for the MIS problem in restricted LoS networks. This has important applications, as it provides a semi-online solution to a particular instance of the scheduling problem. Finally we extend the EPTAS result to the MWIS problem.

1 Introduction

LoS network. A wireless network typically consists of wireless devices that use data connections to communicate wirelessly. Geometric graphs often provide a good model for such networks with vertices representing wireless devices, and edges representing communication between pairs of devices. Various types of geometric graphs have been proposed to model wireless sensor networks. The disk intersection model [5] is a commonly used one. Vertices are placed in some physical space with the communication range for a vertex represented by a circle of some prescribed radius. Edges are formed between pairs of vertices whose circles overlap. A benefit of this model is its ability to capture the constraint of communication range restriction. This restriction implies that vertices should be close in distance in order to communicate. Another constraint exhibited by real world wireless networks are line of sight restrictions, often due to the presence of a large number of obstacles, like those often found in urban settings

for example. With this restriction vertices can only communicate if they are both close in distance and also share a direct line of sight. 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. [9] introduced the notion of random Line of Sight networks to provide a model that incorporates both. Their work focused on structural properties of the model focusing on connectivity. Since then connectivity in higher dimensions, percolation and communication problems within the same model have been studied [3, 6, 8].

For positive integers k and n let $\mathbb{Z}_{n,k} = \{(x, y) : x \in \{1, \ldots, n\},\ y \in \{1, \ldots, k\}\}$ denote the underlying point set. For two points $p_1 = (x_1, y_1), p_2 = (x_2, y_2) \in \mathbb{Z}_{n,k}$, let $d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$ denote the Manhatten distance between p_1 and p_2. The distance metric we use differs from the one used in [9] which relies on the underlying point set being toroidal to ease calculations. We say points p_1 and p_2 share a line of sight if $x_1 = x_2$ or $y_1 = y_2$. A graph $G = (V, E)$ is said to be a Line of Sight (LoS) network with parameters n, k and ω if there exists an embedding $f_G : V \to \mathbb{Z}_{n,k}$ such that $\{u, v\} \in E$ if and only if $f_G(u)$ and $f_G(v)$ share a line of sight and $d(f_G(u), f_G(v)) < \omega$. We refer to ω as the range parameter which is used to model the communication range restriction. The range parameter can be a constant or grow monotonically as a function of the parameters n or k.

MIS. We focus on the Maximum Independent Set (MIS) problem [7] in LoS networks. The MIS in a graph can be seen as a measure of network dispersion and independent sets share links with other important graph properties like vertex covers [7]. For $k \times n$ line of sight networks, large independent sets could help in covering scenarios like the following: "New York has many more streets than avenues. On parade days the police may want to dominate all junctions by also maximising presence". In general graphs finding the largest independent set is NP-hard [10] and even finding good approximation solutions are difficult [11]. For LoS networks in particular, if $\omega = 2$ or n, it is not difficult to see that the problem can be solved optimally. Sangha and Zito [12] showed on the other hand for $\omega = O(n^{1-\epsilon})$ where $0 < \epsilon < 1$ is fixed the problem is NP-hard. They also provide approximation results, showing the problem admits a 2-approximation for any ω and an efficient polynomial time approximation scheme (EPTAS) [4] for constant ω.

Additional applications. The abstract problem we study also finds application [2] in the following scheduling problems. Suppose an advertisement company manages advertisements from some number of clients (cf. k) over a long period of discrete time (cf. n). At any time advertisements of some subset of clients are available to be aired but the company can only select a certain number (cf. 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 time (cf. ω). The goal of the company is to schedule the airing of these advertisements satisfying the constraints and maximise the number of advertisements aired (cf. MIS). This

problem has one slight difference from the LoS problem in the sense that the restriction with ω only applies on one dimension (the time dimension) but not the other (the client dimension). Nevertheless, as to be showed later, the solution we develop can be adapted to solve this problem.

Our contribution. In this paper we study the MIS problem on restricted LoS networks. We focus on the restricted case that parameter k is a constant and $k < \omega$, in which case we show in Section 3 that the problem becomes polynomial time solvable via a dynamic programming (DP) algorithm. We also show in Section 4 that the DP algorithm can be extended to solve (i) the maximum weighted independent set problem (MWIS) on restricted LoS networks, (ii) MWIS on LoS networks with $k > \omega$ but both k and ω being constants, and (iii) the advertisement scheduling problem mentioned above. We then in Section 5 apply the DP algorithm to develop an EPTAS which improves the EPTAS in [12] by showing that when k is restricted to being constant the algorithm no longer requires ω to be constant. In addition we show that the EPTAS leads to a semi-online algorithm [1] with performance ratio $(1+\epsilon)$, which requires a look-ahead distance dependent on ϵ. This gives a semi-online algorithm for the scheduling problem in the case where $k < \omega$ and $l = 1$.

2 Problem Definitions and Preliminaries

The dynamic programming algorithm uses arrays consisting of $0, 1$ elements. An array of size $x \times y$ consists of x rows and y columns. We start by introducing some notations.

- For any array A, a sub-array $A[i..j]$ contains columns i, $i+1$, $\cdots$, j.
- For any two arrays A_1, A_2 of size $x \times y$, we say that A_1 *agrees with* A_2, denoted by $A_1 \leq_a A_2$, if $A_1[i,j] \leq A_2[i,j]$ for all $1 \leq i \leq x$ and $1 \leq j \leq y$.
- For any array A, we denote by $h(A)$ the "head" subarray of A containing all but the last column of A; and $t(A)$ the "tail" subarrary of A containing all but the first column of A. That is, $h(A)$ and $t(A)$ have $y-1$ columns if A has y columns.
- A_1 is said to be *tail-aligned* with A_2 if $t(A_1)$ is the same as $h(A_2)$, i.e., $t(A_1) \equiv h(A_2)$. In this case, we say A_2 is *head-aligned* with A_1.
- Let the *column sum* of an array A at column y be $\mathrm{cs}(A, y) = \sum_x A[x, y]$

Given a LoS network G in $\mathbb{Z}_{n,k}$ let $\mathrm{array}(G)$ satisfies $\mathrm{array}(G)[i,j] = 1$ if and only if location $(i,j) \in \mathbb{Z}_{n,k}$ in the LoS embedding of G contains a vertex, otherwise $\mathrm{array}(G)[i,j] = 0$ Figure 1 provides an example. Given $\mathrm{array}(G)$ of size $k \times p$ where $\omega \leq p$, an independent array I of $\mathrm{array}(G)$ is any array of size $k \times p$ satisfying (i) $I \leq_a \mathrm{array}(G)$ and (ii) I contains at most one 1 in each column and for any row i and distinct columns j_1, j_2, if $I[i,j_1] = 1$ and $I[i,j_2] = 1$ it must be the case that $|j_1 - j_2| \geq \omega$. We refer an independent array W specifically of size $k \times \omega$ as a "feasible array". Since any feasible array has exactly ω columns it contains at most one 1 per column but also at most one 1 per row. We denote by $\mathcal{F}$ the set of all feasible of arrays of size $k \times \omega$.

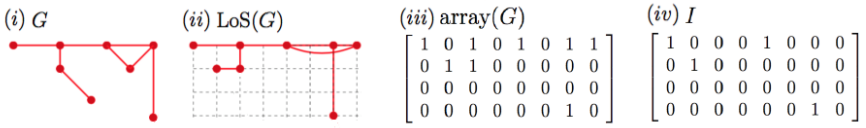

Fig. 1. Figure (i) is a graph G and Figure (ii) is its LoS embedding in $\mathbb{Z}_{8,4}$ with $\omega = 4$. Figure (iii) represents the array layout of G and Figure(iv) is an independent array of largest array sum, corresponding to the largest independent set in the graph G.

There is a clear connection between an independent array I of array(G) and an independent set $\mathcal{I}$ of the LoS network G. More precisely given a set $\mathcal{I} \subseteq V$ of a LoS network $G = (V, E)$ embedded in $\mathbb{Z}_{n,k}$ consider the array I satisfying $I[i, j] = 1$ if and only if location $(i, j) \in \mathbb{Z}_{n,k}$ contains a vertex in $\mathcal{I}$. We observe the following:

1. $\mathcal{I}$ is an independent set of G if and only if I is an independent array of the array G.
2. $|\mathcal{I}| = \sum_{i=1}^{k} \sum_{j=1}^{n} I[i, j]$.

We refer to the quantity $\sum_{i=1}^{k} \sum_{j=1}^{n} I[i, j]$ as the array sum of I which we denote by $as(I)$. Using the two observations above it follows that finding the maximum independent set in a LoS network G embedded in $\mathbb{Z}_{n,k}$ is equivalent to finding the independent array of the array G with the largest array sum, we refer to such an array as a largest independent array. In Section 3 we show how our DP algorithm finds the largest independent array of array(G) by scanning the feasible arrays of array(G).

3 Dynamic Programming

3.1 Algorithm

In this section we present a dynamic programming (DP) algorithm for finding a largest independent array. For simplicity we refer to array(G) as G. It will be clear from the context when referring to the LoS network G instead of the array G. Given the array G of size $k \times n$, for the purposes of the DP algorithm we prepend G with ω columns consisting entirely of 0's. We index these columns from $-(\omega - 1), \ldots, 0$. Thus the input array G is of size $k \times (\omega + n)$. The DP algorithm works by sequentially computing the array sums of $G[-(\omega - 1)..j]$ in G. The process keeps track of various sums MIS(j, W) depending on the independent set choices in the right-most column of $G[-(\omega - 1)..j]$. For $j = 0, \ldots, n$ let $\mathcal{F}_{G,j} \subseteq \mathcal{F}$ be the set of feasible arrays W satisfying $W \leq_a G[j - \omega + 1..j]$. Note that in particular any independent array I of $G[-(\omega - 1)..j]$ for $1 \leq j \leq n$ satisfies $I[j - \omega + 1..j] \equiv W$ for some $W \in \mathcal{F}_{G,j}$.

Algorithm 1 describes how to compute the array sum on an independent array of $G[-(\omega - 1)..j]$ from information about $G[-(\omega - 1)..(j - 1)]$. More specifically

Algorithm 1 Computing the size of the largest independent array in G

Initialise: $\mathrm{MIS}(0, \vec{0}) = 0$, where $\vec{0}$ is the $k \times \omega$ array of all 0's.
for $j = 1, \ldots, n$ **do**
  for $W \in \mathcal{F}$ **do**
    if $W \in \mathcal{F}_{G,j}$ **then**
      $\mathrm{MIS}(j, W) = \max_{W' \in \mathcal{F}_{G,j-1} : t(W') \equiv h(W)} \mathrm{MIS}(j-1, W') + \mathrm{cs}(W, \omega)$
    else
      $\mathrm{MIS}(j, W) = 0$
    end if
  end for
  $\mathrm{MIS}(j) = \max_{W \in \mathcal{F}_{G,j}} \mathrm{MIS}(j, W)$
end for

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

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

Fig. 2. Figure (i) shows the first 8 columns of an array G and the independent array I' of $G[1..8]$ has 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 $t(W') \equiv h(W)$ and that I can be obtained from I' by appending the last column of W to I'.

if $W \in \mathcal{F}_{G,j}$ we try to extend the independent arrays in $G[-(\omega-1)..(j-1)]$ to independent arrays in $G[-(\omega-1)..j]$. Let I' be an independent array in $G[-(\omega-1)..(j-1)]$ such that $I[(j-\omega)..(j-1)] \equiv W'$ for some $W' \in \mathcal{F}_{G,j-1}$ and $t(W') \equiv h(W)$. By considering the next column of G, we extend I' to an independent array I of $G[-(\omega-1)..j]$ which satisfies $I[j-\omega+1..j] \equiv W$. There are two cases for the sum $\mathrm{cs}(W, \omega)$:

1. $\mathrm{cs}(W, \omega)$ is 0 meaning that there is no entry in the last column of W. Then $\mathrm{as}(I) = \mathrm{as}(I')$ and hence the array sum for our independent set does not increase;
2. $\mathrm{cs}(W, \omega)$ is 1 meaning that we can increase the array sum of our independent array by 1; note that since W is feasible, $\mathrm{cs}(W, \omega)$ is at most 1.

The new independent array can be obtained by extending the one for $G[-(\omega-1)..(j-1)]$ by appending the ω-th column of W. Figure 2 shows an example.

3.2 Correctness

In this section we prove the correctness of Algorithm 1. We first prove in Lemma 1 that it is sufficient to consider $\mathcal{F}_{G,j}$ and then the main result in Theorem 2.

Lemma 1. *For $j = 0, \ldots, n-1$ for each $W'_1 \in \mathcal{F}_{G,j}$ there exists $W_1 \in \mathcal{F}_{G,j+1}$ such that $t(W'_1) \equiv h(W_1)$ and for each $W_2 \in \mathcal{F}_{G,j+1}$ there exists $W'_2 \in \mathcal{F}_{G,j}$ such that $t(W'_2) \equiv h(W_2)$.*

Proof. Take $W'_1 \in \mathcal{F}_{G,j}$ and consider the simple feasible array W_1 satisfying (i) $t(W'_1) \equiv h(W_1)$ and (ii) the last column of W_1 consists entirely of 0's. Combining (i) and (ii) with the fact that $W'_1 \leq_a G[j-\omega+1..j]$ we conclude $W_1 \leq_a G[j-\omega+2..j+1]$ and thus $W_1 \in \mathcal{F}_{G,j+1}$. Similarly given $W_2 \in \mathcal{F}_{G,j+1}$ consider the feasible array W'_2 satisfying (i) $t(W'_2) \equiv h(W_2)$ and (ii) the first column consists entirely of 0's. Using similar reasoning to the first case we can conclude $W'_2 \in \mathcal{F}_{G,j}$. □

Theorem 2. *For $1 \leq j \leq n$, $\mathrm{MIS}(j)$ computed by Algorithm 1 gives the size of the maximum independent set in the graph $G[j]$ induced by the vertices of the first j columns in $\mathbb{Z}_{n,k}$.*

Proof. Recall that $\mathrm{MIS}(j) = \max_{W \in \mathcal{F}_{G,j}} \mathrm{MIS}(j, W)$ for all $1 \leq j \leq n$ and let $\mathrm{OPT}(j)$ denote the size of the maximum independent set in the graph $G[j]$ induced by the vertices of the first j columns of $\mathbb{Z}_{n,k}$ for all $1 \leq j \leq n$. We prove the theorem by induction on j. Firstly Lemma 1 proves that for each $W \in \mathcal{F}_{G,j}$ there exists W' such that $\max_{W' \in \mathcal{F}_{G,j-1}: t(W') \equiv h(W)} \mathrm{MIS}(j-1, W')$ is well defined. Next consider the case $j = 0$, since the first ω columns (indexed from $-(\omega-1), \ldots, 0$) consist entirely of 0's it implies that $\mathrm{MIS}(0) = 0$. In addition let $\vec{0} \in \mathcal{F}$ denote the feasible array consisting entirely 0's then $\mathcal{F}_{G,0} = \{\vec{0}\}$ and consequently $\mathrm{MIS}(0) = \mathrm{MIS}(0, \vec{0}) = 0$.

Base case: If $\mathrm{cs}(G[-(\omega-1)..1], 1) = 0$ then $\mathrm{OPT}(1) = 0$. $\mathcal{F}_{G,1} = \{\vec{0}\}$ and clearly $\mathrm{MIS}(1, \vec{0}) = 0$. Using the facts that (i) $\mathcal{F}_{G,0} = \{\vec{0}\}$, (ii) $\mathrm{MIS}(0, \vec{0}) = 0$, (iii) $\mathrm{cs}(\vec{0}, \omega) = 0$ and (iv) $t(\vec{0}) = h(\vec{0})$ it follows that $\mathrm{MIS}(1, \vec{0}) = \mathrm{MIS}(0, \vec{0}) + cs(\vec{0}, \omega)$. For all $W \notin \mathcal{F}_{G,1}$, $\mathrm{MIS}(1, W) = 0$ as it must be the case that $W \leq_a G[-(\omega-2)..1]$. This means that $\mathrm{MIS}(1) = 0$, which equals to $\mathrm{OPT}(1)$.

Otherwise $\mathrm{cs}(G[-(\omega-1)..1], 1) > 0$ meaning $\mathrm{OPT}(1) = 1$. In this case $\mathcal{F}_{G,1} \neq \{\vec{0}\}$. Thus there exists $W \in \mathcal{F}_{G,1}$ satisfying $t(\vec{0}) = h(W)$ and $W \neq \vec{0}$, and thus $\mathrm{cs}(W, \omega) = 1$. Since W it head-aligned with $\vec{0}$ and contains a 1 in its final column we conclude $\mathrm{MIS}(1, W) = 1$. Finally since $\mathcal{F}_{G,0} = \{\vec{0}\}$ and $\mathrm{MIS}(0, \vec{0}) = 0$ it follows that $\mathrm{MIS}(1, W) = \mathrm{MIS}(0, \vec{0}) + cs(W, \omega)$. Again for all $W \notin \mathcal{F}_{G,1}$, $\mathrm{MIS}(1, W) = 0$. Then $\mathrm{MIS}(1)$ equals to 1, which equals $\mathrm{OPT}(1)$.

Inductive step: Suppose that the result holds for all columns of G indexed from $1, \ldots, j-1$, i.e., $\mathrm{MIS}(i)$ equals to $\mathrm{OPT}(i)$ for all $1 \leq i < j$. We show that this implies $\mathrm{MIS}(j)$ equals to $\mathrm{OPT}(j)$. Assume on the contrary that there exists an independent array I in $G[-(\omega-1)..j]$ satisfying $\mathrm{as}(I) > \mathrm{MIS}(j)$ and suppose $W^* \in \mathcal{F}_{G,j}$ satisfies $W^* \equiv I[j-\omega+1..j]$. Then $\mathrm{as}(I) > \mathrm{MIS}(j) \geq \mathrm{MIS}(j, W^*)$ and thus

$$\mathrm{as}(I) > \max_{W' \in \mathcal{F}_{G,j-1}: t(W') \equiv h(W^*)} \mathrm{MIS}(j-1, W') + \mathrm{cs}(W^*, \omega). \tag{1}$$

Consider the independent array I' in $G[-(\omega-1)..j-1]$ obtained by removing the last column of I. Then it follows that $\mathrm{as}(I') = \mathrm{as}(I) - \mathrm{cs}(I, j)$. Furthermore consider the simple feasible array $W'' \in \mathcal{F}_{G,j-1}$ satisfying $I'[j-\omega..j-1] \equiv W''$,

then $t(W'') \equiv h(W^*)$. In addition $\text{cs}(I, j) = \text{cs}(W^*, \omega)$ since the last column of I is the same as the ωth column of W^*. Thus $\text{as}(I') = \text{as}(I) - \text{cs}(W^*, \omega)$, substituting this into Inequality (1) we obtain

$$\text{as}(I') > \max_{W' \in \mathcal{F}_{G,j-1} : t(W') \equiv h(W^*)} \text{MIS}(j-1, W').$$

Since $I'[j-2-\omega..j-1] \equiv W''$ and $W'' \in \mathcal{F}_{G,j-1}$ satisfying $t(W'') \equiv h(W^*)$ this is a contradiction to the optimality of $\max \text{MIS}(j-1, W')$ for $W' \in \mathcal{F}_{G,j-1}$ satisfying $t(W') \equiv h(W^*)$. □

3.3 Time Complexity

In this section we provide an upper bound on the worst case time complexity of the DP algorithm. We describe how the use of n separate bipartite graphs allow us to compute $\text{MIS}(j)$ for $1 \leq j \leq n$. Each bipartite graph consists of two sets of feasible arrays namely those which agree with $G[-(\omega-1)..(j-1)]$ and $G[-(\omega-1)..j]$ respectively. Edges between classes represent pairs of arrays which are tail-head aligned.

For an array G and $0 \leq j \leq n-1$ $B_j = (\mathcal{F}_{G,j}, \mathcal{F}_{G,j+1}, E)$ is a directed bipartite graph to model the tail-head alignment of arrays with classes $\mathcal{F}_{G,j}$ and $\mathcal{F}_{G,j+1}$. For a pair of simple feasible arrays $W \in \mathcal{F}_{G,j}$ and $W' \in \mathcal{F}_{G,j+1}$, $(W, W') \in E$ if and only if $t(W) \equiv h(W')$.

For a directed graph $G = (V, E)$ and vertex $v \in V$ let $N^-(v) = \{u \in V : (u, v) \in E\}$ and $d^-(u) = |N^-(u)|$. Similarly let $N^+(v) = \{u \in V : (v, u) \in E\}$ and $d^+(u) = |N^+(u)|$. Let $\Delta^+(G) = \max_{v \in V} d^+(v)$ denote the maximum out-degree of G. The bipartite graph B_j is used to compute $\text{MIS}(j+1, W)$ for each $W \in \mathcal{F}_{G,j+1}$ by considering $N^-(W)$ and selecting the array $W' \in N^-(W)$ for which $\text{MIS}(j, W')$ is maximised.

An important piece of information required for the time complexity is to obtain an upper bound on $|\mathcal{F}_{G,j}|$ for all $0 \leq j \leq n$. We do this by obtaining an upper bound on $|\mathcal{F}|$. We show that $|\mathcal{F}| = \Theta(\omega^k)$ through the following two observations. Firstly $|\mathcal{F}| \leq (\omega+1)^k$ since each simple feasible array contains at most one 1 per row, or not contain a 1 at all. Secondly $\frac{(\omega)!}{(\omega-k)!} \leq |\mathcal{F}|$ since there are precisely $\omega(\omega-1)(\omega-2)\cdots(\omega-(k-1))$ simple feasible arrays with exactly one 1 in each row. Thus $|\mathcal{F}_{G,j}| = O(\omega^k)$. We make use of the following lemma in the calculation of the worst case running time of the algorithm.

Lemma 3. *For each B_j, the maximum out degree $\Delta^+(B_j) \leq k+1$*

Proof. For each array in $W \in \mathcal{F}_{G,j}$ there are at most k arrays $W' \in \mathcal{F}_{G,j+1}$ with a single 1 in the final column satisfying $t(W) \equiv h(W')$ (one in each of the k possible locations). Combining this with the array consisting of entirely 0's in its final column gives us a $k+1$ possible feasible arrays. □

Theorem 4. *The worst case running time of the DP algorithm is $O(nk\omega^k)$.*

Proof. For each $W \in \mathcal{F}_{G,j+1}$ using Algorithm 1 we obtain $\text{MIS}(j+1, W)$ by comparing $\text{MIS}(j, W')$ for each $W' \in N^-(W)$. Using Lemma 3 and the fact that $|\mathcal{F}_{G,j}| = O(\omega^k)$ it can be seen that there at most $O(k\omega^k)$ computations per bipartite graph B_j. Finally given that there are n bipartite graphs B_j we obtain a worst case running time of $O(nk\omega^k)$. □

4 Extensions

In this section we extend our DP algorithm in Section 3 taking weight into account and show how it provides solutions to the following three problems: (i) The maximum weighted independent set problem for $k < \omega$ for constant k, (ii) The maximum weighted independent set problem for $k > \omega$ for constant k and ω, and (iii) A weighted version of the scheduling problem with parameter $1 \leq l \leq k$ for constant k.

Framework. W.l.o.g., we normalise the weight such that the minimum non-zero weight is 1, in other words, $G[i,j] = 0$ or $G[i,j] \geq 1$ for all $[i,j]$. Let $\mathcal{W}$ be the set of all $k \times \omega$ arrays with $0, 1$ entries. A basis set $\mathcal{F} \subseteq \mathcal{W}$ satisfies the followings (i) $\{\vec{0}\} \in \mathcal{F}$, (ii) If $W \in \mathcal{F}$ then $\exists W' \in \mathcal{F}$ where $t(W) \equiv h(W')$ and the last column of W' is all 0's and (iii) if $W \in \mathcal{F}$ then $\exists W'' \in \mathcal{F}$ where $t(W'') \equiv h(W)$ and the first column of W'' is all 0's. We extend some notations of Section 2 to account for the array generalisation. Given arrays G and I of the same size $I \leq_a G$ if (i) $G[i,j] = 0 \Rightarrow I[i,j] = 0$ and (ii) $1 \leq I[i,j] \leq G[i,j]$ for all $G[i,j] \neq 0$. Since I may contain values greater than 1 and W contains only $0, 1$ we introduce an additional notion of equivalence, denoted by $\equiv_a$. Given an array I of size $k \times \omega$, we say that $I \equiv_a W$ provided $I[i,j] = 0$ if and only if $W[i,j] = 0$.

We extend G to an array of size $k \times (\omega + n)$ with columns indexed from $-(\omega-1), \ldots, n$ and the first ω columns consisting of entirely 0's. Given a basis set $\mathcal{F}$ let $\mathcal{F}_{G,j}$ denote the set of feasible arrays $W \in \mathcal{F}$ satisfying $W \leq_a G[j-\omega+1..j]$. The goal is to find an array I of maximum array sum satisfying $I \leq_a G$ and $I[j-\omega+1..j] \equiv_a W$ for some $W \in \mathcal{F}_{G,j}$ for all $1 \leq j \leq n$. It is important to note that $\vec{0} \in \mathcal{F}_{G,j}$ for all $1 \leq j \leq n$ and so the array of size $k \times (\omega + n)$ consisting entirely of 0's satisfies the required conditions proving such an array always exists. Let $\text{OPT}(j)$ denote the array sum of the largest array satisfying the conditions for $G[-(\omega-1)..j]$ for all $1 \leq j \leq n$. We are required to compute $\text{OPT}(n)$.

Algorithm 1 can be extended by modifying the main recurrence as follows: $F(0, \vec{0}) = 0$ and for each $W \in \mathcal{F}$ and $1 \leq j \leq n$ let

$$F(j, W) = \begin{cases} \max\limits_{W' \in \mathcal{F}_{j-1} : t(W') \equiv h(W)} F(j-1, W') + W[\omega]^T \cdot G[j] & \text{if } W \in \mathcal{F}_{G,j}\ , \\ 0 & \text{otherwise.} \end{cases}$$

Note $W[\omega]^T \cdot G[j]$ denotes the dot product of the ωth column of W and the jth column of G. Let $F(j) = \max_{W \in \mathcal{F}_{G,j}} F(j, W)$, we use the following theorem to calculate $\text{OPT}(n)$.

In the full paper we prove that the recurrence and the associated dynamic programming algorithm gives an optimal value.

Theorem 5. *$F(j)$ is equal to $\mathrm{OPT}(j)$ for all $1 \leq j \leq n$.*

We analyse the worst case running time of the algorithm in a similar way to the case for the maximum independent set problem let $B_j = (\mathcal{F}_{G,j}, \mathcal{F}_{G,j+1}, E)$ for $0 \leq j \leq n-1$. Let $\Delta^+(B_j)$ denote the largest out-degree of an array in B_j for $0 \leq j \leq n-1$ and let $\Delta^+(\mathcal{F}) = \max_j(\Delta^+(B_j))$. We then prove the following lemma regarding the running time.

Lemma 6. *The worst case running time of the generalised DP is $O(n\Delta^+(\mathcal{F})|\mathcal{F}|)$.*

Proof. Given B_j the number of computations required to compute $F(j+1, W)$ for $W \in \mathcal{F}_{j+1}$ is proportional to the in-degree $d^-(W)$. Thus the total number of computations required for B_j is proportional to the number of edges which is at most $\Delta^+(\mathcal{F})|\mathcal{F}|$, since there are n bipartite graphs, the result follows. □

Applications of the extension. We now show how we solve the three problems introduced at the start of the section by choosing the basis set $\mathcal{F}$ corresponding to the problem.

Weighted independent set. We consider the maximum weighted independent set problem in LoS networks with $k < \omega$, where each weight assigned to a vertex has a value least 1. $G[i, j]$ is the weight assigned to the vertex in location $[i, j]$ of the LoS embedding of G. Since this is just the weighted version our initial problem we keep the same basis $\mathcal{F}$ which consists of all arrays with at most one 1 in each column and row.

Weighted independent set for larger k. We consider the maximum weighted independent set problem in LoS networks with parameter $k > \omega$ where $k \in \mathbb{N}$ is a constant. In this case an independent set $\mathcal{I}$ can have more than one 1 in a column. In particular $\mathcal{I}$ satisfies that (i) for distinct columns j_1, j_2 if $I[i, j_1] = 1$ and $I[i, j_2] = 1$ then $|j_1 - j_2| \geq \omega$ and (ii) for distinct rows i_1, i_2 if $I[i_1, j] = 1$ and $I[i_2, j] = 1$ then $|i_1 - i_2| \geq \omega$. Thus the basis set $\mathcal{F}$ represents the set of $k \times \omega$ arrays having at most one 1 per row and for each column the distance between any pair of 1's needs to be at least ω.

Weighted scheduling problem. We consider the scheduling problem where the parameter $1 \leq l \leq k$ with the addition that prices charged have different weights that are at least the value 1. Thus $G[i, j]$ contains the price charged to client i on day j. In this problem the basis set $\mathcal{F}$ is the set of all $k \times \omega$ arrays which consist of at most one 1 in each row and at most l, 1's in each column.

We use Lemma 6 to calculate the worst-case running times are $O(nk\omega^k)$ for the first problem, $O(nt^t(\omega)^{(t+1)\omega})$ for the second where $t = \lceil \frac{k}{\omega} \rceil$, and $O(nk^l\omega^k)$ for the third. Full details are provided in the full paper.

5 EPTAS

The DP algorithms in Sections 3 and 4 give optimal solutions to an offline version of the MIS problem in LoS networks and scheduling problem where the

entire input is known in advance. This is unrealistic for example the duration in the scheduling problem is large possibly spanning a year or more then it is likely the input evolves over time. It is desirable to take a more online approach with a good approximation performance. We improve the running time of the EPTAS in [12] based on the DP algorithms, providing a solution which is semi-online; in particular we assume we are allowed to observe the input up to a certain "look-ahead" distance. We show how the look-ahead distance influences the approximation ratio achieved.

Given a LoS network G, let G_j and $\overline{G_j} = G \backslash G_j$ denote the induced subgraph of G consisting of vertices which are embedded in the region with x-coordinates from 1 to $j\omega$ and from $j\omega+1$ to n, respectively. Let I_r be a maximum independent set in G_r. We determine a value r^* which is the "stopping point" of the overhead distance necessary to achieve $(1+\epsilon)$-approximation. Precisely, we let r^* be the smallest integer such that $|I_{r^*+1}| < (1+\epsilon)|I_{r^*}|$. This means that for any $1 < r \leq r^*$, $|I_r| \geq (1+\epsilon)|I_{r-1}|$ (note that $|I_r| \leq kr$ due to the structural properties of a LoS network embedding). We first show an upper bound on r^* (proof is deferred to the full paper).

Lemma 7. $r^* \leq \left(\frac{1+\sqrt{1+4\ln(1+\epsilon)\ln(k)}}{2\ln(1+\epsilon)} \right)^2$

To obtain a $(1+\epsilon)$-approximation to the maximum independent set once r^* is obtained we remove G_{r^*+1} from the graph G and apply the procedure iteratively to the graph $\overline{G_{r^*+1}}$. If I' is the independent set obtained from applying the procedure to $\overline{G_{r^*+1}}$ then we show that $I_{r^*} \cup I'$ is a $(1+\epsilon)$-approximation to the maximum independent set in G.

Lemma 8. *Suppose that I' is a $(1+\epsilon)$-approximate independent set in $\overline{G_{r^*+1}}$, then $I \equiv I_{r^*} \cup I'$ is $(1+\epsilon)$-approximate for G.*

Proof. Recall that I_{r^*+1} is the largest independent set in G_{r^*+1}, since $|I_{r^*+1}| < (1+\epsilon)|I_{r^*}|$ it follows that I_{r^*} is a $(1+\epsilon)$-approximate independent set on G_{r^*+1}. Using the properties of LoS networks, for any vertex $v \in I_{r^*}$ the distance between v and a neighbour $u \in N(v)$ is at most ω. Thus the neighbourhood $\cup_{v \in I_{r^*}} N(v)$ belongs to G_{r^*+1} and we can conclude that $I_{r^*} \cup I'$ is an independent set in G. We denote by α the independence number. Finally, $\alpha(G) \leq \alpha(G_{r^*+1}) + \alpha(\overline{G_{r^*+1}})) \leq (1+\epsilon)|I_{r^*}| + (1+\epsilon)|I'| = (1+\epsilon)|I_{r^*} \cup I'|$, giving us the required result. □

Suppose we define one iteration of the algorithm as the process of reaching the first stopping point and second iteration as the next process of reaching the second stopping point and so on. Given $r^* = O(\omega)$ using Theorem 4 we can deduce computing I_r using the DP algorithm has a worst case running time of $O(\omega k \omega^k)$. Since we repeat this calculation r^* times in each iteration, the worst case running time of an iteration is $O(k\omega^{k+2})$. Finally since there are at most n iterations the EPTAS has a worst case running time of $O(nk\omega^{k+2})$.

We now turn our attention to the uses of the EPTAS for the scheduling application in the case where k is constant, $k < \omega$ and $l = 1$, note under these

restrictions the goal of the scheduling problem is equivalent to the MIS problem. Suppose the advertisement company does not have a full schedule but would like to start processing a schedule given some partial information. The EPTAS shows a look-ahead distance of $c_1\omega$ where $c_1 = \left(\frac{1+\sqrt{1+4\ln(1+\epsilon)\ln(k)}}{2\ln(1+\epsilon)}\right)^2$ is sufficient. Once the we have computed the first stopping point r^* we can process the schedule up to $r^*(\omega+1)$ with a $(1+\epsilon)$-approximation guarantee. We then repeat this process when the next stopping point is computed and so on. A portion of the schedule of length $c_1\omega$ is sufficient to guarantee a stopping point is found. Hence we say that the EPTAS uses a $c_1\omega$ look-ahead distance.

Theorem 9. *For any $\epsilon > 0$, we have an EPTAS of time complexity $O(nk\omega^{k+2})$ provided we have a look-ahead of $c_1\omega$, where $c_1 = \left(\frac{1+\sqrt{1+4\ln(1+\epsilon)\ln(k)}}{2\ln(1+\epsilon)}\right)^2$.*

Extensions. Similar to Section 4, we show that with some small modifications our EPTAS can be used for the maximum weighted independent set problem. We assume however that it is known a priori that there exists some global parameter $w_{\max} > 1$ which is constant such that for each vertex v in our graph $W(v) \leq w_{\max}$.

I_r is defined as the largest weighted independent set in G_r. We define the weight of I_r as $W(I_r) = \sum_{v \in I_r} W(v)$. Then r^* is defined as the smallest integer such that $W(I_{r^*+1}) < (1+\epsilon)W(I_{r^*})$. I.e., for any $1 < r \leq r^*$, $W(I_r) \geq (1+\epsilon)W(I_{r-1})$ and $W(I_r) \leq krw_{\max}$. The proof of the following lemma follows from Lemma 7 by setting $k' = kw_{\max}$.

Lemma 10. $r^* \leq \left(\frac{1+\sqrt{1+4\ln(1+\epsilon)\ln(k')}}{2\ln(1+\epsilon)}\right)^2$

The proof of the following theorem is left for the full paper.

Theorem 11. *Suppose that I' is a $(1+\epsilon)$-approximate weighted independent set in $\overline{G_{r^*+1}}$, then $I \equiv I_{r^*} \cup I'$ is $(1+\epsilon)$-approximate for G.*

6 Conclusions

In this paper we study the WMIS problem on restricted LoS networks where parameter k is a constant, and propose a polynomial time dynamic programming algorithm for the problem. We also use the DP algorithm to develop an EPTAS that applies to a semi-online algorithm with performance ratio $(1+\epsilon)$ and a look-ahead distance dependent on ϵ, for any $\epsilon > 0$.

For future work there are various avenues to explore. One immediate direction is to study the LoS network with different ranges of various parameters. It is interesting to determine the complexity of the problem (polynomial time solvable or NP-hard) given different values of the parameters. We can also extend the problem definition such that the distance restriction ω may take two different

values ω_1 and ω_2 for each of the two dimensions. Another direction is to study other optmisation problems on LoS networks. Furthermore, we can explore other scheduling problems with constraints that can modeled in a similar way as a LoS network and adapt solutions to these scheduling problems.

References

1. Susanne Albers. Online algorithms: a survey. *Mathematical Programming*, 97(1-2):3–26, 2003.
2. Richard Bellman, Augustine O Esogbue, and Ichiro Nabeshima. *Mathematical Aspects of Scheduling and Applications*. Elsevier, 2014.
3. Béla Bollobás, Svante Janson, and Oliver Riordan. Line-of-sight percolation. *Combinatorics, Probability and Computing*, 18(1-2):83–106, 2009.
4. Marco Cesati and Luca Trevisan. On the efficiency of polynomial time approximation schemes. *Information Processing Letters*, 64(4):165–171, 1997.
5. Sung Nok Chiu, Dietrich Stoyan, Wilfrid S Kendall, and Joseph Mecke. *Stochastic geometry and its applications*. John Wiley & Sons, 2013.
6. Artur Czumaj and Xin Wang. Communication problems in random line-of-sight ad-hoc radio networks. In *International Symposium on Stochastic Algorithms*, pages 70–81. Springer, 2007.
7. Reinhard Diestel. *Graph theory*. Springer, 2000.
8. Linda Farczadi and Luc Devroye. Connectivity for line-of-sight networks in higher dimensions. *Discrete Mathematics & Theoretical Computer Science*, 15, 2013.
9. Alan Frieze, Jon Kleinberg, R Ravi, and Warren Debany. Line-of-sight networks. *Combinatorics, Probability and Computing*, 18(1-2):145–163, 2009.
10. Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of NP-completeness, 1979.
11. Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(38), 1997.
12. Pavan Sangha and Michele Zito. Finding large independent sets in line of sight networks. In *Algorithms and Discrete Applied Mathematics - Third International Conference, CALDAM 2017, Sancoale, Goa, India, February 16-18, 2017, Proceedings*, pages 332–343, 2017.