Interference-Aware Broadcast Scheduling in Wireless Networks

Gruia Calinescu\textsuperscript{1},* Sutep Tongngam\textsuperscript{2}

Department of Computer Science, Illinois Institute of Technology, 10 W. 31st St., Chicago, IL 60616, U.S.A.

Abstract

In this paper, we study the \textit{Interference-Aware Broadcast Scheduling} problem, where all nodes in the Euclidean plane have a transmission range and an interference range equal to $r$ and $\alpha r$ for $\alpha \geq 1$, respectively. Minimizing latency is known to be NP-Hard even when $\alpha = 1$. The network radius $D$, the maximum graph distance from the source to any node, is also known to be a lower bound.

We formulate the problem as Integer Programs (IP) and optimally solve moderate-size instances. We also propose six variations of heuristics, which require no pre-processing of inputs, based on the number of receivers gained by each additional simultaneous transmitting node. The experimental results show that the best heuristics give solutions that exceed the optimal solutions by only 13-20\%, the optimum solutions. Further, an $O(\alpha D)$ schedule is proven to exist yielding an $O(\alpha)$ approximation algorithm.

Keywords: latency minimization, broadcast, integer linear program, heuristics, approximation algorithm

1. Introduction

Broadcast is a fundamental operation in wireless networks, the objective of which is to send a message from a node, called \textit{source}, to all other nodes in the network. Tseng et al. point out in (1) many applications in ad-hoc networks, e.g., finding a route to a particular host, paging a host, and sending an alarm signal, where the broadcast operation is used. They also consider the characteristics of wireless equipment, mobile equipment in particular, such as power limitation, channel utilization, and energy efficient requirement, which make a single hop communication performed by a long transmission range node undesirable. For these cases, to distribute messages over the network requires multi-hop communication, or data forwarding.

In some situations, such as an emergency, disaster, storm forecasting, or other urgencies, a message should be broadcast to all nodes as soon as possible, or within minimum timeslots.

\textsuperscript{*} Corresponding author
\textsuperscript{1} Email addresses: calinescu@iit.edu (Gruia Calinescu), tongsut@iit.edu (Sutep Tongngam)
\textsuperscript{2} Research supported in part by NSF grants CCF-0515088 and NeTS-0916743
\textsuperscript{1} Research supported in part by NSF grant CCF-0515088 and by the Royal Thai Government. Current affiliation: Department of Computer Science, National Institute of Development Administration, Thailand

Preprint submitted to Ad Hoc Networks December 28, 2010
The term timeslot is used to represent a period of time for nodes to transmit. In each timeslot, nodes which previously received the message can send that message (transmit). If a single node transmits, all the nodes within its transmission range receive the message. Multiple transmitting nodes may cause interference and only some of the nodes within their transmission range receive the message. A schedule dictates which nodes transmit in which timeslots. The latency of a schedule is the number of timeslots used until all nodes in the network completely receive the message. For a given network, the Broadcast Scheduling problem asks to find a schedule of minimum latency. Such a schedule gives the latency of the network.

Without interference and collision in consideration, nodes that already received the message are allowed to transmit that message in the next timeslot. By creating a graph representing nodes by vertices and communications between nodes by edges (we call it the communication graph), one can optimally solve the problem by applying breadth first search (BFS) starting from the source. The depth of the resulting tree will be the latency of the network; the schedule is that nodes at graph distance $j - 1$ from the source transmit the message in timeslot $j$.

In the situation where interference and collision are of concern, nodes will not receive the message if an interference and/or a collision occurs at them. A collision occurs at node $v$ if two or more nodes within transmission range of $v$ transmit at the same time. Likewise, an interference occurs at $v$ when $v$ is receiving the message from one node and some other node within the interference range of $v$ is transmitting the message simultaneously. Note that the interference is defined as a ratio $\alpha$ to the transmission range; when $\alpha = 1$ the definitions of interference and collision coincide. This is a simplified model which, as we see below, was used in literature. More realistic models appear in, for example, Moscibroda et al. (2). See also (3) and (4) for scheduling in those models.

When the interference range is assumed to be equal to the transmission range, the interference ratio $\alpha$ is equal to 1. However, as described in (5), the interference range can be different from, in fact larger than, the transmission range, i.e., $\alpha > 1$. Therefore Collision-Free Broadcast Scheduling algorithms found in, for instance, (6), (7), (8), (9), or (10), where transmission range and interference range are assumed to be the same, are not sufficient for the case $\alpha > 1$.

### 1.1. Previous work

The Euclidean model is used when identical nodes in the network can be represented as points in the Euclidean plane and the distance between two nodes is denoted by the Euclidean distance. We normalize the distances such that the transmission range is 1. The communication graph of an instance is a special type of graph called unit disk graph (UDG). By defining the collision and interference as at least two neighbors of node $v$ transmit packets at the same time, i.e., $\alpha = 1$, Gandhi et al. prove in (10) that finding a minimum-latency broadcast schedule with the collision constraint is NP-Hard in the Euclidean model. They also propose a distributed Collision-Free Broadcast Scheduling algorithm with latency $O(D)$, where $D$ is the radius of the communication graph. The graph radius $D$ is defined as the maximum graph distance from the source of the broadcast. Thus $D$ is the depth of the BFS tree rooted at the source.

S.C.-H. Huang et al. propose three progressively improved approximation algorithms for the same problem, also in the Euclidean model. Their centralized algorithms are based on connected dominating set, $k$-independent set, and node coloring of the input graph. They claim in (11) that their algorithms produce broadcast schedules with latency at most $24D - 23$, $16D - 15$, and $D + O(\log D)$.

In a distributed setting, Emek et al. (12) obtain matching upper and lower bounds of $\Theta(\min(D+g^2,D\log g))$, where $g$, called the granularity of the network, is the inverse of the minimum dis-
tance between any two nodes. These bounds hold if all nodes are awake (and may transmit messages) from the beginning, and the upper bounds are obtained by deterministic algorithms. They also present algorithms and lower bounds for the case where the nodes other than the source are initially idle and cannot transmit until they hear a message for the first time. Further discussion of distributed algorithms appears in (13).

Recently, Z. Chen et al. (5) propose a centralized algorithm to approximately solve the Interference-Aware Broadcast Scheduling problem for $\alpha > 1$. Their algorithm is based on breadth first search tree construction. They claim that their algorithm achieves a constant 26 approximation ratio when the interference range is twice the transmission range, i.e., $\alpha = 2$. They further claim an $O(\alpha^2)$ approximation ratio.

While this work was in preparation, Mahjourian et al. (14) published an $O(\alpha^2)$-approximation algorithm and a greedy heuristic. They analyze experimentally their two algorithms, comparing their results to the depth of the BFS tree, and to the results of the Chen et al. (5) algorithm.

1.2. New results

In this paper, we consider the Interference-Aware Broadcast Scheduling problem in the Euclidean model. Our contributions include the Integer Programming (IP) formulations to optimally solve moderate-size instances of the problem.

Six greedy heuristics are also presented in this paper, and they differ from the one in (14). Unlike those proposed in (5) and (15), where instances need pre-processing, i.e., computing a BFS-tree or a constant density spanner, respectively, our heuristics do not need any pre-processing of instances. As of our knowledge, we are the first to compare the results of greedy heuristics to the optimum solutions obtained from Integer Programming.

According to our theoretical result, an $O(\alpha D)$ schedule can be computed by a centralized algorithm in $\mathcal{O}(n^2)$ time; therefore an $O(\alpha)$ approximation algorithm is obtained. Here else-where in the paper $n$ is the number of nodes in the given network/graph.

1.3. Related Work

Broadcasting has also been studied in the graph model, where the communication graph is an arbitrary undirected graph. It is easy to see that the graph radius $D$ with respect to the source serves as a lower bound for the latency of any broadcast schedule (6). When collisions are of concern, a complete message reception is defined as follows: a node has received a complete message if and only if exactly one of its neighbors transmits that message at the time of receiving (this is akin to $\alpha = 1$). Most of the papers prove bounds in terms of $D$.

Alon et al. (6) give a family of undirected radius-2 graphs with latency $\Omega(\log^2 n)$. Chlamtac and Weinstein present in (7) a centralized algorithm giving a bound of $O(D \ln^2(n/D))$ on the required timeslots, even in a directed graph. Gaber and Mansour later show in (16) the existence of a broadcast schedule with latency $O(D + \log^5 n)$ for any graph, and give a polynomial time centralized algorithm to output such a schedule.

For distributed protocols, in general, neither global knowledge of node location and identity nor synchronization is assumed to be prior known for all nodes. Each node only knows its identity and its neighbors. Lower bounds on the number of rounds required for any deterministic and randomized, distributed broadcasting protocol have been obtained by Bruschi and Pinto (8), Kushilevitz and Mansour (9), and Kowalski and Pelc (17). Protocols with number of rounds close to the lower bounds appear in (17), Chrobak et al. (18), and Chlebus (19).

M. Onus et al. (15) use the following model: the communication graph is a UDG or a more general version of UDG, and the interference range given by $\alpha > 1$ means that the transmission
Figure 1: An UDG with nodes \(x\) and \(y\) at UDG distance \(n - 2\) and Euclidean distance at most 2. In our model with \(\alpha = 2\), the transmission from \(x\) to \(u\) causes interference at \(y\), who cannot receive in the same timeslot a packet from \(v\).

of a node \(x\) interferes with the receiving by \(y\) of the message sent by \(z\) (with \(z \neq x\)) if the graph distance from \(x\) to \(y\) is at most \(\alpha\). It should be noted that this is not the same as the Euclidean model, as one can have two nodes at Euclidean distance 2 and UDG distance \(n - 2\). See Figure 1 for an illustration. The main result of (15) is a distributed broadcasting protocol requiring, with high probability, \(O(D + \log n)\) rounds/timeslots to deliver the message from the source to all nodes. Their protocol is based on a given constant density spanner (see details in (20)). When \(\alpha\) is not considered as a constant, however, their protocol and proof give a schedule with latency \(O(\alpha^2 D + \log n)\).

2. Preliminaries

Our work is based on the Euclidean model assuming all nodes have the transmission range equal \(r\) and interference range be \(\alpha r\) for \(\alpha \geq 1\). When normalizing \(r\) to 1, the interference range is equal to \(\alpha\). We use \(|v, x|\) to denote the Euclidean distance between nodes \(v\) and \(x\).

We informally define the two problems we study, with precise definitions coming towards the end of this section. The **Interference-Aware Broadcast Scheduling** problem is to find the minimum number of timeslots until all nodes receive the message, in the model where a node \(v\) receives a packet if there is no interference at \(v\) while another node transmits to \(v\). The **Interference-Free Broadcast Scheduling** problem, on the other hand, is to find the minimum number of timeslots until all nodes receive the message, in the model where no interference is allowed at nodes that did not receive the message yet. This constraint prevents nodes from receiving incorrect messages at all.

Next we give the definitions of collision and interference. Then we define message reception followed by the definitions of correct schedules for both problems.

**Definition 1.** A collision occurs at a node \(v\) in timeslot \(j\) if \(v\) did not receive the message before timeslot \(j\) and two distinct nodes within \(v\)'s transmission range are transmitting in timeslot \(j\).

In Figure 2, for example, a collision will occur at node \(B\) if nodes \(X\) and \(Y\) transmit at the same time. In Figure 3, however, no collision occurs at \(B\) even when both \(X\) and \(Y\) transmit at the same time since \(B\) already received the message.
Definition 2. An interference occurs at a node $v_r$ in timeslot $j$ if $v_r$ did not receive the message before timeslot $j$ and two distinct nodes, one within $v_r$’s transmission range and the other within $v_r$’s interference range, are transmitting in timeslot $j$.

In Figure 3, given $\alpha = 2$, an interference will occur at node $A$ if nodes $X$ and $B$ transmit at the same time.

Definition 3. A node $v_r$ has received the message in timeslot $j$ if some other node $v_t$ within a transmission range of $v_r$ is transmitting the message and no collision/interference occurs at $v_r$ in timeslot $j$.

Definition 4. Given set of nodes $V$ and $R_0 = \{\text{source}\}$, a correct interference-aware broadcast schedule $S$ with latency $T$ is a collection of sets of transmitting nodes in each timeslot $i \in \{1, 2, ..., T\}$, called $B_i$, and of sets of receiving nodes in each timeslot $i \in \{1, 2, ..., T\}$, called $R_i$, such that

1. $\forall i \in \{1, 2, ..., T\} \ B_i \subseteq \bigcup_{j=0}^{i-1} R_j$

A node can transmit in a timeslot if and only if it has received the message in some previous timeslot.
2. \( \forall i \in \{1,2,...,T\} \forall v_r \in R_i \exists v_t \in B_i(\|v_r, v_t\| \leq 1 \land \forall v'((v_r \in B_i \setminus v_t) \to (\|v_r, v'_t\| > \alpha))) \)

For any timeslot, a node \( v_r \) receives the message if there exists only one node \( v_t \) within \( v_r \)'s transmission range that is transmitting the message, and no other node within \( v_r \)'s interference range is transmitting.

3. \( \bigcup_{j=0}^{T} R_i = V \)

All nodes must receive the message within \( T \) timeslots.

4. \( \forall i \in \{1,2,...,T\} \forall j \in \{1,2,...,T\} ((j \neq i) \to (R_i \cap R_j = \phi)) \)

The reception of each node will be counted at most once (the first time), therefore no node appears in more than one set of \( R \).

In the scenarios from Figure 2, given \( \alpha = 2 \), in an interference-aware schedule it is allowed for both nodes \( X \) and \( Y \) to transmit at the same time: nodes \( A \) and \( C \) receive the message, while node \( B \) does not.

**Definition 5.** Given \( n \) nodes in the Euclidean plane, a source node, and an interference range \( \alpha \) with \( \alpha \geq 1 \), the INTERERENCE-AWARE BROADCAST SCHEDULING (IABS) problem is finding a correct interference-aware broadcast schedule with minimum latency.

**Definition 6.** Given set of nodes \( V \) and \( R_0 = \{\text{source}\} \), a correct interference-free broadcast schedule \( S_{cf} \) with \( T \) timeslots is a collection of sets of transmitting nodes in each timeslot \( i \in \{1,2,...,T\} \), called \( B_i \), and sets of receiving nodes in each timeslot \( i \in \{1,2,...,T\} \), called \( R_i \), such that

1. \( \forall i \in \{1,2,...,T\} \forall v_r \in R_i \exists v_t \in B_i(\|v_r, v_t\| \leq 1 \land \forall v'((v_r \in B_i \setminus v_t) \to (\|v_r, v'_t\| > \alpha))) \)

A node can transmit in a timeslot if and only if it has received the message in some previous timeslot.

2. \( \forall i \in \{1,2,...,T\} \forall v_r \in R_i \exists v_t \in B_i(\|v_r, v_t\| \leq 1 \land \forall v'((v_r \in B_i \setminus v_t) \to (\|v_r, v'_t\| > \alpha))) \)

For any timeslot, a node \( v_r \) receives the message if there exists only one node \( v_t \) within \( v_r \)'s transmission range that is transmitting the message, and no other node within \( v_r \)'s interference range is transmitting.

3. \( \bigcup_{j=0}^{T} R_i = V \)

All nodes must receive the message within \( T \) timeslots.

4. \( \forall i \in \{1,2,...,T\} \forall j \in \{1,2,...,T\} ((j \neq i) \to (R_i \cap R_j = \phi)) \)

The reception of each node will be counted at most once (the first time), therefore no node appears in more than one set of \( R \).

5. \( \forall i \in \{1,2,...,T\} \forall v \in B_i \forall v' \in V ((\|v, v'\| \leq 1) \to v' \in \bigcup_{j=0}^{T} R_j) \)

In interference-free broadcast, no collision/interference occurs at any node which did not already receive the message. Above there is a logically equivalent formulation, with the non-trivial equivalence explained next. Indeed, the condition says that if \( v \) transmits at time \( i \) and \( v' \) is within the transmission range of \( v \), then \( v' \) is going to receive the message at time \( i \), unless it has received it before. Then indeed no interference can occur at some \( u \) during timeslot \( i \): an interference occurring at \( u \) means that \( u \) has not received the message previously, and there is a node \( x \) that transmits to \( u \), and another node \( y \) whose transmission interferes with \( u \) receiving the message from \( x \). But in this case condition 2 above ensures that \( u \in R_y \) and therefore \( u \notin \bigcup_{j=0}^{T} R_j \), and the condition 5 does not hold for \( i, v = x \) and
\(v' = u\). For the reverse, if condition 5 does not hold, then there exist a node \(v\) transmitting at time \(i\) and a node \(v'\) within the transmission range of \(v\) with \(v'\) not receiving the message during the first \(i\) timeslots (recall that each node \(v\) is counted as receiving the message only the first time it receives it). This can happen only if an interference occurs at \(v'\) in timeslot \(i\).

In the scenarios from Figure 3, given \(\alpha = 2\), in an interference-free schedule it is allowed for both nodes \(X\) and \(Y\) to transmit at the same time: nodes \(A\) and \(C\) receive the message, while node \(B\) already received it. However, in the scenario from Figure 2, given \(\alpha = 2\), in an interference-free schedule it is not allowed for both nodes \(X\) and \(Y\) to transmit at the same time: a collision/interference happens at node \(B\), who has not received the message.

**Definition 7.** Given \(n\) nodes in the Euclidean plane, a source node, and an interference range \(\alpha\) with \(\alpha \geq 1\), the **Interference-Free Broadcast Scheduling (IFBS)** problem is finding a correct interference-free broadcast schedule with minimum latency.

### 3. Integer Programs for Broadcast Scheduling problems

Integer and linear programs have been proposed before for related problems. Björklund et al. (21) do so for TDMA scheduling - where all interference is disallowed. Our approach also has 0-1 variables to represent which nodes transmit in a given timeslot, and differs as we also use variables to represent which nodes are receivers in a given timeslot.

#### 3.1. Interference-Aware Broadcast Scheduling

Given \(T = \) number of timeslots, source \(v_0\), set of nodes \(V = \{v_1, v_2, \ldots, v_N\}\) where all nodes have a path to \(v_0\), interference ratio \(\alpha\), and \(\Delta_i\) number of nodes within interference range of \(v_i\), the objective is to find the maximum number of nodes that have received the message under the defined constraints. If that number is equal to the number of nodes, \(N\), the IABS instance has a feasible solution with latency \(T\). We seek the minimum possible latency, so the program below must be solved for \(T = 1, 2, \ldots\) until the objective equals \(N\). Note that \(T\) cannot exceed \(N\), as a solution with \(N\) timeslots always exists: while running BFS, in timeslot \(j\), for \(j = 1, 2, \ldots, N\), the \(j^{th}\) node extracted from the queue transmits.

The program has 0-1 variables \(x_{i,j}\), for \(0 \leq i \leq N\) and \(1 \leq j \leq T\), and \(y_{i,j}\), for \(0 \leq i \leq N\) and \(0 \leq j \leq T\). The idea is to have \(x_{i,j} = 1\) represent that a node \(v_i\) transmits the message at timeslot \(j\), and \(y_{i,j} = 1\) represent that node \(v_i\) receives the message for the first time at timeslot \(j\). We use the notation \(i' \sim i\) if \(|v_{i'}v_i| \leq 1\), and \(i' \leftrightarrow i\) if \(|v_{i'}v_i| \leq \alpha\). The integer program corresponding to the IABS instance is the following:

\[
\text{Maximize } \sum_{i=1, j=1}^{N,T} y_{i,j}
\]

Subject to

\[
\sum_{j=0}^{T-1} y_{i,j} - x_{i,j} \geq 0; \quad \forall i \in \{0, 1, \ldots, N\}, \forall t \in \{1, 2, \ldots, T\} \tag{1}
\]
The corresponding to the IFBS instance is the following:

\[ \sum_{j=0}^{T} y_{i,j} \leq 1; \quad \forall i \in [0, 1, \ldots, N] \tag{2} \]

\[ y_{i,j} \leq \sum_{r=i}^{T} x_{r,j}; \quad \forall i \in [1, 2, \ldots, N], \forall j \in [1, 2, \ldots, T] \tag{3} \]

\[ \sum_{r=i}^{T} x_{r,j} + \Delta_{i} y_{i,j} \leq \Delta_{i} + 1; \quad \forall i \in [1, 2, \ldots, N], \forall j \in [1, 2, \ldots, T] \tag{4} \]

\[ y_{i,j} \text{ and } x_{i,j} \in \{0, 1\} \quad \forall i \in [0, 1, \ldots, N], \forall j \in [1, 2, \ldots, T] \tag{5} \]

\[ y_{0,0} = 1 \tag{6} \]

\[ y_{i,0} = 0; \quad \forall i \in \{1, 2, \ldots, N\} \tag{7} \]

**Claim 8.** The IP (3.1) has a feasible solution, within \( T \) timeslots, with objective equal to the number of nodes, \( N \), if and only if there exists a correct interference-aware broadcast schedule with latency \( T \).

**Proof.** The first condition of Definition 4 is equivalent to Constraint 1 above. The third condition of Definition 4 is equivalent to the objective function of the IP being \( N \). The fourth condition of Definition 4 implies Constraint 2 above, and is implied by Constraint 2 if the objective function of the IP is \( N \). The second condition of Definition 4 is equivalent to constraints 3 and 4 above. Indeed, Constraint 3 says that if node \( i \) is to receive the message in timeslot \( j \) \((y_{i,j} = 1)\), one node which has \( i \) in its transmission range must transmit \( \sum_{r=i}^{T} x_{r,j} \geq 1 \). Also, no other node should interfere: Constraint 4 makes \( \sum_{r=i}^{T} x_{r,j} \leq 1 \) whenever \( y_{i,j} = 1 \). \( \square \)

### 3.2. Interference-Free Broadcast Scheduling

Given \( T \) number of timeslots, source \( v_0 \), set of nodes \( V = \{v_1, v_2, \ldots, v_N\} \) where all nodes have a path to \( v_0 \), interference ratio \( \alpha \), and \( \Delta_i \) number of nodes within interference range of \( v_i \), the objective is to find the maximum number of nodes that have received the message under the defined constraints. If that number is equal to the number of nodes, \( N \), the IFBS instance has a feasible solution with latency \( T \). We seek the minimum possible latency, so the program below must be solved for \( T = 1, 2, \ldots \) until the objective equals \( N \). Note that \( T \) cannot exceed \( N \), as a solution with \( N \) timeslots always exists: while running BFS, in timeslot \( j \), for \( j = 1, 2, \ldots, N \), the \( j^{th} \) node extracted from the queue transmits.

The program has \( 0 \sim T \) variables \( x_{i,j} \), for \( 0 \leq i \leq N \) and \( 1 \leq j \leq T \), and \( y_{i,j} \), for \( 0 \leq i \leq N \) and \( 0 \leq j \leq T \). The idea is to have \( x_{i,j} = 1 \) represent that a node \( v_i \) transmits the message at timeslot \( j \), and \( y_{i,j} = 1 \) represent that node \( v_j \) receives the message for the first time at timeslot \( j \). Again, we use the notation \( i' \sim i \) if \( |v_i v_{i'}| \leq 1 \), and \( i' \Leftrightarrow i \) if \( |v_i v_{i'}| \leq \alpha \). The integer program corresponding to the IFBS instance is the following:

\[
\begin{align*}
\text{Maximize} & \quad \sum_{i=1, j=1}^{N, T} y_{i,j} \\
\text{Subject to} & \quad \sum_{j=0}^{t-1} y_{i,j} - x_{i,j} \geq 0; \quad \forall i \in \{0, 1, \ldots, N\}, \forall t \in \{1, 2, \ldots, T\} \tag{1}
\end{align*}
\]
Claim 9. The IP (3.2) has a feasible solution, within T timeslots, with objective equal to N, if and only if there exists a correct interference-free broadcast schedule with latency T.

Proof. The first condition of Definition 6 is equivalent to Constraint 1 above. The third condition of Definition 6 is equivalent to the objective function of the IP being N. The fourth condition of Definition 6 implies Constraint 2 above, and is implied by Constraint 2 if the objective function of the IP is N. The second condition of Definition 6 is equivalent to constraints 3 and 4 above. Indeed, Constraint 3 says that if node i is to receive the message in timeslot j (y_{i,j} = 1), one node which has i in its transmission range must transmit (\sum_{r \neq i} x_{r,j} \geq 1). Also, no other node should interfere: Constraint 4 makes \sum_{r \neq i} x_{r,j} \leq 1 whenever y_{i,j} = 1. The fifth condition of Definition 6 is equivalent to Constraint 8 above. \qed

3.3. Integrality Gap

Constraint 4 can be slightly improved to:

\[ \sum_{r \neq i} x_{r,j} + (\Delta_i - 1)y_{i,j} \leq \Delta_i; \quad \forall i \in \{1, 2, \ldots, N\}, \forall j \in \{1, 2, \ldots, T\}. \]

In respect to the integrality gap, the relevant question is: If the linear programming relaxation, obtained by replacing Constraints (5) by 0 \leq y_{i,j} and 0 \leq x_{i,j}, has objective N for a certain value T, is it true that the integer program has objective N for \beta T? \beta would be the approximation ratio.

The linear programming relaxation always has a solution with objective value N for T = D + 1: If vertex v is at distance d > 0 from the root, set y_{v,d} = x_{v,d+1} = y_{v,d+2} = x_{v,d+2} = 1/2, and set x_{0,1} = x_{0,2} = 1/2. In non-Euclidean instances, as in Subsection 1.3, \beta could be as large as Θ(\log^2 n), and in Euclidean instances \beta could be as large as Θ(\alpha), as shown later in Lemma 10. In conclusion, this linear programming relaxation is not useful for an approximation algorithm.

4. Greedy Heuristics

In this section, we present six variations of greedy algorithms. Here, we define B_i as the set of nodes which transmit the message in timeslot i and R_j as the set of nodes which receive the message in timeslot j. R_0 = \{source\}.
4.1. Interference-Free Greedy

We term IF-FA the Interference-Free First Available greedy heuristic (Algorithm 1 has the pseudocode). The heuristic starts at timeslot 0 - when the source is put in \( R_0 \). Then, for \( k \), the next timeslot, the heuristic finds from all \( R_j \) with \( 0 \leq j < k \), the first node \( v_t \) such that, when \( v_t \) transmits, it does not interfere with the reception of existing nodes in \( R_k \), and existing nodes in \( B_j \) do not interfere with nodes receiving from \( v_t \).

Once that node \( v_t \) is found, the heuristic puts it in \( B_k \), and puts the additional receivers in \( R_k \). The process continues until all nodes are checked once as candidates for \( v_t \). After that, the heuristic begins the next timeslot and repeats until all the nodes have been put in some \( R_j \).

To reduce the running time (analyzed using the pseudocode), we use the following implementation. For each node \( v \), we keep as linked lists \( I_v \), the set of nodes within interference range of \( v \), and \( C_v \), the set of nodes within transmission/communication range of \( v \). We also keep \( M_j = \bigcup_{i=0}^{j-1} R_i \), and \( Q_j \) as the set of nodes not in \( M_j \) interfered by \( B_j \). The sets \( R_j, B_j, M_j \) and \( Q_j \) are kept as bit vectors. The pseudocode is given in Algorithm 1.

**Algorithm 1 Interference-Free First Available greedy**

1: \( \text{timeSlot}=0; \ M_1 = \{ \text{source} \} \)
2: while \( (M_{\text{timeSlot+1}} \neq V) \) do
3: \( \text{timeSlot++} \)
4: Initialize \( B_{\text{timeSlot}} = \emptyset, R_{\text{timeSlot}} = \emptyset, Q_{\text{timeSlot}} = \emptyset \)
5: for each node \( v \) do
6: if \( (v \in M_{\text{timeSlot}}) \) then
7: \( v \) is good
8: for each node \( w \in I_v \) do
9: if \( (w \in R_{\text{timeSlot}}) \) then
10: \( v \) is bad
11: end if
12: end for
13: for each node \( w \in C_v \) do
14: if \( (w \in Q_{\text{timeSlot}}) \) then
15: \( v \) is bad
16: end if
17: end for
18: if \( (v \text{ still good}) \) then
19: \( B_{\text{timeSlot}} = B_{\text{timeSlot}} \cup \{v\} \)
20: \( R_{\text{timeSlot}} = R_{\text{timeSlot}} \cup C_v \setminus M_{\text{timeSlot}} \)
21: \( Q_{\text{timeSlot}} = Q_{\text{timeSlot}} \cup I_v \setminus M_{\text{timeSlot}} \)
22: end if
23: end if
24: end for
25: \( M_{\text{timeSlot+1}} = M_{\text{timeSlot}} \cup R_{\text{timeSlot}} \)
26: end while
27: Output \( \text{timeSlot} \)

For a fixed \( j \), it takes \( O(n) \) time to initialize \( B_j, R_j, \) and \( Q_j \), and to update \( M_j \). All the conditions the \( \text{if} \) statements or the \( \text{for} \) statements must check can be checked in constant time.
Thus the running time is $O(n^2)$ per timeslot. At least one node is added each timeslot, as when $B_j = R_j = \emptyset$, the connectivity of the communication graph implies there is an edge in this graph with one endpoint in $M_j$ and one outside $M_j$; the endpoint in $M_j$ would enter $B_j$. Thus the number of timeslots does not exceed $n$, and the total running time of IF-FA (Algorithm 1) is $O(n^3)$. A more careful analysis yields $O(n \cdot \sum_{i=1}^{n-1} (|I_i| + 1))$.

The Interference-Free Max Available (IF-MA) greedy heuristic differs from IF-FA in that IF-MA finds an available node which yields the maximum number of new receivers among all the available nodes. That is, we do not execute lines 18-22 of Algorithm 1 for the first “good” $v$ met, but first compute for each such $v$ the number of receivers of node $v$ not in $M_{\text{timeslot}}$, and only execute lines 18-22 for the $v$ maximizing the number above. This comes at the expense of longer running time since before updating $B_j$ we try all possible nodes as $v$. And this extra loop can cause another $n$ rounds; therefore the running time of this heuristic is $O(n^2)$.

4.2. Interference-Aware Marginal Greedy

We term IA-FAA the Interference-Aware First Available Marginal greedy heuristic (Algorithm 2 has the pseudocode). The heuristic starts at timeslot 0 when the source is put in $R_0$. Then, for $k$, the next timeslot, the heuristic finds from all $R_j$ with $0 \leq j < k$, the first node $v_i$ such that, when $v_i$ transmits, it does not interfere with the reception of existing nodes in $R_k$, and there are new receivers of $v_i$ that are not receiving interference from existing nodes of $B_k$.

Once that node $v_i$ is found, the heuristic puts it in $B_k$ and puts the additional receivers in $R_k$. The process continues until all nodes are checked once as candidates for $v_i$. After that, the heuristic begins the next timeslot and repeats until all the nodes have been put in some $R_j$.

To reduce the running time (analyzed after pseudocode), we use the following data structures, as in the previous algorithm. Precisely, for each node $v$, we keep as linked lists $I_v$, the set of nodes within interference range of $v$, and $C_v$, the set of nodes within transmission/communication range of $v$. We also keep $M_j = \bigcup_{i=0}^{n} R_i$, and $Q_j$ as the set of nodes not in $M_j$ interfered by $B_j$. The sets $R_j$, $B_j$, $M_j$ and $Q_j$ are kept as bit vectors. The pseudocode is given in Algorithm 2.

For a fixed $j$, it takes $O(n)$ time to initialize $B_j$, $R_j$, and $Q_j$, and to update $M_j$. All the conditions the if statements or the for statements must check can be checked in constant time. Thus the running time is $O(n^2)$ per timeslot. At least one node is added each timeslot, as when $B_j = R_j = \emptyset$, the connectivity of the communication graph implies there is an edge in this graph with one endpoint in $M_j$ and one outside $M_j$; the endpoint in $M_j$ would enter $B_j$. Thus the number of timeslots does not exceed $n$, and the total running time of IF-FA (Algorithm 2) is $O(n^3)$. A more careful analysis yields $O(n \cdot \sum_{i=1}^{n-1} (|I_i| + 1))$.

The Interference-Aware Max Available Marginal (IA-MAM) greedy heuristic differs from IA-FAA in that IA-MAM finds an available node which gives the maximum number of new receivers. We keep track for each eligible $v$ of the quantity $|C_v \setminus (Q_{\text{timeslot}} \cup M_{\text{timeslot}})|$ in lines 13-17 instead of just setting the progress variable. Lines 18-22 of Algorithm 2 are not executed for the first “good” $v$ with progress, but for the $v$ maximizing the quantity above. This comes at the expense of longer running time since before updating $B_j$ we try all possible nodes as $v$. And this extra loop can cause another $n$ rounds; therefore the running time of this heuristic is $O(n^2)$.

4.3. Interference-Aware Accumulated Greedy

We term IA-AFA the Interference-Aware First Available Accumulated greedy heuristic (Algorithm 3 has the pseudocode). The heuristic starts at timeslot 0 when the source is put in $R_0$. Then, for $k$, the next timeslot, the heuristic finds from all $R_j$ with $0 \leq j < k$, the first node $v_i$
Algorithm 2 Interference-Aware First Available Marginal greedy

1: \( \text{timeSlot} = 0; \ M_1 = \{\text{source}\} \)
2: \( \text{while} \ (M_{\text{timeSlot}+1} \neq V) \ do \)
3: \( \text{timeSlot}++ \)
4: \( \text{Initialize} \ B_{\text{timeSlot}} = \emptyset, \ R_{\text{timeSlot}} = \emptyset, \ Q_{\text{timeSlot}} = \emptyset \)
5: \( \text{for} \ (\text{each node } v) \ do \)
6: \( \text{if} \ (v \in M_{\text{timeSlot}}) \ then \)
7: \( v \text{ is good; progress is bad} \)
8: \( \text{for} \ (\text{each node } w \in I_v) \ do \)
9: \( \text{if} \ (w \in R_{\text{timeSlot}}) \ then \)
10: \( v \text{ is bad} \)
11: \( \text{end if} \)
12: \( \text{end for} \)
13: \( \text{end for} \)
14: \( \text{if} \ (w \notin Q_{\text{timeSlot}} \cup M_{\text{timeSlot}}) \ then \)
15: \( \text{progress is good} \)
16: \( \text{end if} \)
17: \( \text{end for} \)
18: \( \text{if} \ (v \text{ and progress are good}) \ then \)
19: \( B_{\text{timeSlot}} = B_{\text{timeSlot}} \cup \{v\} \)
20: \( R_{\text{timeSlot}} = R_{\text{timeSlot}} \cup (C_v \setminus (Q_{\text{timeSlot}} \cup M_{\text{timeSlot}})) \)
21: \( Q_{\text{timeSlot}} = Q_{\text{timeSlot}} \cup I_v \setminus M_{\text{timeSlot}} \)
22: \( \text{end if} \)
23: \( \text{end if} \)
24: \( \text{end for} \)
25: \( M_{\text{timeSlot}+1} = M_{\text{timeSlot}} \cup R_{\text{timeSlot}} \)
26: \( \text{end while} \)
27: \( \text{Output } \text{timeSlot} \)
such that, when \( v_t \) transmits, the number of new receivers of \( v_t \) that are not within interference range from existing nodes of \( B_t \) exceeds the number of existing receivers from \( R_t \) that are within interference range from \( v_t \).

Once such node \( v_t \) is found, the heuristic puts it in \( B_t \) and updates \( R_t \). The process continues until all nodes are checked once as candidates for \( v_t \). After that, the heuristic begins the next timeslot and repeats until all the nodes have been put in some \( R_t \).

To reduce the running time (analyzed after pseudocode), we use the following data structures, as in the previous algorithms. Precisely, for each node \( v \), we keep as linked lists \( I_v \), the set of nodes within interference range of \( v \), and \( C_v \), the set of nodes within transmission/communication range of \( v \). We also keep \( M_j = \bigcup_{i=0}^{j-1} R_t \), and \( Q_j \) as the set of nodes not in \( M_j \) interfered by \( B_j \). The sets \( R_t, B_j, M_j \) and \( Q_j \) are kept as bit vectors. The pseudocode is given in Algorithm 3.

**Algorithm 3 Interference-Aware First Available Accumulated greedy**

1. \( \text{timeSlot}=0; M_1 = \{\text{source}\} \)
2. while \((M_{\text{timeSlot}+1} \neq V)\) do
3. \( \text{timeSlot}++ \)
4. Initialize \( B_{\text{timeSlot}} = \emptyset, \quad R_{\text{timeSlot}} = \emptyset, \quad Q_{\text{timeSlot}} = \emptyset \)
5. for (each node \( v \)) do
6. if \((v \in M_{\text{timeSlot}})\) then
7. if \((|R_{\text{timeSlot}} \cap I_v| < |C_v \setminus (Q_{\text{timeSlot}} \cup M_{\text{timeSlot}})|)\) then
8. \( B_{\text{timeSlot}} = B_{\text{timeSlot}} \cup \{v\} \)
9. \( R_{\text{timeSlot}} = (R_{\text{timeSlot}} \setminus I_v) \cup (C_v \setminus (Q_{\text{timeSlot}} \cup M_{\text{timeSlot}})) \)
10. \( Q_{\text{timeSlot}} = Q_{\text{timeSlot}} \cup I_v \setminus M_{\text{timeSlot}} \)
11. end if
12. end if
13. end for
14. \( M_{\text{timeSlot}+1} = M_{\text{timeSlot}} \cup R_{\text{timeSlot}} \)
15. end while
16. Output \( \text{timeSlot} \)

For a fixed \( j \), it takes \( O(n) \) time to initialize \( B_j, R_j, \) and \( Q_j \) in line 4, and to update \( M_j \) in line 14. In lines 7-11, all the set operations including counting can be done in \( O(|I_v| + 1) \) by going through \( I_v \) and \( C_v \) and using the bitmaps of \( R_j, Q_j, \) and \( M_j \). Thus the running time is \( O(n^2) \) per timeslot. At least one node is added each timeslot, as when \( B_j = R_j = \emptyset \), the connectivity of the communication graph implies there is an edge in this graph with one endpoint in \( M_j \) and one outside \( M_j \); the endpoint in \( M_j \) would enter \( B_j \). Thus the number of timeslots does not exceed \( n \), and the total running time of IA-FAA (Algorithm 3) is \( O(n^2) \). A more careful analysis yields \( O(n \cdot \sum_{i=0}^{N} |I_i| + 1) \).

The Interference-Aware Max Available Accumulated (IA-MAA) greedy heuristic differs from IA-FAA in that IA-MAA finds an available node that gives the maximum increase in \( R_t \). We keep track for each \( v \in M_j \) of the quantity \(|C_v \setminus (Q_{\text{timeSlot}} \cup M_{\text{timeSlot}})| - |R_{\text{timeSlot}} \cap I_v| \). Lines 8-10 of Algorithm 3 are not executed for the first \( v \) with positive quantity above, but for the \( v \) maximizing the quantity. This comes at the expense of longer running time since before updating \( B_j \) we try all possible nodes as \( v \). And this extra loop can cause another \( n \) rounds; therefore the running time of this heuristic is \( O(n^3) \).
5. Experimental Results

In this section, we show the results of experiments for all six variations of the greedy heuristic. Considering networks of nodes randomly distributed over a $4 \times 4$ area, and the source node is also randomly located within the area, we vary number of nodes, source included, to 21, 41, 61, and 81. Based on the UDG model, the transmission range of each node is set to 1, while the interference range is set to 2. We generate 20 instances for each network size and use the same instances for all IPs and heuristics. Note that information shown in all tables are the average of the mentioned number of instances on each heuristic. We compare the results from heuristics and optimum solutions obtained from solving integer programs. Due to the NP-Complete hardness of Integer Programming and computing capacity limitation of our computer, we cannot timely solve an instance with much more than 81 nodes. Note that our HP-XW8000 has spent an average of 6 hours to solve IPs for an 81-instance but a couple of seconds to run the heuristics.

Table 1 shows the depth of BFS trees of the given instances and optimum solution obtained from related IPs. Note that the interference-free IP yields slightly higher solutions than those obtained in the interference-aware model, and both are higher than the depth of BFS tree, which is the lower bound. Recall that the IP in subsection 3.2 has the constraints of the one from subsection 3.1, and has an additional set of constraints.

<table>
<thead>
<tr>
<th>#nodes</th>
<th>BFS</th>
<th>Intf-aware IP</th>
<th>Intf-free IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>5.7</td>
<td>7.1</td>
<td>7.1</td>
</tr>
<tr>
<td>41</td>
<td>6.3</td>
<td>7.4</td>
<td>7.4</td>
</tr>
<tr>
<td>61</td>
<td>5.9</td>
<td>7.4</td>
<td>7.6</td>
</tr>
<tr>
<td>81</td>
<td>5.7</td>
<td>7.5</td>
<td>7.7</td>
</tr>
</tbody>
</table>

Next, in Table 2, we compare the experimental results among heuristics and optimum solutions in the interference-aware category. The greedy heuristics based on maximum available improvement, i.e., IA-MAM and IA-MAA, overcome the other two heuristics based on first available improvement. The more greedy, the better. But one may need to trade-off the running time because, on maximum available concept, IA-MAM and IA-MAA have to run through all the capable nodes before selecting one to transmit, while IA-FAM and IA-FAA run through all the capable nodes only once per timeslot. Also note that more complex IA-MAA usually gives better solutions than IA-MAM does. As described in details in section 4, IA-MAA has to re-compute the set of receivers $R$ in that timeslot every time an additional node is being considered, while IA-MAM computes only the number of receivers gained by that additional node.

The comparison in interference-free category is shown in Table 3. As earlier, the more greedy version, IF-MA, performs significantly better than the simpler version, IF-FA. One can also observe that the results of heuristics in the interference-free category is slightly higher than those of the interference-aware category.

Tables 4 and 5 show the differences in percentage of outputs from each heuristic over optimum solutions. The best heuristic for each instance gives the output exceeding the optimum by only 13-20%.
Table 2: Average optimum interference-aware and related heuristics

<table>
<thead>
<tr>
<th>#nodes</th>
<th>Intf-aware IP</th>
<th>IA-FAM</th>
<th>IA-MAM</th>
<th>IA-FAA</th>
<th>IA-MAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>7.1</td>
<td>8.6</td>
<td>8.5</td>
<td>8.2</td>
<td>8.1</td>
</tr>
<tr>
<td>41</td>
<td>7.4</td>
<td>8.8</td>
<td>8.5</td>
<td>8.9</td>
<td>8.6</td>
</tr>
<tr>
<td>61</td>
<td>7.4</td>
<td>9.5</td>
<td>8.9</td>
<td>9.4</td>
<td>9.0</td>
</tr>
<tr>
<td>81</td>
<td>7.5</td>
<td>9.7</td>
<td>8.7</td>
<td>9.7</td>
<td>8.5</td>
</tr>
</tbody>
</table>

Table 3: Average optimum interference-free and related heuristics

<table>
<thead>
<tr>
<th>#nodes</th>
<th>Intf-free IP</th>
<th>IF-FA</th>
<th>IF-MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>7.1</td>
<td>8.5</td>
<td>8.4</td>
</tr>
<tr>
<td>41</td>
<td>7.4</td>
<td>8.9</td>
<td>8.6</td>
</tr>
<tr>
<td>61</td>
<td>7.6</td>
<td>9.5</td>
<td>9.1</td>
</tr>
<tr>
<td>81</td>
<td>7.7</td>
<td>10.0</td>
<td>8.7</td>
</tr>
</tbody>
</table>

6. Theoretical Results

Lemma 10. There are Euclidean instances of IABS that have minimum latency at least $(\pi/6)(\alpha - 2)D$.

Proof. We assume $\alpha > 2$, or else there is nothing to prove. Take a square grid with adjacent points at distance $(\sqrt{2}/2)(1 + \epsilon)$, where $\epsilon$ is suitable small. Pick an arbitrary point on the grid, as the source $s$, and keep only the points at Euclidean distance at most $\alpha/2$ from $s$. This is the Euclidean instance of IABS.

The points remaining make a UDG with vertex degree at most 4. If we want to maximize graph distance from $s$ while keeping Euclidean distance small we alternate horizontal and vertical edges (see Figure 4). Therefore $D \leq 2(\alpha/2) = \alpha$.

To prove the lemma, we first define the outer disk as the disk centered at the source $s$ with radius $\alpha/2$. Let $n$ be the number of nodes in the instance mentioned above, which is equal to the number of grid points within this outer disk.

Then, we draw a square whose area is $(1/2)(1+\epsilon)^2$, i.e., any side of the square is $(\sqrt{2}/2)(1+\epsilon)$, centered at each grid point. If any arbitrary point $y$ is within the area of the square, $y$ is said to be

Table 4: Differences in percentage between related heuristics and optimum, the interference-aware case

<table>
<thead>
<tr>
<th>#nodes</th>
<th>IA-FAM</th>
<th>IA-MAM</th>
<th>IA-FAA</th>
<th>IA-MAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>21</td>
<td>20</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>41</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>61</td>
<td>28</td>
<td>20</td>
<td>26</td>
<td>21</td>
</tr>
<tr>
<td>81</td>
<td>29</td>
<td>16</td>
<td>29</td>
<td>13</td>
</tr>
</tbody>
</table>
Table 5: % Differences in percentage between related heuristics and optimum, the interference-free case

<table>
<thead>
<tr>
<th>#nodes</th>
<th>IF-FA</th>
<th>IF-MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>41</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>61</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>81</td>
<td>30</td>
<td>13</td>
</tr>
</tbody>
</table>

covered by that grid point. We further draw another circle, called inner disk, centered at \(s\) with 
\((α − (1 + ǫ))/2\) radius (see also Figure 4).

Let \(b\) (see Figure 5) be an arbitrary point within the inner disk and covered by a grid point \(c\) which is outside the inner disk. The distance between \(b\) and \(c\) is at most \((1 + ǫ)/2\), as \(b\) is in a square centered at \(c\) and with sides of length \((\sqrt{2}/2)(1 + ǫ)\). Now we can show that \(c\) is within the outer disk.

Since \(b\) is in the inner disk as given, the distance between \(b\) and the source \(s\) is at most \((α − (1 + ǫ))/2\), the radius of the inner disk, now we have:

\[
|s, c| \leq |s, b| + |b, c| \leq (α − (1 + ǫ))/2 + (1 + ǫ)/2 \leq α/2. \tag{3}
\]

We conclude that \(c\) is within the outer disk.

Because the area of the inner disk is covered by \(n\) grid points within the outer disk, by comparing areas, we can compute \(n\) as the following:

\[
\begin{align*}
n(1/2)(1 + ǫ)^2 & \geq \pi((α − (1 + ǫ))/2)^2 \tag{4} \\
(n/2)(1 + ǫ)^2 & \geq (π/4)(α − (1 + ǫ))^2 \tag{5} \\
n & \geq ((π/2)(α − (1 + ǫ))^2)/(1 + ǫ)^2 \tag{6}
\end{align*}
\]

Any transmitting node will cause interference at all the other nodes, since the nodes are all in a disk of diameter \(α\). So no two nodes can transmit successfully at the same time. A transmission by node \(v ≠ s\) can reach at most three new vertices - as one neighbor of \(v\) must have gotten the message before \(v\).

The latency \(T\) of this instance can be derived as follows: after one time slot, there will be five nodes which received the message, and after that in each round only three more nodes can receive the message. Thus after \(j\) time slots at most \(5 + 3(j − 1)\) can receive the message, and therefore:

\[
n ≤ 5 + 3(T − 1). \tag{7}
\]

Recall that \(D ≤ α\). Using this and Equations 6 and 7, we obtain:

\[
\begin{align*}
T & ≥ n/3 − 5/3 + 1 \tag{8} \\
     & ≥ n/3 − 2/3 \tag{9} \\
     & ≥ ((π/6)(α − (1 + ǫ))^2)/(1 + ǫ)^2 − 2/3 \tag{10}
\end{align*}
\]
Figure 4: Maximized graph distance: $R$ denotes $\alpha/2$, $r$ denotes $(\alpha - (1 + \epsilon))/2$

Figure 5: Point coverage. The grid is given by the dotted lines, and the solid lines separate and define the coverage areas of grid points.
\[
\frac{\pi}{6}(\alpha - (1 + \epsilon)^2)D/\alpha - 2/3 \geq (\pi/6)(\alpha - 2(1 + \epsilon) + (1 + \epsilon^2)\alpha (1 + \epsilon))D - 2/3
\]

We can find some \(\epsilon\) that makes \(T \geq (\pi/6)(\alpha - 2)D - 1\). \(\Box\)

Our main technical result is:

**Theorem 11.** Let \(I = (V, s, \alpha)\) be an Euclidean instance of IABS with \(|V| = n\) nodes, source \(s\) and interference range \(\alpha\), and let \(G = (V, E)\) be the communication graph of \(I\) and \(D\) be the radius of \(G\) with respect to \(s\). There is a centralized \(O(n^2)\) algorithm to produce an interference-free broadcast schedule of \(I\) with at most \(64(2 + \lceil \sqrt{2(\alpha + 3)} \rceil)D + 8(\lceil \sqrt{2(\alpha + 3)} \rceil)^2 + 1\) timeslots.

**Proof.** Note that the communication graph \(G = (V, E)\) of \(I\) is a UDG. First we partition the plane into squares of diameter 1 (thus the side of a square has length \(\sqrt{2}/2\)), such that no node is on the border of such a square. Then we construct an auxiliary graph \(H = (V(H), E(H))\) as follows: \(V(H)\) has one vertex for each cell that contains a node of \(V\), and two vertices/cells of \(V(H)\) are adjacent if there exist two nodes of \(V\), one in each cell, that are adjacent in \(V\). Let \(H' = (V(H'), E(H'))\) be the bi-directed version of \(H\); that is \(V(H') = V(H)\) and for each edge \(e = [u, v]\) of \(E(H)\), \(H'\) has two directed arcs: \(uv\) and \(vu\). See Figure 6 for an illustration.

Figure 6: Ignoring the arrows and numbers for the moment, we use solid line segments to represent a tree \(T\) in the UDG \(G\) and dotted and dashed segments and arcs to represent \(H\). The arrows and \(T\) are used for an illustration of Claim 12: if the conflict-free broadcast schedule in \(H'\) uses the dashed line segments with arrows at timeslots given by the numbers next to each arc, the construction of the claim results in the timeslots given next to each vertex of \(G\) (there is indeed redundancy), with communication following the arrows of \(T\).
For a vertex \( v \in V \), denote by \( \text{cell}(v) \) the cell of \( V(H) \) containing \( v \). For every arc \( e' \in E(H') \), pick adjacent nodes of \( V \), \( t(e') \) and \( h(e') \), such that \( \text{cell}(t(e')) \) is the tail of \( e' \) (in \( H' \)) and \( \text{cell}(h(e')) \) is the head of \( e' \) (in \( H' \)). Such nodes \( t(e') \) and \( h(e') \) must exist, as otherwise \( e' \) cannot be an arc of \( H' \), and if there are several choices for \( t(e') \) and \( h(e') \), make an arbitrary choice. Note that constructing \( H' \), and the functions \( \text{cell} : V \to V(H), t : E(H') \to V, \) and \( h : E(H') \to V \) can be easily done in \( O(n^2) \).

Call two arcs \( e_1 \) and \( e_2 \) of \( H' \) conflicting if \(|t(e_1), t(e_2)| \leq \alpha + 3 \). Let \( Q \) be a subgraph of \( H' \). A conflict-free broadcast schedule for \( Q \) with root \( r \in V(Q) \) and \( t \) timeslots consists of \( t \) sets of arcs of \( E(Q), X_1, X_2, \ldots, X_t \), such that:

1. for each \( i \in \{1, 2, \ldots, t\} \), no two arcs of \( X_i \) are conflicting, and
2. for any \( i \in \{1, 2, \ldots, t\} \) and any arc \( e \in X_i \), either the tail of \( e \) is \( r \), or there exist \( i' < i \) and arc \( e' \in X_{i'} \) such that the tail of \( e \) is the head of \( e' \), and
3. any vertex of \( V(Q) \setminus r \) is the head of some arc in \( \bigcup_{i=1}^{t} X_i \).

**Claim 12.** If \( H' \) has a conflict-free broadcast schedule with \( t \) timeslots rooted at \( \text{cell}(s) \), then \( I \) has an interference-free broadcast schedule with source \( s \) and \( 2t + 1 \) timeslots. Moreover, constructing the schedule for \( I \) from the one of \( H' \) can be done in \( O(n^2) \).

**Proof.** For an illustration, see again Figure 6.

For \( i = 1, 2, \ldots, t \), define \( B_{2} = \{t(e') \mid e' \in X_i\} \) and \( B_{2t+1} = \{h(e') \mid e' \in X_i\} \). Also define \( R_0 = B_1 = \{s\} \), and for \( j = 1, 2, \ldots, 2t + 1 \), define \( R_j = \{v \in V \mid \exists v' \in B_j \text{ such that } vv' \in E(G)\} \setminus \bigcup_{i=0}^{j-1} R_i \). Doing this construction in \( O(n^2) \) is straightforward. Now we verify that this is a correct interference-free broadcast schedule for \( I \); we note that it has \( s \) as the source and \( 2t + 1 \) timeslots.

Indeed, the lack of interference/collisions follows from the fact that for all \( i = 1, 2, \ldots, t \), any two nodes of \( B_2 \) are at Euclidean distance at least \( (\alpha + 3) \) (since the nodes come from non-conflicting arcs of \( E(H') \)), and any two nodes of \( B_{2t+1} \) are at Euclidean distance at least \( (\alpha + 1) \). This last statement is true since, otherwise, if \( v_1, v_2 \in B_{2t+1} \) are such that \( |v_1, v_2| \leq \alpha + 1 \), and \( v_1 = h(e_1) \) and \( v_2 = h(e_2) \), with \( e_1, e_2 \in X_i \), then, using that for every \( e \in E(H') \), \( t(e) \) and \( h(e) \) are adjacent in \( G \), \( |h(e_1), h(e_2)| \leq |t(e_1), h(e_1)| + |t(e_2), h(e_2)| \leq 1 + (\alpha + 1) + 1 = \alpha + 3 \), and therefore \( e_1 \) and \( e_2 \) are conflicting, a contradiction to the fact we started with a non-conflicting broadcast schedule for \( H' \).

Next, we note that \( B_1 \subseteq R_0 \), that for \( i = 1, 2, \ldots, t \), \( B_{2t+1} \subseteq \bigcup_{j=0}^{2t+1} R_j \) since every vertex of \( B_{2t+1} \) is adjacent in \( G \) to some vertex of \( B_2 \) (for every \( e' \in E(H') \), \( t(e') \) and \( h(e') \) are adjacent in \( G \)). Also, for \( i = 1, 2, \ldots, t \), every vertex in \( B_{2t+1} \) is \( t(e) \) for some \( e \in X_i \). Since we started with a conflict-free broadcast schedule of \( H' \), either the tail of \( e \) is \( \text{cell}(s) \), or there exist \( i' < i \) and arc \( e' \in X_i \) such that the tail of \( e \) is the head of \( e' \). In the first case, \( t(e) \in \text{cell}(s) \) and therefore \( t(e) \in (R_0 \cup R_i) \). In the second case, \( t(e) \) and \( h(e') \) are in the same cell and thus adjacent in \( G \), and since \( h(e') \in B_{2t+1} \), we deduce that \( t(e) \in \bigcup_{j=0}^{2t+1} R_j = \bigcup_{j=0}^{2t+1} R_j \).

Finally, \( \bigcup_{i=0}^{2t+1} R_i = V \), as we prove in this paragraph. If \( v = s \), then \( v \in R_0 \), and if \( \text{cell}(v) = \text{cell}(s) \), then \( v \in R_1 \). Otherwise, there exist some \( i \in \{1, 2, \ldots, t\} \) and \( e \in X_i \) such that \( \text{cell}(e) \) the head of the arc \( e \) is \( \text{cell}(v) \). Then, unless \( v \in \bigcup_{j=0}^{2t+1} R_j \), \( v \) is put in \( R_{2t+1} \), as \( \text{cell}(v) = \text{cell}(h(e)) \), \( h(e) \in B_{2t+1} \), and the Euclidean distance between two nodes of the same cell is at most 1. Thus we can convert a conflict-free broadcast schedule of \( H' \) into an interference-free broadcast schedule of \( I \). \qed
We continue with the proof of Theorem 11. Observe that the radius of $H'$, starting from $\text{cell}(s)$, is at most $D$. Indeed, for any vertices $v_1$ and $v_2$ adjacent in $V$, $\text{cell}(v_1)$ and $\text{cell}(v_2)$ are either identical or adjacent in $H$, and thus any path in $G$ has an equivalent path in $H'$ of at most the same length.

Based on the claim above, to prove the theorem it is enough to construct in $H'$ a conflict-free broadcast schedule from $\text{cell}(s)$ with $32(2 + \lceil \sqrt{2}(\alpha + 3) \rceil)D + 4(\lceil \sqrt{2}(\alpha + 3) \rceil)^2$ timeslots. Let $T_H$ be a BFS tree of $H$ rooted at $\text{cell}(s)$.

We further partition the Euclidean plane, and $V(H)$, into square blocks, where a block has $\lceil \sqrt{2}(\alpha + 3) \rceil \times \lceil \sqrt{2}(\alpha + 3) \rceil$ cells. Note that a block occupies a square with sides of length $(\alpha + 3)$ as each cell has sides of length $\sqrt{2}/2$. We group the blocks together into bigger squares, each containing four blocks, and color the four blocks in a group with colors 0, 1, 2, and 3 according to the four quadrants. See Figure 7 for an illustration.

![Figure 7: Block coloring with four colors, where each block has $\lceil \sqrt{2}(\alpha + 3) \rceil \times \lceil \sqrt{2}(\alpha + 3) \rceil$ cells](image)

The blocks are large enough such that, if we follow the rule of using only blocks of the same color and for each such block picking only one arc of $E(H')$ with the tail in the block, we do not pick any pair of conflicting arcs. We construct our non-conflicting broadcast schedule for $H'$ using this rule, and the tree $T_H$, as explained below.

In timeslot $i$, we only use blocks colored $i \mod 4$. The schedule has two phases, inter-block and fill-block, with the intuition that in the inter-block phase we reach blocks, and in the fill-block phase we fill them.

First we prune $T_H$ with the goal that we only keep an arc $e$ if $e$ is needed to reach from $\text{cell}(s)$ cells in blocks other than the block containing the tail of $e$. For an illustration, see Figure 8, ignoring the paths $P_e$ for the moment. This is accomplished as follows. Process $T_H$ in postorder, and for each $v \in V(T_H)$, construct a list (as a bitmap) with the coordinates of the blocks where the descendants of $v$, including itself, lie. In a second postorder traversal, for every $v \neq \text{cell}(s)$ of $T_H$, remove $v$ from $T_H$ if the list of $v$ has only one element and the parent of $v$ is in the same block as $v$. Call $T'$ the tree obtained after these removals; it is indeed a tree since for every vertex removed from $T_H$, its descendants are also removed. We treat $T'$ as directed, with arcs going from parent to child. Note that every arc of $T_H$ with tail and head in different blocks is in $T'$.

The construction $T'$ from $T_H$, done as described above, takes $O(n^2)$ time. In the inter-block phase, we make sure that each vertex of $V(T') \setminus \{\text{cell}(s)\}$ appears as the head of some arc in some
Figure 8: The arcs entering and inside a block (the square inside the solid lines). The dotted arcs and the vertices they reach do not appear in the pruned tree $T'$, which contains only solid and dashed arcs. The inter-block phase described later uses paths $P_i$, four of which appear here with dashed arcs. For example, the path $P_4$ is in fact $P_{e_4}$, where $e_4$ has label 65 here. The possible timeslots could be: $P_1$ has two arcs with timeslots 5 and 9; $P_2$ has seven arcs with timeslots 25, 33, 41, 45, 53, 61, and 69; $P_3$ has two arcs with timeslots 29 and 37; $P_4$ has three arcs with timeslots 49, 57, and 65.

$X_i$. This is accomplished as follows.

**Detailed description of the inter-block phase**

The algorithm proceeds timeslot by timeslot and constructs the sets $X_i$ starting with $X_1$ and then incrementing $i$ - once it starts working on timeslot $i$ it will not modify $X_j$ for $j < i$. At timeslot $j$, call a cell $v \in V(H)$ reached if $v = \text{cell}(s)$ or $v$ is the head of some arc in $\bigcup_{i=0}^{j-1} X_i$.

Consider a block $B$. Let $\text{Out}(B)$ be the set of arcs of $T'$ with the tail in $B$ and the head outside $B$. See Figure 9. For each arc $e \in \text{Out}(B)$, consider the subpath $P_e$ of $T'$ which ends with $e$ and is shortest starting at either $\text{cell}(s)$ or some vertex of $H$ such that the parent in $T'$ of that vertex is outside $B$. For an illustration, see Figure 8.

Figure 9: The arcs going out of a block $B$ could look as above. The solid arcs are some (but not all) edges of $G$. 

21
Initialization: For each arc \( e \in E(T') \), we compute if there is some block \( B \) with \( e \in \text{Out}(B) \), and if so explicitly construct \( P_e \) as a linked list. For any such \( e \), we keep a pointer \( l_e \), initialized to \textit{NULL}, to arcs of \( P_e \). We also keep as a bitmap the set of vertices reached; initially only \textit{cell}(s)\ is reached. A path \( P_e \) is \textit{eligible} if its first vertex has been reached and its last vertex has not been reached; in an invariant of the algorithm that all these paths have \( l_e \neq \textit{NULL} \). For each block \( B \), we keep a queue \( Q_B \); an invariant of the algorithm is that, except during the processing a path \( P_e \), as described below, \( Q_B \) contains the eligible paths \( P_e \) for all \( e \in \text{Out}(B) \). This queue is initialized to be empty for every \( B \)'s which does not contain \textit{cell}(s). For every \( B \) which contains \textit{cell}(s), all the paths \( P_e \), for \( e \in \text{Out}(B) \), that have \textit{cell}(s)\ as the first vertex, are put in the queue \( Q_B \), and for any such \( e, l_e \) is set to be the first arc of \( P_e \).

Processing one timeslot: We process timeslots \( j \) in order starting with 1. For each block \( B \), if it is colored \( j \) mod 4 and \( Q_B \neq \emptyset \), then we extract from \( Q_B \) a path \( P_e \), which we process as follows. We put in \( X_f \) the arc pointed at by \( l_e \) and mark the head of \( e \) as reached. If this arc is not \( e \), then we make \( l_e \) point to the next arc of \( P_e \), and enqueue \( P_e \) (round-robin policy). Elise \( l_e \) does point at \( e \), we search for all possible \( P_e \) such that the head of \( e \) is the first vertex of \( P_e \). For each such \( e' \), we set \( l_e \) to be the first arc of \( P_{e'} \), and enqueue \( P_{e'} \) in \( Q_B \) of the appropriate \( B' \).

Figure 8 shows an example of repeatedly running this processing on a block colored 1 and with four paths.

Stopping: The inter-block phase finishes once every vertex of \( T' \) is reached, which we test using a bitmap after each timeslot.

Claim 13. The inter-block phase outputs a conflict-free broadcast schedule for \( T' \) with at most \( D \cdot 32(2 + [\sqrt{2}(\alpha + 3)]) \) timeslots.

Proof. It is crucial to notice \( |\text{Out}(B)| \leq 4 \cdot 2 \cdot (2 + [\sqrt{2}(\alpha + 3)]) \), since only the cells which are not in \( B \) but have a corner at Euclidean distance at most 1 from \( B \) could have an arc of \( T' \) incoming from \( B \) (see Figure 9), and \( T' \) being a tree implies that no such cell can have two incoming arcs.

The bound on \( |P_e| \) and the round-robin policy ensure that, once the first vertex of a path \( P_f \) is reached, in each interval of \( 4 \cdot 8 \cdot (2 + [\sqrt{2}(\alpha + 3)]) \) timeslots, \( P_e \) is processed and at least one arc of \( P_f \) is used. Thus, by immediate induction on \( |P_f| \) (here \( |P_f| \) is the length of the path \( P_f \)- the number of arcs in it), the head of \( f \) is reached after \( |P_f| \cdot 32(2 + [\sqrt{2}(\alpha + 3)]) \) timeslots from the moment the first vertex of \( P_f \) is reached.

For \( v \in \text{V}(T') \) we let \( d(v) \) be the length of the \( T' \)-path from \textit{cell}(s)\ to \( v \). This algorithm indeed reaches all vertices of \( T' \): any \( x \in T' \) either has its parent in a different block, in which case the arc from \( x's \) parent to \( x \) appears as \( e \) in some \( \text{Out}(B) \), or has a descendant in a different block. Let \( y \) be one such descendant of \( x \), closest to \( x \) in \( T' \). Then the arc from \( y's \) parent to \( y \) appears as \( e \) in \( \text{Out}(B) \), where \( x \in B \), and thus \( x \) is on \( P_e \). Therefore always \( x \) is on some \( P_e \), and then \( x \) is reached provided the first vertex of \( P_e \) is reached. This first vertex is also in \( T' \) and closer to \textit{cell}(s), the root of \( T' \). Thus we can apply induction on \( d(x) \) and obtain not only that \( x \) is reached, but also that it is reached after at most \( d(x) \cdot 32(2 + [\sqrt{2}(\alpha + 3)]) \) timeslots. \( \square \)

Claim 14. Computing the schedule of the inter-block phase takes \( O(n^2) \) time.

Proof. We have at most \( n \) arcs in \( T' \) and \( n \) blocks in total, so determining if \( e \in \text{Out}(B) \) for some \( B \) can be done in \( O(n^2) \). Constructing \( P_e \) for a given \( e \) can be done in \( O(n) \) time using parent pointers in \( T' \). Thus all the steps of Initialization take \( O(n^2) \) time.
All the steps of Processing one timeslot can be easily implemented to run in constant time per block and timeslot, except for searching, given a block \( B \) and one edge \( e \in \text{Out}(B) \), for all possible \( P_e \), with the head of \( e \) is the first vertex of \( P_e \). All (for every timeslot and every block) such searches can be done in \( O(n^2) \) since this is done exactly once for each \( e \), and in \( O(n) \) we can test \( e' \).

Testing the stopping condition is done in \( O(n) \) after each timeslot. Thus, except for the searches, the time spent by the algorithm processing one timeslot is \( O(n) \). The number of timeslots given by the previous claim may not be \( O(n) \), so we are not yet done.

Note that if there is \( \alpha \) unreached vertices in \( T' \) at timeslot \( j \), then at least an arc of \( T' \) will be put in one of \( X_j, X_{j+1}, X_{j+2}, X_{j+3} \), as argued next. Let \( x \) be an unreached vertex highest in \( T' \); \( x \)'s parent \( y \) is reached. We have two cases: if the arc \( xy \in \text{Out}(B) \) for some \( B \), then let \( e = xy \). If not, then \( y \) and \( x \) are in the same block \( B' \), and the construction of \( T' \) ensures that \( x \) has an descendant in another block and thus there exists an edge \( e' \in \text{Out}(B') \) such that both \( x \) and \( y \) belong to \( P_e \).

In either case, the path \( P_f \) (or \( P_e \)) is eligible, as it has both reached and unreached vertices, and since one cannot reach a vertex without reaching all its ancestors in \( T' \), the first vertex of the path must be reached and the last vertex unreached. Therefore, \( Q_f \) (or \( Q_e \)) is not empty. In both cases, \( B \) or \( B' \) will be processed in one of the timeslots \( j, j + 1, j + 2, j + 3 \) and thus one arc from some \( P_f \) (\( f \) may or may not be \( e \) or \( e' \)) will be put in one of \( X_j, X_{j+1}, X_{j+2}, X_{j+3} \).

\( T' \) has at most \( n - 1 \) arcs, and as long as there are unreached vertices, each four timeslots a new arc will be assigned to some \( X_j \). Thus there are at most \( 4n \) timeslots for the inter-block phase.

Combining the \( O(n) \) timeslots with \( O(n) \) time per timeslot (except for searches), \( O(n^2) \) total time for searches, and \( O(n^2) \) time for the initialization gives the running time bound of the claim. \( \square \)

We continue with the proof of Theorem 11. Now that the inter-block phase is complete, the fill-block phase is done as follows: for any timeslot \( j \) and for each block colored \( j \mod 4 \), we pick a cell \( v \) of \( H \) in that block which is not reached but whose parent in \( T_H \) is reached. Then we add to \( X_j \) the arc from the parent of \( v \) to \( v \), resulting in \( v \) being reached. If no such cell \( v \) exists, then we stop the algorithm. As there are at most \((\sqrt{2}^2 + 1)\) cells in a block, the fill-block phase uses at most \( 4((\sqrt{2} + 1))^2 \) timeslots. Finally, every cell \( v \) is reached, since, if \( x \) is the closest ancestor of \( v \) in \( T_H \) with \( x \in V(T') \), then \( x \) is reached in the inter-block phase, and the path from \( x \) to \( v \) in \( T_H \), including \( x \) and \( v \), is contained entirely in one block (this is since, for any vertex \( x \) of \( T_H \) but not \( T' \), \( x \)'s parent in \( T_H \) is in the same block as \( x \)).

By putting together the number of timeslots of the fill-block phase, and using Claims 12, 13, and 14, we finish the proof of Theorem 11. \( \blacksquare \)

At the expense of slightly complicating the proof, if one uses a hexagonal grid instead of a square grid, we believe that the upper bound in the theorem above can be slightly improved to \( 48 \sqrt{3}((\alpha + 4))D + 12((\alpha + 4)^2) + 1 \).

**Theorem 15.** There exists a centralized algorithm with running time \( O(n^2) \) that gives an \( O(\alpha D) \) interference-free broadcast schedule for any Euclidean instance of IABS.

**Proof.** Another algorithm is to construct the UDG \( G \) and \( H' \) as above, but treat \( H' \) as one single block and fill its cells one by one. \( H' \) can have at most \( \pi(D + 1)^2/(1/2) \) cells, since each cell of \( H' \) has area \( 1/2 \) and diameter \( 1 \), which implies it is completely included in the disk rooted at \( s \) with Euclidean radius \( D + 1 \). Thus we can find a conflict-free broadcast schedule for \( H' \) with
2π(D + 1)^2 timeslots and, as in Claim 12, an interference-free broadcast schedule for I with 4π(D + 1)^2 + 1 timeslots.

Now we just balance the bound above with the one from Theorem 11. If D < 10α, we use the bound above, giving us a schedule with at most 4π(10α + 1)(D + 1) + 1 timeslots. Otherwise, if D ≥ 10α, we use Theorem 11 to get an interference-free broadcast schedule with 64(2 + ⌈√2(α + 3)⌉)D + 8(⌈√2(α + 3)⌉)(⌈√2(D/10 + 3)⌉) + 1 timeslots. In both cases, the existence of an O(αD) interference-free broadcast schedule follows.

Recall that any interference-aware broadcast schedule has latency at least D. Then an immediate consequence of the theorem above is:

**Corollary 16.** There is an O(α)-approximation algorithm for INTERFERENCE-AWARE and INTERFERENCE-FREE BROADCAST SCHEDULING on Euclidean instances.

### 7. Conclusion

We formulated the NP-Hard BROADCAST SCHEDULING problems with interference-aware or interference-free transmissions as integer programs and are able to optimally solve moderate-size random instances of the problem.

Then, we presented six variations of heuristics based on availability of and maximum marginal receivers gained by simultaneously transmitting nodes. Note that our heuristics do not need any pre-processing of instances. The experimental results show that the greedy heuristics based on maximum marginal receivers gained by simultaneously transmitting nodes produce outputs closest to optimum solutions, for uniform random instances. In addition, the average solutions from the best heuristics in each category exceeds the average of the optimum solutions by only 13-20%. Our algorithms may very well perform no worse on real-world instances.

Finally, we show that an O(αD) schedule can be computed centralized in O(n^2), thus giving an O(α)-approximation algorithm. We leave open the issue of whether an O(1) approximation is achievable in the Euclidean model.

### References


