



Article

Parallel Algorithm for NP-Hard Problem of Channel Resource Allocation Optimization in Ad Hoc and Sensor Networks

Valeriy Ivanov * and Maxim Tereshonok

Science and Research Department, Moscow Technical University of Communications and Informatics,
111024 Moscow, Russia; m.v.tereshonok@mtuci.ru

* Correspondence: v.i.ivanov@mtuci.ru

Abstract

This paper proposes a technique to estimate the minimal quantity of orthogonal channel resources required for ad hoc and sensor network connectivity. Simultaneously, the resource allocation to each specific line is conducted by grouping lines into concurrent transmission sets. Our proposed technique uses the physical-based interference model assumption, where each node interferes with every other node, turning ad hoc and sensor network performance optimization problems into the NP-hard ones. In contrast to most of the other works with the physical-based interference model assumption, we mitigate the combinatorial explosion of concurrently transmitting line sets considering the global interference instead of localizing the interference with line or space partitioning. We have performed the mitigation, firstly, using pairwise mutually acceptable line sets for each line. Then, based on the limitations of pairwise sets, we construct the tree of mutually acceptable interfering line sets. Then, from the created tree, we find the minimal set cover of concurrently transmitting line sets. The tree construction has the exponential worst-case time and space complexity if all lines in the network can transmit together. By randomly pruning the tree and using the genetic algorithm to find the pruned tree which gives the same minimal set cover as the full tree, we have reduced the worst space and time complexities to the polynomial ones. We have devised our technique with parallelism in mind to speed up the resource allocation optimization even more. Based on an extensive simulation study with random network topologies of sizes up to 250 nodes and the average number of lines up to 2000, we estimated the time and space complexity for different tree pruning and optimization techniques and found the effective ones.

Keywords: 6G; ad hoc network; artificial intelligence; channel resource allocation; concurrent resource usage; global heuristic optimization; mutual interference; NOMA; parallel distributed algorithm; sensor network



Academic Editors: Pranav Gangwani
and Alexander Perez-Pons

Received: 27 June 2025

Revised: 24 July 2025

Accepted: 5 August 2025

Published: 8 August 2025

Citation: Ivanov, V.; Tereshonok, M. Parallel Algorithm for NP-Hard Problem of Channel Resource Allocation Optimization in Ad Hoc and Sensor Networks. *Future Internet* **2025**, *17*, 362. <https://doi.org/10.3390/fi17080362>

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

An ad hoc network is a self-organizing network of mobile radio nodes without fixed infrastructure [1]. A sensor network is a type of ad hoc network with gateway nodes for telemetry output [2].

The problem of estimating the minimal quantity of orthogonal channel resources required for the network connectivity is essential for ad hoc and sensor networks [3]. Another connected problem is to decide which resource to allocate to each specific line.

The aforementioned connected problems come down to the optimal resource allocation problem for network capacity maximization [3]. In our case, the maximization of capacity

is performed with the search of line sets which can transmit simultaneously using the same specific orthogonal resource. Maximal throughput can be achieved if all lines transmit simultaneously using one orthogonal resource.

Network capacity (throughput, unit of information per unit of time) is the essence of any network and the ultimate target of performance optimization [1]. The research on ad hoc network capacity can be divided into two parts: the estimation of achievable capacity, and the ways to achieve maximal theoretical capacity. There are three main types of network capacity [1]:

1. The sum of all active lines' successful reception rates (the case of transmission between two neighbor nodes).
2. The sum of all active routes' successful reception rates (the case of multi-hop transmission between non-neighbor nodes).
3. The sum of all active route trees' successful reception rates (the case of multicast transmission from one node to multiple nodes).

The throughput of routes depends on line throughput. Therefore, great research attention is paid to the line throughput (the first type mentioned above).

Several works have proposed achievable capacity estimations for the following: common case of ad hoc networks [1,4–6]; multicast data flows in ad hoc networks [7]; mobile ad hoc networks [8], where the capacity dependence on the network mobility was demonstrated; ad hoc networks with directional antennas [9,10]; ad hoc networks with spatial diversity using multiple input and output antennas (MIMO) [11,12]; ad hoc networks with infrastructure nodes [13]; capacity–delay tradeoffs in delay-tolerant ad hoc networks [14–17]; multipacket reception using network coding [18–20].

Works that search for ways to achieve theoretically maximal capacity are usually based either on heuristic optimization of randomly allocated resources [21] or on simplified cases or approximation where resource allocation can be expressed as a convex optimization problem [22].

Usually, the capacity maximization parameters are chosen beforehand, but the work [21] utilizes the elephant herding lion optimizer (EHLO) algorithm to pinpoint critical attributes of a vehicular ad hoc network to use for resource allocation. Afterwards, for scalability, the network is divided into clusters with the multiple fuzzy c-means (MFCM) algorithm. Subsequently, the crossover and mutation (CM)-centered marine predator algorithm (MPA) is used for resource allocation optimization.

Some works incorporate routing into resource optimization, because routes define active lines. For example, the work [22] proposes a cross-layer routing and radio resource allocation algorithm in the case of Massive MIMO and orthogonal frequency division multiple access (OFDMA) wireless ad hoc networks. The proposed algorithm simultaneously finds data routes, sub-carrier schedules, and power allocations that maximize network throughput by converting a non-convex resource allocation optimization problem into a convex one.

Channel resources can be allocated to more than data transmission. The paper [23] proposes an algorithm to allocate spectrum and power between positioning and data transmission. MIMO systems allow us to estimate the signal's angle of arrival. With multiple nodes, it is possible to estimate the location of each node in an ad hoc network. Location-based routing and resource allocation allow for higher network throughput, but at the same time, some part of network throughput is spent on node localization. This work considers the tradeoff between positioning and data transmission. The network is separated into clusters based on nodes' direct line-of-sight probability. Afterwards, the allocation of power and spectrum between localization and data transmission is chosen for clusters using the Generalized Benders Decomposition (GBD) algorithm. The work [24]

proposes the algorithm for time slot allocation between data transmission and channel sensing. Good channel state estimates can provide higher line throughput. The work [25] deals with frequency-selective channel striving to simultaneously maximize the mean network SNR and spectrum efficiency.

For flying ad hoc networks, numerous algorithms of simultaneous optimization of trajectory and network throughput are proposed. These algorithms allocate some bandwidth for trajectory negotiation [26].

In delay-tolerant ad hoc networks, channel resources can be traded for storage resources. The work [27] considers the transmission resource and the storage resource allocation as a queuing process. This work derives the expressions for per-node throughput, expected end-to-end delay, and network throughput. The analysis of derived expressions gives the optimal setting for resource allocation.

Substantial number of works employ artificial intelligence (AI) for resource allocation [28]. The work [29] uses the self-organizing feature-mapping model to process multi-dimensional data of the network state to perform multi-associated parameter aggregation-based routing and resource allocation. The work [30] uses the deep neural network attention mechanism and Q-learning for the resource allocation in the smart city hierarchical edge computing network. The work [31] uses dynamic unsupervised fuzzy clustering to find all resource combinations for the different network services. The work [32] uses recurrent neural networks for joint scheduling and power control. The work [33] proposed a graph kernel-based support vector machines method for the link scheduling. Many works formulate the resource allocation problem as a maximum weighted independent set problem [34] and solve it with AI approaches. The work [35] uses deep learning and fuzzy logic, the work [36]—uses a tree-based graph convolutional network, and the work [37]—uses a tree-based graph convolutional network with deep Q-learning.

The works [38,39] have analyzed the following problem: given the set of lines and their powers, what lines are allowed to transmit to reach the maximal sum of the lines' throughput. The work [38] showed that if all line transmission powers are lower than critical powers for each line, then the sum of line throughputs (network capacity) is larger than the throughput of one line from the set. If the power is larger than the critical power, then the throughput of one line will be larger than the sum, because of the interference, so only one line at a time should be allowed to transmit. The work [39] has expanded the work [38] and proved that subsets of multiple transmitting lines can maximize the sum of line throughput. However, the authors have not shown how to search for acceptable line subsets.

Many works assume a binary or graph interference model [35], where specific pairs of wireless nodes interfere with each other or not. In contrast, the physical-based interference model assumes that all nodes emitting radio waves interfere with each other. The physical based interference model is based on signal power and propagation decay [40–42]. The physical-based model can reflect reality better than the graph-based model. However, global line interaction in the physical-based model turns resource allocation into an NP-hard problem [40]. By partitioning the network area into subspaces or by partitioning the lines into subsets, global interference becomes localized. The work [43] partitions line set into two sets: the lines that are close to the maximum transmission range of senders and other lines. The work [44] partitions the network area into large squares and then partitions each large square into small squares. Next, the work utilizes partition and shifting strategies to find multiple sets of geometrically well-separated links. The work [45] uses the hypergraph approach to partition links into sets. The work [46] is based on the fact that a short link can tolerate large interference while a long link tolerates small interference. The work partitions all links into disjoint local link sets that are spatially separated from

each other. The work [41] partitions the network area into hexagons and shows that hexagon partitions achieve better network throughput than the square ones. The work [47] compared square, triangular, and hexagon partitions and proved that the hexagon partition gives the highest network throughput.

Our proposed solution employs the physical interference-based model, but, in contrast to the most of the other physical-based model algorithms, we have tried to mitigate the combinatorial explosion instead of using line or space partitioning. We have performed the mitigation, firstly, using pairwise mutually acceptable line sets for each line. Then we construct the tree of mutually acceptable interfering line sets. New leaf nodes are chosen from the ever decreasing set intersection of the pairwise mutually acceptable line sets of the lines in the path from the parent node to the root. Then we apply a minimal set cover algorithm to find the minimal number of concurrently transmitting line sets. Tree construction has exponential worst-case time and space complexity if all lines in the network can transmit together. By randomly pruning the tree and using the genetic algorithm to find the pruned tree which gives the same minimal set cover as the full tree, we have reduced the worst space and time complexities to the polynomial ones. We have devised our solution with parallelism in mind to speed up the resource allocation even more.

Our solution offers the following key features:

- Each main step of the proposed solution is parallelizable. It enables large-scale distributed computation for large ad hoc networks.
- Searching for acceptable interfering line sets leads to a combinatorial explosion. To overcome this problem, the proposed solution creates acceptable line sets. These line sets are created using tree construction with pairwise limitations and pruning.
- The proposed solution employs a centralized approach to optimize network capacity without clusterization or partitioning in contrast to most approaches for large ad hoc networks [48–50].
- The proposed solution guarantees fairness; some channel resource will be given to each active line.
- The solution guarantees a bit error rate lower than the requested threshold for each line.
- The optimization is conducted in a natural complex search space, rather than in the simplified convex alternative.
- Parallel transmission with one orthogonal resource includes the case of NOMA. The proposed algorithm can be used to optimize NOMA groups in 6G cellular networks and ad hoc networks. Application of our solution to NOMA can be realized as follows: All outgoing lines of every node can be grouped and replaced with one or multiple lines to the most distant recipient node inside the group. Those lines will be NOMA encoded, containing the signals for all lines in the group. As a result, we replace multiple outgoing lines with fewer NOMA lines. Grouping lines into NOMA lines into various combinations and using our proposed solution, we can find the set of NOMA lines which requires the minimal number of orthogonal channel resources.
- The proposed solution makes real-time decentralized resource allocation possible using machine learning and artificial intelligence. Let us elaborate on the machine learning application. Generating a large number of random topologies, we combine lines into groups that can use the same channel resource, obtaining the training set. The number of lines cannot be larger than the squared number of nodes, and the number of groups cannot be larger than the number of lines. For example, we can train the following neural network with the following inputs and outputs. Inputs: all node coordinates; vector of transmission powers; each node's receiver noise plus interference. Outputs: multiple binary vectors, each of them representing the group of

lines with the same orthogonal resource. The inference of the neural network trained with the results of our solution can be much faster than using our solution directly.

- The proposed solution provides capacity estimation for arbitrary network configurations and physical layer specifications.

2. Materials and Methods

2.1. Problem Statement and Solution Overview

Given:

- Ad hoc or sensor network topology (node coordinates; line properties between nodes) and the set of active lines. The set of active lines can consist of either all lines or some subset of lines, which can be based, for example, on routes between nodes exchanging information or a network minimal spanning tree.
- Each line of the ad hoc or sensor network uses one equal (in terms of bandwidth) orthogonal channel resource. The examples are as follows: time slots of equal length, frequency channels of equal bandwidth, etc.
- Each line can use a different type of signal. For example, given a specific frequency band of the same width, one line can use QAM-16 modulation, and the other line can use QPSK modulation. The type of signal defines the bit error curve.
- Each line has a power budget for other lines' interference. If there is no budget for other lines' interference, then only one line can transmit using one specific orthogonal channel resource. In this case, no simultaneous use of shared channel resource is allowed.
- Maximal acceptable bit error rate for each line.
- Each line has its own transmission power to satisfy the maximal bit error rate requirement under the given signal propagation loss, receiver thermal noise power, and power budget for interference from other lines. Lines with different distances between the transmitter and receiver have different transmission powers, hence interference to other lines.

Find:

- The minimal quantity of orthogonal channel resources to allow active lines transmission with a bit error rate below the allowed threshold.
- The resource allocation for each specific line.

The proposed solution consists of two phases.

First phase: Find line sets with mutual interference within given bounds (mutually acceptable interfering line sets).

Second phase: Find the minimal set cover of the mutually acceptable interfering line sets.

We set the definitions to describe the algorithm to find mutually acceptable interfering line sets in the table (Table 1).

The ideas of mutually acceptable interfering line sets and pairwise mutually acceptable lines are illustrated in Figures 1 and 2. Nodes are depicted with circles; lines between nodes represent communications lines; multicolored bars associated with adjacent lines represent interference budgets of these lines. Interference budget bar colors are the same as the colors of lines that produce this interference.

Table 1. The definitions to describe the algorithm to find mutually acceptable interfering line sets.

Symbols	Descriptions
T	A tree representing sets of lines with mutual interference lower than acceptable threshold.
L	All lines set.
μ	An empty line.
n	The node of set tree of layer.
$n \in T$	
$n = \{l, n_{previous}\}$	$l \in L$ is the line of node; $n_{previous}$ is the previous node.
$v = \{\mu, v\}$	The line set tree root node; the only node which contains an empty line μ .
$l(n)$	A line of node n .
A_l	The pairwise mutually acceptable line set for the line $l \in L$. l can transmit simultaneously with any single line from A_l : any pair of lines $\{l, \forall x \in A_l\}$ can use the same orthogonal channel resource.
$A(l)$	A function (algorithm) to obtain the pairwise mutually acceptable line set.
$A(\mu) \rightarrow L$	
$A(l \neq \mu) \rightarrow A_l \subseteq \{L \setminus l\}$	
D_n	A set of the lines taken from the set of nodes constituting the path to the tree root from node n .
$D_v = \{\mu\}$	
$D_{n \neq v} = \{l(n), \dots, \mu\}$	
C_n	Candidate successor set of lines for the node n .
$C(n)$	A function (algorithm) to obtain C_n .
$C(v) \rightarrow L$	
$C(n \neq v) \rightarrow C_n \subseteq \{L \setminus D_n\}$	
S_n	Successor set of lines for the node n .
$S(n)$	Function (algorithm) to obtain S_n .
$S(v) \rightarrow L$	
$S(n \neq v) \rightarrow S_n \subseteq C_n$	
$P_{n,l} = \{l \cup D_n\}$	Set union of line l and lines in the path to the root node from the node n .
$I_{n,l}$	Boolean value, if true: lines in the line set $P_{n,l}$ mutually interfere with each other within given bounds; if false: lines in this line set cannot use the same orthogonal channel resource.
$I(P_{n,l})$	A function (algorithm) to find $I_{n,l}$.

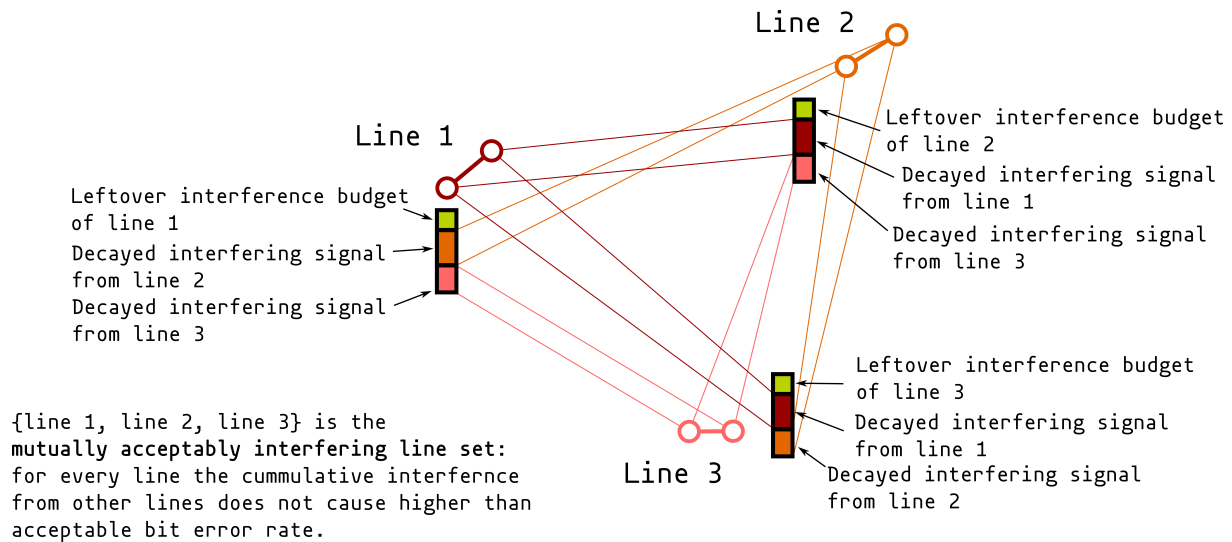


Figure 1. Mutually acceptable interfering line sets illustration.

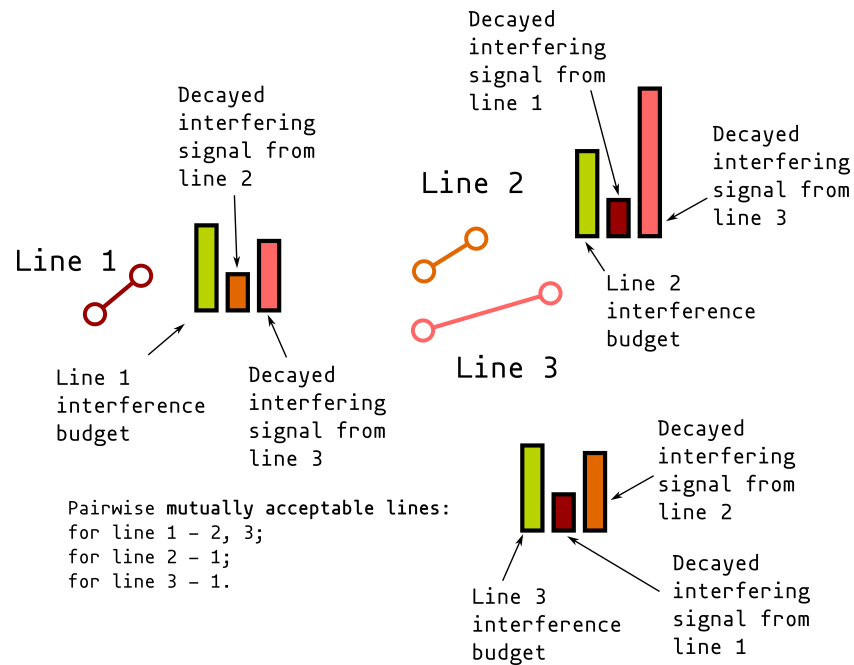


Figure 2. Pairwise mutually acceptable lines illustration.

As can be seen in Figures 1 and 2, the mutually acceptable interfering line sets can be constructed sequentially, using only pairwise mutually acceptable interfering lines. We search for mutually acceptably interfering line sets using the tree construction technique (Figure 3).

We construct the tree using a breadth-first approach:

1. $\forall l \in L$ find A_l with $A(l)$.
2. Create root node of tree with empty line μ .
3. Keep creating the next layer of the tree until the next layer is empty.
4. For each leaf node f obtain the line set $\{D_f \setminus \mu\}$; this line set represents the set of lines with mutual interference lower than the acceptable threshold. Add line set to the set of line sets.

The set of nodes of the same depth we call a layer. The way to create the next layer for each node n of the current layer is as follows:

1. Find C_n using $C(n)$.

2. Find S_n using $S(n)$.
3. Create nodes based on S_n and add them to the tree as children nodes of node n .

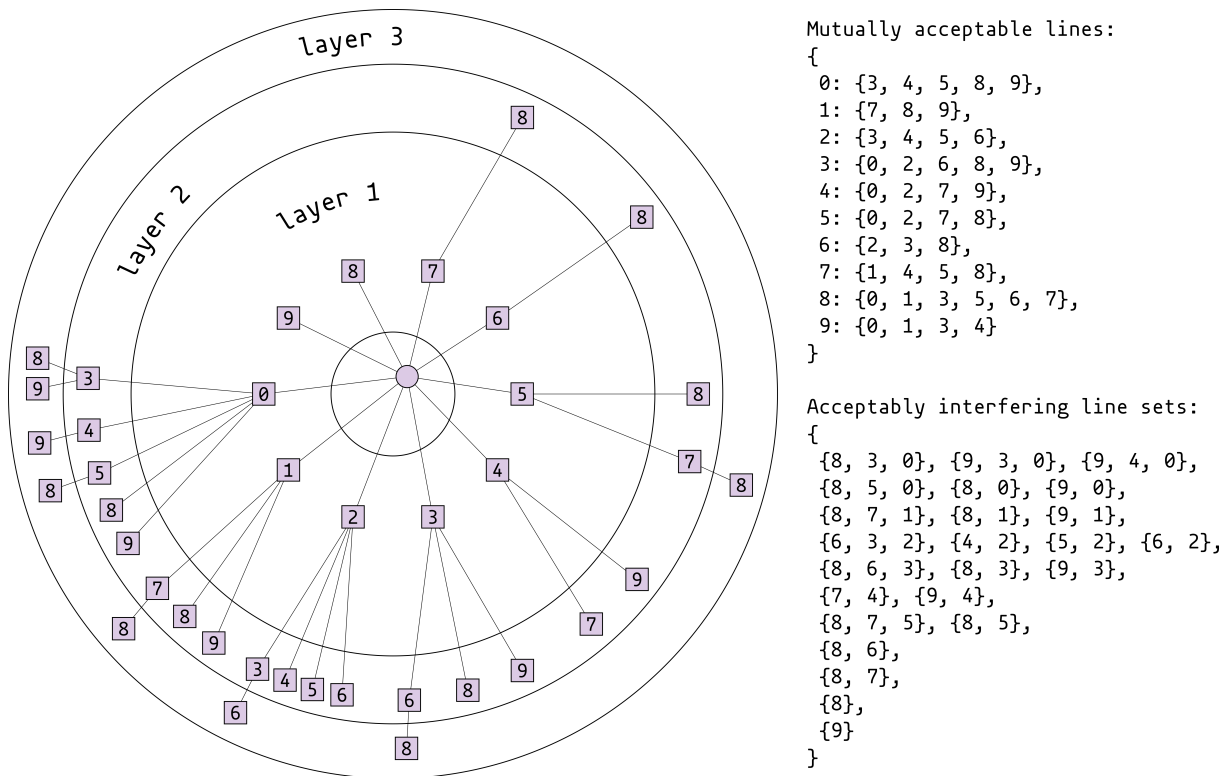


Figure 3. Mutually acceptable interfering line sets search.

Let us illustrate the example of constructing the tree of mutually acceptable interfering line sets (Figure 3). First, we start with the root node with an empty line. Then we create layer 1 nodes from the successor set which consists of all of the lines (lines with identifiers from 0 to 9). Nodes with lines 8 and 9 of layer 1 have no lines in their mutually acceptable line sets A_8 and A_9 with identifiers higher than 8 and 9, correspondingly, so they have no children (we will explain later why we create children nodes with higher value identifiers than the line identifier of the parent). A_0 has lines $\{3, 4, 5, 8, 9\}$ the identifiers of which are higher; so, assuming function $I()$ always returns true, we obtain five children for node 0 of layer 1. The node with line 3 of layer 2 (which is a child of the node with line 0 of layer 1) has two children with lines 8 and 9. Lines of those children are elements from the set intersection $\{A_0 \cap A_3\}$ whose identifiers are higher than 3. Traversing from the leaf node, we obtain the set of mutually acceptable interfering lines, for example, line set $\{8, 3, 0\}$.

Now, let us define the functions to find mutually acceptable lines $A()$, to find candidate sets $C()$, and to find successor sets $S()$.

2.2. Function to Find Mutually Acceptable Lines

Let us show how to find mutually acceptable line sets (function $A()$). There are three main limitations for other lines that can transmit together with the given line:

1. Transceiver simplex (or half-duplex) operation mode.
2. Spatial orthogonality by means of beamforming with the help of multiple antennas and synchronized transceivers connected to each antenna.
3. Interference from other lines.

In practice, one transceiver cannot transmit and receive simultaneously being connected to one antenna. Using different antennas for the receiver and transmitter requires spatial separation which is usually impractical for portable devices in ad hoc and sensor networks. As a result, transceivers in ad hoc and sensor networks operate in simplex mode. For the connection between two neighbor nodes it will mean half-duplex line. Simplex mode means that when a source node transmits to one of its neighbor nodes, it cannot receive data from other nodes, and vice versa. In terms of incoming and outgoing lines: if the node transmits, all incoming lines are forbidden; if the node receives, all outgoing lines are forbidden.

Let us use the following definitions to express the sets of allowed lines based on simplex operating mode:

- α —source node.
- N_α —neighbor nodes of node α , the nodes to which the lines exist.
- $n_\alpha \in N_\alpha$ —one of the neighbor nodes of the node α .
- (α, n_α) —line from the node α to one of its neighbors n_α .
- $A_{(\alpha, n_\alpha)}^{simplex}$ —sets of allowed lines based on the simplex operating mode for line (α, n_α) .

$$A_{(\alpha, n_\alpha)}^{simplex} = L \setminus \{(\alpha, n_\alpha)\} \cup \{(n, \alpha) : n \in N_\alpha\} \cup \{(n_\alpha, n) : n \in N_{n_\alpha}\}. \quad (1)$$

Digital beamforming gives us spatial orthogonality (Figure 4). It allows us to transmit simultaneously to multiple nodes or to receive simultaneously from multiple nodes.

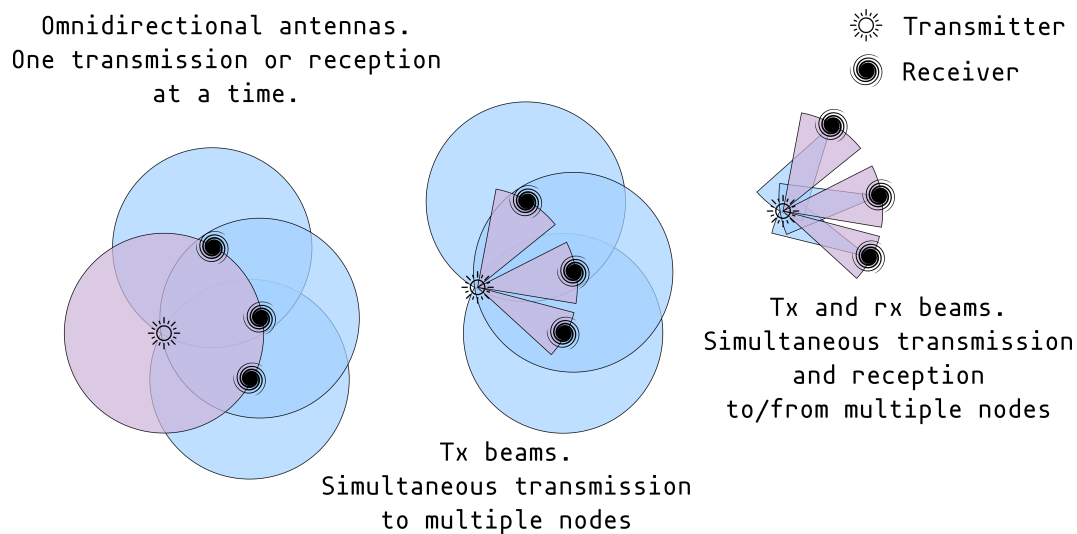


Figure 4. Spatial orthogonality.

There are three main cases of spatial orthogonality (Figure 4):

1. No spatial separation. Transmission and reception utilize omnidirectional antennas. We can send data only to one neighbor node. The receiving node can only receive data from the source node.
2. Beamforming for transmission and omnidirectional antenna for reception. We can send data simultaneously to multiple neighbor nodes as long as they are far enough to be outside one transmission beam, i.e., we can activate multiple outgoing lines. However, receiving neighbor nodes can only receive data from one sending node.
3. Beamforming for transmission and reception. This case extends the previous one: receiving neighbor nodes can receive data from other nodes too.

Allowed line sets based on spatial orthogonality for omnidirectional antennas for transmission and reception:

$$A_{(\alpha, n_\alpha)}^{spatial} = L \setminus \{ \{(\alpha, n) : n \in N_\alpha\} \cup \{(n, n_\alpha) : n \in N_{n_\alpha}\} \}. \quad (2)$$

Allowed line sets based on spatial orthogonality for the transmission beamforming only:

$$A_{(\alpha, n_\alpha)}^{spatial} = L \setminus \{(n, n_\alpha) : n \in N_{n_\alpha}\}. \quad (3)$$

Allowed line sets based on spatial orthogonality for the transmission and reception beamforming:

$$A_{(\alpha, n_\alpha)}^{spatial} = L \setminus \{\emptyset\}. \quad (4)$$

Interference limitation is the key one: lines cannot “coexist” if their mutual interference does not allow the required bit error rate for each line. To obtain the set of allowed lines $A^{interference}$ based on interference, we need to test the given line under the interference of each other line separately and estimate the bit error rate. If the bit error rate is less than the target one for the line, then the interfering line belongs to the set $A^{interference}$.

One of the most comprehensive ways to estimate the bit error rate for multiple mutually interfering lines is to use baseband equivalent simulation for the signals, propagation effects, and transceiver digital signal processing (DSP) pipelines [51]. However, this approach is computationally intensive.

Additive white Gaussian noise (AWGN) interference is the simplest assumption and does not require modeling the intrinsics of transceiver DSP stages. One of the main requirements is signal statistical independence. Line signals usually can be considered statistically independent ones due to random transmitted information, random signal carrier phase offset, random carrier frequency fluctuation, random channel propagation conditions, random clock drifts in the case of time division multiple access, etc. Having found the SINR for the line under the AWGN assumption, we can calculate the bit error rate based on the known formulas for various signal types.

We can derive the formula which depends only on distances between nodes and signal-to-thermal-noise ratio (SNR) for the signal-to-interference-and-noise ratio (SINR) for every line in a network:

$$SINR = 1 / ((1/SNR) + (1/SIR)) = 1 / ((1/SNR) + \sum (D_{interfering\ line} / D_{interfering\ tx\ to\ rx})), \quad (5)$$

using the following assumptions:

- AWGN independent interference.
- Free space propagation signal attenuation.
- Orthogonal channel resources are time slots, close frequency channels, with which we can neglect the free space propagation attenuation difference between different frequencies, set of mutually orthogonal code sequences using time or close frequencies, or both, and unique permutations of equal size of the aforementioned orthogonal resources.
- Every transceiver has the same uniform omnidirectional antenna with isotropic gain equal to one.
- Every line has an equal target SNR, and every transceiver uses power control to reach that target.

If the target SNR is equal to the minimal necessary to reach the target bit error rate, we cannot have any interference from other lines, and we have to assign different orthogonal resources for each line. If we have some line budget, we can allow line interference.

Let us elaborate on the distances in the equation with the help of Figure 5. We have the line from the node 1 transmitter to the node 2 receiver. Active lines between nodes 5 and 6 and nodes 3 and 4 are causing interference to the receiver of node 2. The inverse signal-to-interference ratio (SIR) for the line between nodes 1 and 2 is equal to the following:

$$\frac{1}{SIR_{line\ 1 \rightarrow 2}} = \frac{distance_{5 \rightarrow 6}}{distance_{5 \rightarrow 2}} + \frac{distance_{3 \rightarrow 4}}{distance_{3 \rightarrow 2}}. \quad (6)$$

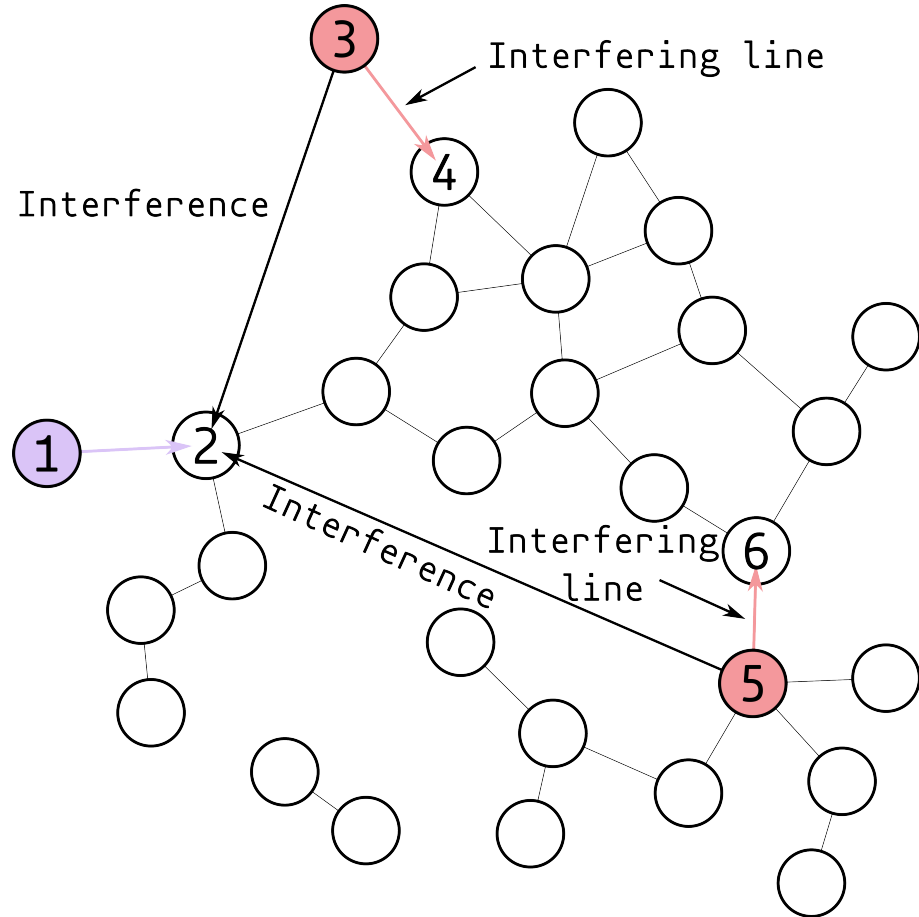


Figure 5. Line interference in ad hoc network.

We now have all the components to compute the mutually acceptable lines. The set of unilaterally acceptable lines is as follows:

$$A_l^{unilateral} = \{A_l^{simplex} \cap A_l^{spatial} \cap A_l^{interference}\}. \quad (7)$$

And the mutually acceptable lines for the line l are as follows:

$$A_l = \{x : x \neq l \wedge x \in A_l^{unilateral} \wedge l \in A_x^{unilateral}\}. \quad (8)$$

2.3. Function to Find the Candidate Sets

Let us sequentially assign the identifier to each line as a rising natural number sequence; $id(l)$ is the identifier of line l .

We are constructing the tree of mutually acceptable interfering line sets (Figure 3). The candidate set is based on the current node n and all the previous ones up to the tree root:

$$C_n = \{x : (x \in \bigcap_{z \in D_n} A_z) \wedge (id(x) > id(l(n)))\}. \quad (9)$$

Essentially, we are taking the lines that do not mutually interfere with every line in the set D_n of the current node n . Then we take the lines whose identifiers are higher than the node n line.

Getting the candidate set in this way, we make each set in the tree of mutually acceptable interfering line sets unique. We still explore all of the line sets: the line with the highest identifier has an empty candidate set, but the lines with lower identifiers will possibly have the line with the highest identifier in their sets.

2.4. Function to Find the Successor Sets

The successor set S_n is the subset of the candidate set C_n . We do the following to obtain the successor set. For every line $l \in C_n$ find $I_{n,l}$ using the function $I()$. If $I_{n,l} = true$, add line l to the S_n .

Function $I()$ is the function to check for each line in the set if its bit error rate is lower than the target one under interference from other lines. The bit error rate can be estimated with baseband simulation of signals, their propagation and DSP at transceivers, or, under the assumption of AWGN, we can sum up the powers of other interfering lines and obtain the bit error rates with the known formulas for many types of signals.

2.5. Minimal Set Cover Search

Having constructed the tree of mutually acceptably interfering line sets, we take all the line sets and search for the minimal number of sets which cover every line to find the minimal quantity of orthogonal resources. It turns out to be a minimal set cover problem, which is one of Karp's 21 NP-complete problems [52]. We have chosen the Caprara, Fischetti, and Toth algorithm (CFT) to solve the minimal set cover problem [53]. It is a sequential algorithm, but many parallel algorithms can be adopted for this problem [54–57].

2.6. Random Tree Pruning Based Search

With a large number of lines, we will obtain the combinatorial explosion of paths in the tree of mutually acceptable interfering line sets. To mitigate that, we can randomly prune the tree, by not choosing more than the given amount of nodes in the current layer. We need to set the maximal number of nodes to not be less than the number of lines, so that the first layer has all lines, and we are guaranteed to have the sets covering all lines. The second reason to make the maximum number of nodes equal to the number of lines owes to the fact that the number of sets in the minimum set cover over the full tree is always equal to or less than the number of lines. This means that the pruned tree needs to accommodate the number of leaf nodes to be more than or equal to the number of lines to guarantee the possibility of finding the min set cover over the pruned tree equal to the one over the full tree.

The higher the maximum number of nodes per layer, the higher the probability of randomly creating the pruned tree which gives the same minimal set cover as the full tree. Let us define the pruning scale factor which can be any natural number starting from one. If the pruning scale factor equals one, then the maximum number of nodes per layer is equal to the number of lines. The pruning scale factor cannot be less than one.

So, instead of using the full tree for finding min set cover, we need to find the pruned tree which gives the same min set cover. We can use the pruning information to represent the pruned tree. With the optimization of pruning information, we can find the best tree. We represent the pruning information for each layer with two elements: the number of layer nodes before pruning, and the set of indices chosen to be left after pruning.

To prune, we first find all nodes of the next layer. Then we randomly sample the nodes with one of the following distributions of choice: uniform, linear, or Pareto. Before sampling, we collect nodes of the next layer into the next layer node list and sort nodes by

the identifier of their line in increasing order. To sample only unique nodes and to decrease the number of sampling operations if most of the nodes were chosen, we do the following: if the node was already chosen, we traverse to the end of the next layer node list until we find a not chosen node. If we reach the end of the list without finding an eligible node, we traverse backwards from the first chosen node to the start of the list. If the number of nodes in the layer is less than the maximum allowed number of nodes, then we take them all without sampling.

With the uniform distribution, we sample nodes with equal probability. With linear and Pareto distributions, we choose nodes having lower line identifiers with higher probability than nodes having higher line identifiers. This is realized because nodes with lower identifiers can have more children with higher probability, hence larger concurrently transmitting line sets.

We use the scale parameter for the linear and Pareto distribution, which shows how many times the probability of choosing the first node in the list having the lowest line identifier is higher than choosing the last node having the highest line identifier. For the Pareto distribution, we set the parameter $\beta = 1$. And the parameter α is variable.

Thus, we can generate randomly pruned trees. However, pruned trees do not always contain all the line sets of the smallest minimum set cover, unlike the complete tree. Thus, we need to perform the heuristic optimization to find the pruned tree which gives us the smallest minimum set cover.

We will use two ways to perform optimization: random search and genetic algorithm (GA) search.

Both approaches have the following common parameters: batch size (for the GA it is the population size), minimal number of iterations and the number of iterations after the best found solution has not improved. It is floating and represented in the ratios of the number of iterations that have happened. For example, if the number of iterations after the best found solution has not improved is 2 times larger than the number of iterations when improvement has happened. Consequently, if improvement happened at the 30th iteration, we stop at the 60th iteration if no new improvement has happened. If new improvement has happened after the 50th iteration, we slide the last iteration forward to the 100th iteration. The GA has one additional parameter: mutation probability.

A single iteration of the random search is described as the following pseudocode (Algorithm 1):

Algorithm 1: Algorithm of a single iteration of the random search

- 1: Create randomly pruned trees batch.
 - 2: Find the minimal set cover for each tree.
 - 3: Find the tree with the smallest minimal set cover.
 - 4: **if** (the size of smallest minimal cover < the size of best solution minimal cover)
 then
 - 5: Update the best solution.
 - 6: Update the iteration when the best solution has improved.
 - 7: **endif**
-

The stopping criterion for the random search is met when two conditions are fulfilled: the minimal number of iterations has happened and the number of iterations after the best found solution has not improved.

The crossover and mutation operators for the GA are the following. For each layer of the tree, we record the set of chosen node indices after pruning for the layer is finished. The

crossover performs the pruning information exchange and is described as the following pseudocode (Algorithm 2):

Algorithm 2: Algorithm of the pruning information exchange

- 1: Find the deepest equal layer between the two solutions (trees).
 - 2: {First solution unique indices set} $\leftarrow$ {first solution indices set} $\setminus$ {second solution indices set}.
 - 3: {Second solution unique indices set} $\leftarrow$ {second solution indices set} $\setminus$ {first solution indices set}.
 - 4: {Common between the two solutions indices set} $\leftarrow$ {first solution indices set} $\cap$ {first solution indices set}.
 - 5: Crossover point $\leftarrow$ uniformly randomly chosen integer number between 1 and the size of first solution unique indices set.
 - 6: Swap parts of unique indices sets for the two solutions between 1 and crossover point inclusively.
 - 7: {First solution indices set} $\leftarrow$ {common between the two solutions indices set} $\cup$ {first solution unique indices set}.
 - 8: {Second solution indices set} $\leftarrow$ {common between the two solutions indices set} $\cup$ {second solution unique indices set}.
 - 9: Discard all the layers after the deepest equal layer for the first solution.
 - 10: Discard all the layers after the deepest equal layer for the second solution.
 - 11: Using the new chosen sets for pruning, regenerate the next layer after the deepest equal for the first solution.
 - 12: For the first solution, regenerate other layers with random pruning.
 - 13: Using the new chosen sets for pruning, regenerate the next layer after the deepest equal for the second solution.
 - 14: For the second solution, regenerate other layers with random pruning.
-

Mutation discards all of the crossover results and creates new random pruning information. The mutation after crossover is performed as in the following pseudocode (Algorithm 3):

Algorithm 3: Algorithm of the mutation after crossover

- 1: Take the deepest equal layer from the crossover.
 - 2: Discard all the layers after the deepest equal layer.
 - 3: Regenerate the tree after the deepest equal layer with random pruning.
-

Performing crossover and mutation starting from the next layer after the deepest equal layer, we preserve the pruning information for the previous layers. For the starting random population, the deepest equal layer is equal to one. It is the layer with all the lines. With crossover, we can evolve the next layer to obtain trees with smaller minimal set covers. However, the layers after the next one are pruned randomly. We can obtain better results if we sequentially optimize pruning data for each layer by evolving the population multiple times and increasing the deepest equal layer. To do that we use the following cycle of the genetic algorithm (Algorithm 4):

Algorithm 4: Algorithm of the genetic algorithm cycle

```

1: Perform one layer evolution.
2: Take the best solution (the solution with the smallest minimal set cover).
3: if (the number of nodes of the next after the current deepest equal layer  $\leq$  the
   allowed number of nodes after pruning) then
4:     Exit.
5: endif
6: Create the set  $S$  of  $N$  ( $N$  is the population size) empty solutions.
7: for  $i = 0$  to  $N - 2$ 
8:      $S[i] \leftarrow$  create mutated copy of the best solution after the current deepest
       equal layer plus one.
9: endfor
10:  $S[N - 1] \leftarrow$  best solution.
11: goto perform one layer evolution.

```

The iteration of one layer tree evolution is described in Algorithm 5:

Algorithm 5: Algorithm of the iteration of one layer tree evolution

```

1: Randomly split the current population into pairs.
2: Perform the crossover between pairs and the mutation of resulting offsprings.
3: Find the minimal set cover for every tree in the offsprings set.
4: Unite the current population and offspring population into one population.
5: Sort the united population in the rising order of minimal set cover size.
6: Take the best, first, and half of the united population into the next population.
7: Find the tree with the smallest minimal set cover among the new population. It is
   the first tree in the population, as we previously sorted the population.
8: if (the size of smallest minimal cover  $<$  the size of best solution minimal cover)
   then
9:     Update the best solution.
10:    Update the iteration when the best solution has improved.
11: endif

```

The stopping criterion for the one layer tree evolution is the same as the stopping criterion for the random search.

2.7. Complexity Estimation

Let us discuss the space and time complexity estimation for the minimal orthogonal channel resource allocation and concurrently transmitting lines grouping.

The space complexity is determined by the mutually acceptable interfering line sets tree space complexity which is equal to the number of tree nodes. The space complexity for the algorithm without tree pruning is just the node number in one tree. Space complexity for the random search and GA algorithms equals the largest sum of multiple pruned trees' node numbers among multiple iterations.

The time complexity consists of the tree construction time complexity and minimal set cover search algorithm time complexity over that tree.

The time complexity of tree construction is the sum of the successor set creation time complexity for each node. The time complexity of successor set creation for each node is equal to $N(C_n) \cdot (T + 1)$, where T is layer depth and $N(C_n)$ is the size of the candidate set. It is proportional to the size of the successor set multiplied by the time complexity of mutual interference estimation. The time complexity of mutual interference estimation

is proportional to the number of lines in the set, which is equal to the node layer depth plus one.

The tree construction time complexity for the algorithm without tree pruning is equal to the time complexity of one tree creation. The time complexity for the random search with tree pruning is equal to the sum of time complexities for all trees created during all iterations. The time complexity for GA is equal to the random search plus the time taken to regenerate trees after crossover and mutation.

Let us now estimate the worst-case time and space complexities for the search with full tree construction, with pruned tree construction, and for greedy search (Table 2).

Table 2. Worst time and space complexities for the search with full tree construction, pruned tree construction, and greedy search.

Algorithm	Worst Space Complexity	Worst Time Complexity
Full tree	$O(2^N \cdot N)$	$O(2^N \cdot N^2)$
Pruned tree	$O(N^3)$	$O(N^5)$
Greedy search	$O(N^2)$	$O(N^3)$

We can represent the set of lines with a binary sequence of size N , where N is the number of lines, so that $\{1, 0, 0, \dots, 0\}$ is the set with line 1, $\{0, 1, 0, \dots, 0\}$ is the set with line 2, $\{1, 1, 1, \dots, 1\}$ is the set with all lines, and so on. The number of all line sets is the number of permutations for the binary sequence of size N without a full zero sequence (empty line set). This number is equal to $2^N - 1$. The worst case for full tree construction is when all lines can coexist together, when we need to check all of the line sets. So, considering that the size of one set cannot be greater than N , the worst case of space complexity for the construction of the full tree is $O((2^N - 1) \cdot N) \approx O(2^N \cdot N)$. The time complexity of checking mutual interference inside one set cannot be larger than $N \cdot (N - 1)$, so the worst-case time complexity is $O(2^N \cdot N \cdot (N - 1)) \approx O(2^N \cdot N^2)$.

With pruning, the number of sets cannot exceed the product of the maximum number of layers (it is equal to the number of lines N) multiplied by the maximum number of nodes per layer (it is equal to the number of lines multiplied by the scale factor $K \rightarrow N \cdot K$): $N \cdot N \cdot K = N^2 \cdot K$. As stated previously, the size of the set cannot be greater than N , so the worst space complexity cannot exceed $O(N^2 \cdot K \cdot N) \approx O(N^3)$. We have omitted the scale factor K , because it is constant. With pruned trees, we create multiple trees, the quantity of which can be constant too; so for the complexity estimation, the number of iterations can be omitted too. Before pruning, we need to create all the nodes of children in the current layer, the maximal possible number of children is $N - 1$, and the maximum time complexity to estimate sets for mutual interference cannot be greater than $N \cdot (N - 1)$, so the worst time complexity for pruned trees is $O(N^2 \cdot (N - 1) \cdot N \cdot (N - 1)) \approx O(N^5)$.

Let us propose a naive greedy search algorithm for mutually acceptable interfering line sets and estimate its worst space and time complexity. We start with the first line and check sequentially for all other lines if they are interfering acceptably. If, for example, the second line is interfering acceptably, we add it to the line set. Afterwards we check for the set of two lines, if we can add line number 3, and so on. We exclude the found set from the line set. For the remaining line set, beginning from the first line, we repeat the same process until no lines are remaining. The worst case is when all lines can coexist together; we check $N - 1$ line sets with sizes from 2 to N . The worst case time complexity for this case is $O((N - 1) \cdot N \cdot (N - 1)) \approx O(N^3)$. If in the first iteration we obtain the set which does not include all lines, we work with the smaller N values on subsequent iterations which sum up to N . Assuming that $\forall P, Q, N, \sigma \in \mathbb{N} : P + Q = N \implies P^\sigma + Q^\sigma < N^\sigma$, the worst time complexity of greedy search cannot exceed $O(N^3)$. Line set size cannot be

greater than N . The number of line sets in the proposed greedy search also cannot exceed N (the case when all lines can transmit only separately from each other). So, the worst space complexity of greedy search is $O(N^2)$. At the same time, greedy search does not provide optimal or even suboptimal solution in most cases.

Worst-case complexities are presented on the Table 2. As we can see, tree pruning, compared to the full tree, shifts complexity from the exponential to the polynomial one, which is a quite substantial complexity reduction.

2.8. Parallelization Considerations

Parallelization is the availability of independent computations. The following criteria are present for the creation of a tree of mutually acceptable interfering line sets:

- Mutually acceptable line sets calculation is independent for every line.
- For each node, the creation of the candidate set is independent.
- The creation of candidate set for one node is sequential.
- The creation of a successor set is based on each element in the candidate set and is independent.
- To check if the element in the candidate set can be promoted to the successor set, we need to perform mutual interference check for the set of lines. Estimating interference is independent for each line in the set.
- Choosing which nodes to prune in the next layer is sequential.

For the random and GA search, the creation of the initial population and solution estimation is independent for each solution; crossover and mutation of the solution are independent for the pairs of solutions. Sorting the solutions and searching for the best one in the current iteration can be parallel. Each iteration of the random search is parallel, and for the GA each iteration is sequential.

Minimal set cover search can be parallel depending on the algorithm.

3. Results

The results of the proposed algorithm evaluation are described in this section. There are two extreme cases of topologies. The first one is when lines are too separated from each other, so that cumulative interference is negligible irrespective of the number of lines; then all lines can transmit together. The second one is a fully connected network when all lines are close together; then only one line at a time can transmit. The more interesting topology is an intermediate case, when nodes are uniformly distributed in the space, and their transmission ranges only slightly overlap with some closest nodes. Also, there are two extreme cases of SNR values. If the SNR value is chosen only to combat receiver thermal noise, then all lines must transmit separately. If the SNR is chosen to be quite large, we, essentially, neglect the thermal noise, and only the interference from other lines places the limit on the simultaneous line transmission; we obtain the case with the largest number of line sets to choose from. Irrespective of how large the SNR is, relative interferes will be the same, if all lines' SNR is equal, and transmission power is chosen considering the space propagation signal decay to reach exactly the given SNR. For modeling, we have chosen the SNR to be equal to 30 dB (received signal power is 1000 times more than thermal noise power), which is reachable in many wireless communication systems and allows for low bit error rates for many modulation types.

To obtain the intermediate case of topology, we generate the following random network topologies (Figure 6). On the Figure 6, nodes are depicted as red circles, the radius of circle is proportional to the number of lines coming from the node. We use the maximal connection distance approach: if nodes are separated for less than the maximal connection distance, then they have bi-directional lines. To create a random topology, we create randomly

distributed points inside the circle of diameter 1. Firstly, we create the points with the coordinates uniformly distributed in the range from -0.5 to 0.5 and then reject points outside of the circle. We set the maximal connection distance equal to the estimated minimal one until the network is still connected. To do that, we start with the maximal connection distance equal to 1 and, with 30 iterations of dichotomy search, gradually reduce it up until the point where the network splits into the subcomponents.

We have chosen the target bit error rate for the modeling based on the following assumptions. The most widespread packet size in the Internet is 1500 bytes. The packet error rate is quite often required to be equal to 0.01. We can calculate the packet bit error rate using the following equation: $packetBitErrorRate = 1 - (1 - packetErrorRate)^{1/packetSizeInBits}$. The target bit error rate is then equal to the 8.375×10^{-7} . Using QPSK modulation, the target SINR can be calculated with the reverse of the following equation: $bitErrorRateQPSK = erfc(\sqrt{SINR})/2$. For the 8.375×10^{-7} bit error rate the target SINR is about 12 dB. Given that the SNR is equal to the 30 dB, we have about 18 dB for the other lines' interference.

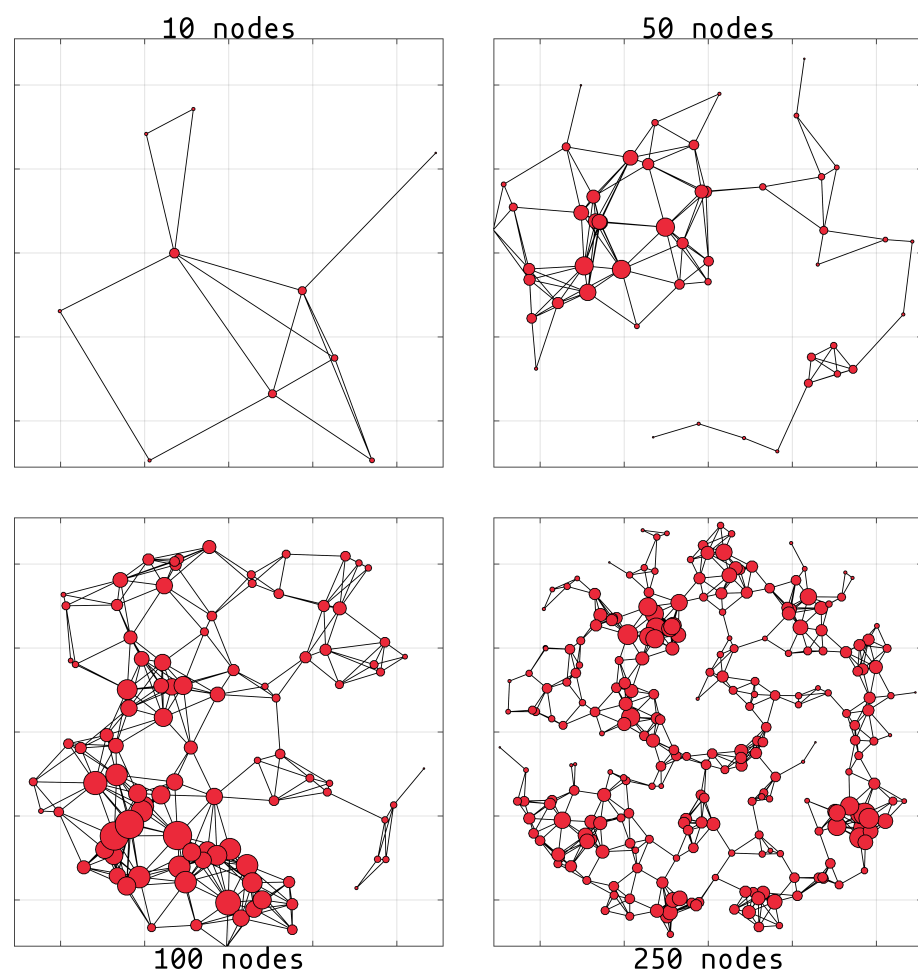


Figure 6. Random network topologies.

We have performed a grid search to estimate the good parameters for pruning distributions and the GA: distribution scale factor: $\{10, 100, 1000\}$; Pareto α parameter: $\{0.001, 1000\}$; GA mutation parameter: $\{0.1, 0.2\}$. In total, we have tested 30 permutations with the number of nodes from 10 to 70 with step 10 and 30 random topologies for each test. The batch size is 128. The scale factor for pruning is equal to 1. The number of iterations after the best found solution has not improved is 2 times larger than the iteration when improvement has happened. The minimal number of iterations is 20. The AWGN mutual interference assumption is used. The SNR is 30 dB.

The cost value is the ratio of the minimal quantity of orthogonal resources to the number of lines. The lower the cost value, the better; a cost equal to 1 is the worst one. Figure 7 shows the cost values averaged over 30 random topologies. Figure 8 shows the place of algorithm parameters set in the sorted sequence of cost values; the less place, the better. Table 3 shows the sum of sorted ranks over all the number of nodes.

We have chosen pruning with uniform distribution: scale factors: $\{1, 2, 4\}$; batch size: 128; mutation probability: 0.1; minimal number of iterations: 20; number of iterations to stop after the solution has improved: 2 times larger than the last improvement iterations; AWGN mutual interference assumption is used; SNR: 30dB. We have performed experiments on 30 random network topologies for each quantity of nodes from 10 through 250 with a step of 10. Averaged over 30 random topologies, results are presented in Figures 9 and 10. Results without pruning are called ultimate. Experiments without pruning are realized for up to 60 nodes, for GA up to 100 nodes, and for random search the whole range.

Table 3. Grid search for the settings of proposed algorithm, ranks of settings.

Settings Names	Rankings
uniform, mP = 0.1	14
pareto (10.0, 0.001), mP = 0.1	20
linear (10.0), mP = 0.1	26
pareto (10.0, 0.001), mP = 0.2	27
uniform, mP = 0.2	36
linear (10.0), mP = 0.2	36
linear (100.0), mP = 0.1	50
linear (1000.0), mP = 0.1	58
linear (100.0), mP = 0.2	61
linear (1000.0), mP = 0.2	69
pareto (100.0, 0.001), mP = 0.1	73
pareto (100.0, 0.001), mP = 0.2	82
pareto (1000.0, 0.001), mP = 0.1	86
pareto (1000.0, 0.001), mP = 0.2	97
uniform, random search	106
linear (10.0), random search	113
pareto (10.0, 0.001), random search	117
linear (100.0), random search	127
linear (1000.0), random search	133
pareto (100.0, 0.001), random search	139
pareto (1000.0, 0.001), random search	147
pareto (10.0, 1000.0), mP = 0.1	154
pareto (100.0, 1000.0), mP = 0.1	161
pareto (1000.0, 1000.0), mP = 0.1	168
pareto (10.0, 1000.0), mP = 0.2	175
pareto (100.0, 1000.0), mP = 0.2	182
pareto (1000.0, 1000.0), mP = 0.2	189
pareto (10.0, 1000.0), random search	196
pareto (100.0, 1000.0), random search	203
pareto (1000.0, 1000.0), random search	210

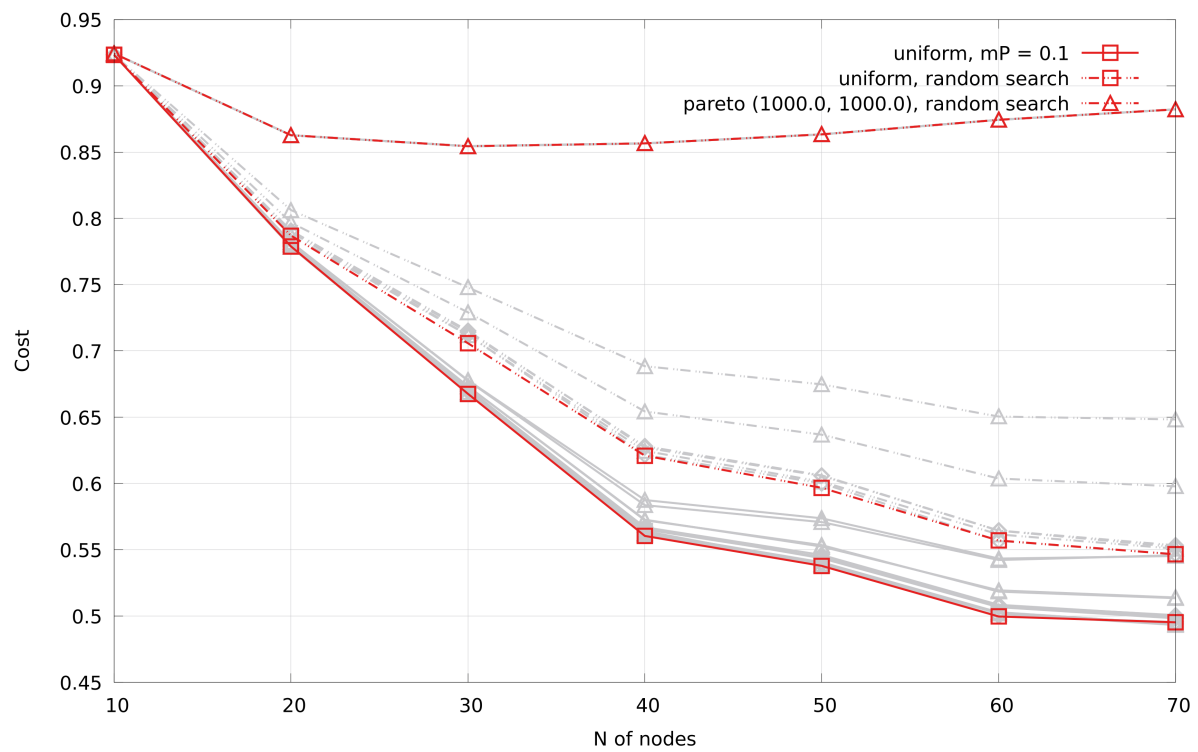


Figure 7. The cost for different settings of the algorithm.

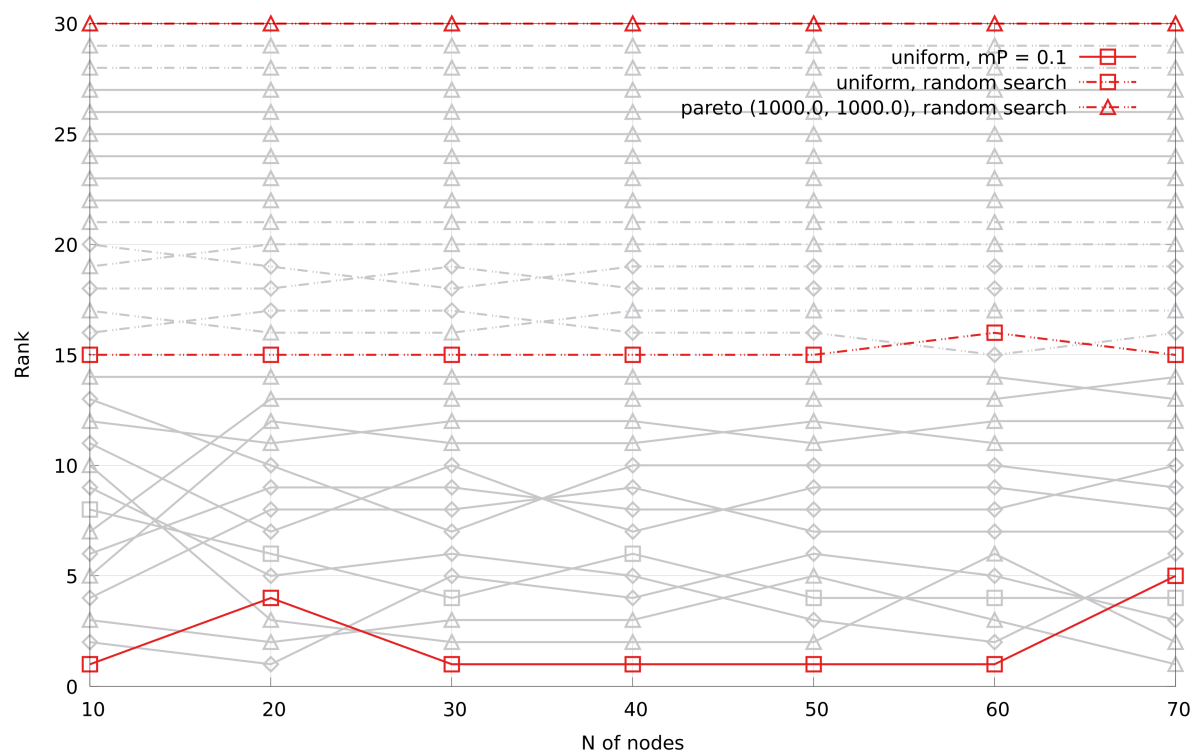


Figure 8. Rankings of different settings of the algorithm.

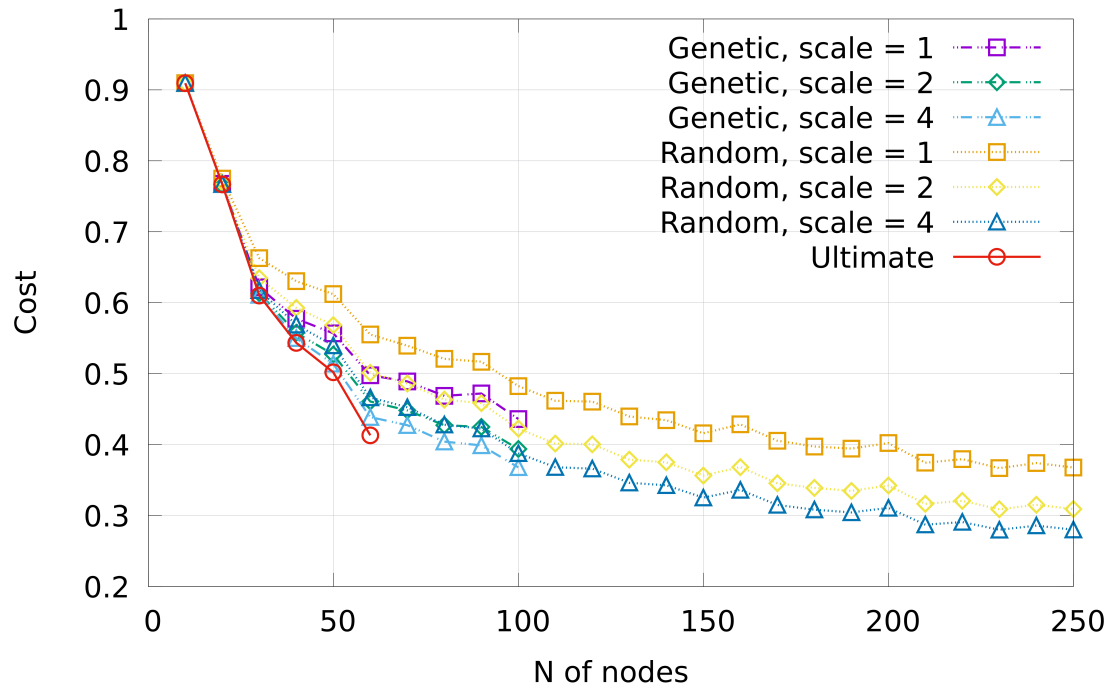


Figure 9. Cost (the ratio of minimal quantity of orthogonal resources to the number of lines).

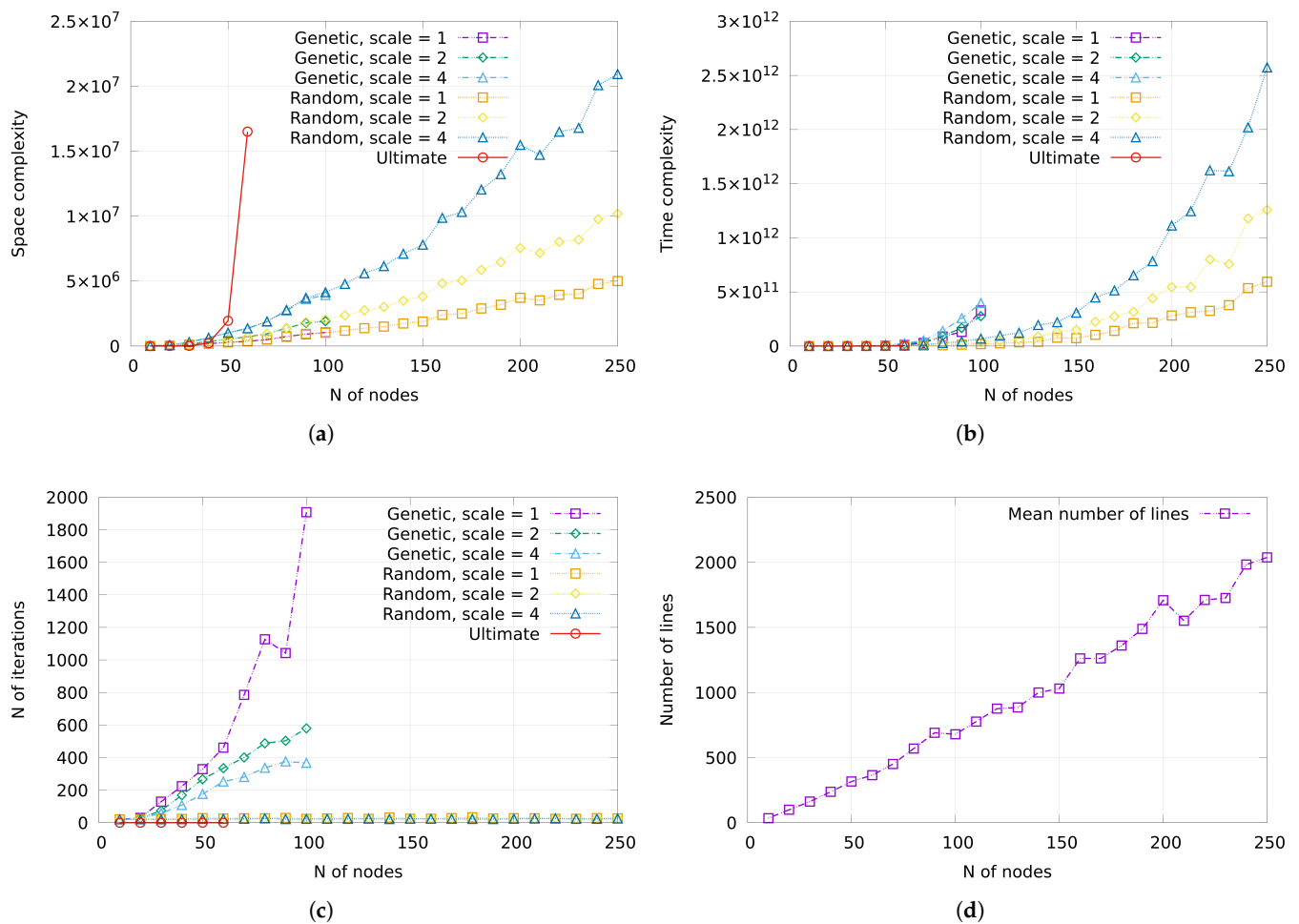


Figure 10. (a) Mean space complexity. (b) Mean time complexity. (c) Mean number of iterations. (d) Mean number of lines.

4. Discussion

The grid search was conducted to choose the settings of the proposed algorithm. The results of the grid search (Table 3, Figures 7 and 8) reveal the following. The genetic algorithm search performs better than the random search algorithm. Lower mutation probability, $mP = 0.1$ in our case, provides better results than $mP = 0.2$ and $mP = 1$ (random search is equivalent to the genetic algorithm with $mP = 1$). The uniform distribution for pruning gives better results than the linear and Pareto ones. The more the linear and Pareto distribution resembles the uniform one, the better the results. In all cases, we obtain monotonically decreasing cost function values as the number of nodes grows, except for the random search with extremely skewed Pareto distribution, where we have a convex cost function.

We have performed experiments on 30 random network topologies for each quantity of nodes from 10 to 250 with step 10 under the AWGN mutual interference assumption. Examples of random topologies consisting of 10, 50, 100, and 250 nodes are shown in Figure 6. The results are presented in Figures 9 and 10.

We can see the following dependence for all algorithm settings: the more nodes in the network, the lower the ratio of minimal orthogonal resource quantity to the active lines quantity. Without pruning the tree of mutually acceptable interfering line sets, we obtain the best results.

Because the size of the tree of mutually acceptable interfering line sets grows exponentially (Figure 10a), we stopped the experiments with the complete tree at 60 nodes. The larger the pruning scale factor, the closer the results are to the complete tree case. Specifically, the results of the genetic algorithm with scale factor 4 are quite similar to the complete tree. The difference between results for scale factors 2 and 4 is less than between scale factors 1 and 2.

The stopping criterion for the genetic algorithm and random search is the sliding maximum number of iterations, which is 2 times larger than the last best solution improvement iteration. In addition, we forced the minimum number of 20 iterations. Figure 10c shows the mean number of iterations. Random search stops slightly above the minimal number of iterations regardless of the number of nodes. The genetic algorithm improves the solution for a larger number of iterations. The number of iterations for the genetic algorithm grows with the number of nodes. We see that, with smaller trees (smaller pruning scale factor), the genetic algorithm takes much more iterations to stop.

The time complexity is shown in Figure 10b. It grows proportionally with increase in space complexity. Time complexity is the highest for the genetic algorithm: we stopped the experiments with the genetic algorithm at 100 nodes. Despite the variety of pruning scale factors, the time complexity of the genetic algorithm differs slightly due to a very different number of iterations. The fastest and simplest way is to use random search. Its cost function is close to the genetic algorithm with the growing number of nodes.

Our proposed algorithm only mitigates the combinatorial explosion but does not eliminate it. To model the extremely large-scale scenario (tens of thousands, hundreds of thousands, or millions of nodes), we will, eventually, need some high-performance computing cluster or even a supercomputer. To make our algorithm even more scalable, we have to, with some heuristic, generate the multiple partitions of line sets and apply our proposed algorithm to each partition. Even though we partition, we still consider the global interference without localizing it. In the case of line partitioning, the minimal achievable number of resources is equal to the number of line partitions. We now have the optimization problem with two optimization spaces: the number of partitions and the partitioning composition itself. With this optimization, we can eventually stumble upon the same resource assignment solution as if we had applied our algorithm directly to the

extremely large number of nodes. The case with only the partition of the lines reduces the aforementioned optimization into our proposed algorithm.

The experiments were conducted with the AWGN interference model, free-space propagation model, and uniform node distribution. But there can be many other cases of node distributions, interference, and space propagation models. The worst time complexity for mutual interference estimation still depends on the number of lines in the set irrespective of the interference and propagation models.

We need to mention the mutual pairwise interference limitation. We assume that if one or both lines exceed the interference thresholds of each other, then adding another line to the pair will make the interference worse. Hence, our proposed algorithm is not applicable in the case of constructive interference. An example of constructive interference is cooperative beamforming, when multiple nodes transmit the same packet to one node to enhance the resulting SNR.

As for the node distribution, lots of “clumped” together nodes would not be able to transmit together either due to a simplex restriction or due to mutual pairwise interference exceeding the threshold. Thus, tree construction will be much faster.

5. Conclusions

In this paper, we have proposed the parallel heuristic algorithm for the NP-hard problem of global minimal orthogonal channel resource allocation and grouping of concurrently transmitting lines for ad hoc and sensor networks.

Given the set of active lines, bit error rate requirements, and the model to estimate mutual interference, we can, in a parallel manner, find the minimal quantity of orthogonal resources for the lines. Simultaneously, we discover which lines can concurrently transmit together using one common orthogonal resource.

This is achieved with the developed approach of incrementally creating the tree with mutually acceptable interfering line sets using pairwise line limitations, and, using the minimal set cover algorithm, we obtain the smallest set of concurrently transmitting line sets. To combat the combinatorial explosion of tree size with the increase in active line quantity, we use tree pruning. Afterwards, we use the genetic algorithm to optimize the tree pruning information to find the pruned tree with the sets that give the smallest minimal set cover. The tree pruning lowers the worst case exponential complexity of full tree construction to the polynomial complexity.

We conducted a grid search to find the best parameters of the proposed technique. Using the parameters found, we conducted experiments that show the minimal quantity of orthogonal resources, space complexity, and time complexity of the proposed technique for random network topologies with up to 250 nodes and the mean number of lines up to 2000. The cases without tree pruning, with random search pruning, and with genetic algorithm pruning are considered. Genetic algorithm pruning achieves the best result, but has relatively high time complexity. Contrary, random pruning gives suboptimal results with substantially lower time complexity.

Due to the parallel nature and the combinatorial explosion of line sets mitigation, the proposed algorithm can be used for the optimal orthogonal channel resource allocation for large scale ad hoc and sensor networks.

An additional benefit of the algorithm lies in its usability to create training sets for machine learning algorithms for the real-time distributed channel resource allocation and to optimize the NOMA groups of 6G cellular networks.

Author Contributions: Conceptualization, V.I.; methodology, V.I.; software, V.I.; validation, V.I.; investigation, V.I.; data curation, V.I.; writing—original draft preparation, V.I. and M.T.; writing—review and editing, V.I. and M.T.; visualization, V.I.; supervision, M.T.; project administration, M.T.; funding acquisition, M.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research has received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors thank Ilya V. Golovkin for his valuable insights, Nikolay V. Klenov for his substantial suggestions, and Olga V. Ilina for her consulting and inspiration.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

6G	Sixth-generation technology for wireless communications
AI	Artificial Intelligence
AWGN	Additive white Gaussian noise
CFT	Caprara, Fischetti, and Toth algorithm for the minimal set cover search
DSP	Digital signal processing
EHLO	Elephant herding lion optimizer
GA	Genetic optimization algorithm
GBD	Generalized Benders Decomposition
MFCM	Multiple Fuzzy C-means Clustering algorithms
MIMO	Multiple Input–Multiple Output
MPA	Marine predator optimization algorithm
NOMA	Non-orthogonal multiple access
OFDMA	Orthogonal frequency division multiple access

References

1. Hekmat, R. *Ad-Hoc Networks: Fundamental Properties and Network Topologies*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
2. Karl, H.; Willig, A. *Protocols and Architectures for Wireless Sensor Networks*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
3. Sarangapani, J. *Wireless Ad Hoc and Sensor Networks: Protocols, Performance, and Control*; CRC Press: Boca Raton, FL, USA, 2017.
4. Li, J.; Blake, C.; De Couto, D.S.; Lee, H.I.; Morris, R. Capacity of ad hoc wireless networks. In Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy, 16 July 2001; pp. 61–69.
5. Toumpis, S.; Goldsmith, A.J. Capacity regions for wireless ad hoc networks. *IEEE Trans. Wirel. Commun.* **2003**, *2*, 736–748. [\[CrossRef\]](#)
6. Toumpis, S.; Goldsmith, A. Ad hoc network capacity. In Proceedings of the Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers (Cat. No. 00CH37154), Pacific Grove, CA, USA, 29 October–1 November 2000; Volume 2, pp. 1265–1269.
7. Li, X.Y.; Tang, S.J.; Frieder, O. Multicast Capacity for Large Scale Wireless ad Hoc Networks. In Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking, Montreal, QC, Canada, 9–14 September 2007; pp. 266–277.
8. Grossglauser, M.; Tse, D.N. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.* **2002**, *10*, 477–486. [\[CrossRef\]](#)
9. Spyropoulos, A.; Raghavendra, C.S. Capacity bounds for ad-hoc networks using directional antennas. In Proceedings of the IEEE International Conference on Communications (ICC'03), Anchorage, AK, USA, 11–15 May 2003; Volume 1, pp. 348–352.
10. Li, P.; Zhang, C.; Fang, Y. The capacity of wireless ad hoc networks using directional antennas. *IEEE Trans. Mob. Comput.* **2010**, *10*, 1374–1387. [\[CrossRef\]](#)
11. Hunter, A.M.; Andrews, J.G.; Weber, S. Transmission capacity of ad hoc networks with spatial diversity. *IEEE Trans. Wirel. Commun.* **2008**, *7*, 5058–5071. [\[CrossRef\]](#)
12. Jiang, C.; Shi, Y.; Hou, Y.T.; Kompella, S. On the asymptotic capacity of multi-hop MIMO ad hoc networks. *IEEE Trans. Wirel. Commun.* **2011**, *10*, 1032–1037. [\[CrossRef\]](#)

13. Liu, B.; Thiran, P.; Towsley, D. Capacity of a wireless ad hoc network with infrastructure. In Proceedings of the 8th ACM International Symposium on MOBILE ad Hoc Networking and Computing, Montreal, QC, Canada, 9–14 September 2007; pp. 239–246.
14. Lin, X.; Shroff, N.B. The fundamental capacity-delay tradeoff in large mobile ad hoc networks. In Proceedings of the Third Annual Mediterranean Ad Hoc Networking Workshop, Bodrum, Turkey, 27–30 June 2004.
15. Comaniciu, C.; Poor, H.V. On the capacity of mobile ad hoc networks with delay constraints. *IEEE Trans. Wirel. Commun.* **2006**, *5*, 2061–2071. [\[CrossRef\]](#)
16. Neely, M.J.; Modiano, E. Capacity and delay tradeoffs for ad hoc mobile networks. *IEEE Trans. Inf. Theory* **2005**, *51*, 1917–1937. [\[CrossRef\]](#)
17. Urgaonkar, R.; Neely, M.J. Network capacity region and minimum energy function for a delay-tolerant mobile ad hoc network. *IEEE/ACM Trans. Netw.* **2011**, *19*, 1137–1150. [\[CrossRef\]](#)
18. Garcia-Luna-Aceves, J.; Sadjadpour, H.R.; Wang, Z. Challenges: Towards truly scalable ad hoc networks. In Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking, Montreal, QC, Canada, 9–11 September 2007; pp. 207–214.
19. Garcia-Luna-Aceves, J.; Sadjadpour, H.R.; Wang, Z. Extending the capacity of ad hoc networks beyond network coding. In Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing, Shanghai, China, 21–25 September 2007; pp. 91–96.
20. Wang, H.; Fan, P.; Letaief, K.B. Maximum flow and network capacity of network coding for ad-hoc networks. *IEEE Trans. Wirel. Commun.* **2007**, *6*, 4193–4198. [\[CrossRef\]](#)
21. Isaac, R.A.; Sundaravadivel, P.; Marx, V.N.; Priyanga, G. Enhanced novelty approaches for resource allocation model for multi-cloud environment in vehicular Ad-Hoc networks. *Sci. Rep.* **2025**, *15*, 9472. [\[CrossRef\]](#)
22. Kordbacheh, H.; Oskouei, H.D.; Mokari, N. Robust cross-layer routing and radio resource allocation in massive multiple antenna and OFDMA-based wireless ad-hoc networks. *IEEE Access* **2019**, *7*, 36527–36539. [\[CrossRef\]](#)
23. Luo, X.; Lu, X.; Yin, B.; Yang, K. Resource allocation for joint communication and positioning in mmWave ad hoc networks. *IEEE Trans. Veh. Technol.* **2023**, *73*, 2187–2201. [\[CrossRef\]](#)
24. Wang, J.; Bai, L.; Chen, J.; Wang, J. Starling flocks-inspired resource allocation for ISAC-aided green ad hoc networks. *IEEE Trans. Green Commun. Netw.* **2023**, *7*, 444–454. [\[CrossRef\]](#)
25. Teng, Y.; Yang, H.; Yao, Q.; Gu, R.; Sun, Z.; Xu, A.; Liu, F.; Zhang, J. SRS-proactive-aware resource allocation based on all-optical wavelength converters in C+ L band optical networks. *J. Light. Technol.* **2024**, *42*, 6632–6646. [\[CrossRef\]](#)
26. Wang, J.; Jiang, C. *Flying ad Hoc Networks: Cooperative Networking and Resource Allocation*; Springer Nature: Berlin/Heidelberg, Germany, 2022.
27. Liu, J.; Xu, Y.; Li, Z. Resource allocation for performance enhancement in mobile ad hoc networks. *IEEE Access* **2019**, *7*, 73790–73803. [\[CrossRef\]](#)
28. Cerquitelli, T.; Meo, M.; Curado, M.; Skorin-Kapov, L.; Tsiropoulou, E.E. Machine learning empowered computer networks. *Comput. Netw.* **2023**, *230*, 109807. [\[CrossRef\]](#)
29. Yang, H.; Yao, Q.; Bao, B.; Yu, A.; Zhang, J.; Vasilakos, A.V. Multi-associated parameters aggregation-based routing and resources allocation in multi-core elastic optical networks. *IEEE/ACM Trans. Netw.* **2022**, *30*, 2145–2157. [\[CrossRef\]](#)
30. Sun, Z.; Yang, H.; Li, C.; Yao, Q.; Teng, Y.; Zhang, J.; Liu, S.; Li, Y.; Vasilakos, A.V. A resource allocation scheme for edge computing network in smart city based on attention mechanism. *ACM Trans. Sens. Netw.* **2024**, *3*. [\[CrossRef\]](#)
31. Yang, H.; Yao, Q.; Yu, A.; Lee, Y.; Zhang, J. Resource assignment based on dynamic fuzzy clustering in elastic optical networks with multi-core fibers. *IEEE Trans. Commun.* **2019**, *67*, 3457–3469. [\[CrossRef\]](#)
32. Cao, X.; Ma, R.; Liu, L.; Shi, H.; Cheng, Y.; Sun, C. A machine learning-based algorithm for joint scheduling and power control in wireless networks. *IEEE Internet Things J.* **2018**, *5*, 4308–4318. [\[CrossRef\]](#)
33. Shelim, R.; Ibrahim, A.S. A fast graph kernel based classification method for wireless link scheduling on Riemannian manifold. *arXiv* **2021**, arXiv:2106.13707. [\[CrossRef\]](#)
34. Lamm, S.; Schulz, C.; Strash, D.; Williger, R.; Zhang, H. Exactly solving the maximum weight independent set problem on large real-world graphs. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (Alenex)*; SIAM: New Delhi, India, 2019; pp. 144–158.
35. Abbasalizadeh, M.; Vellamchety, K.; Rayavaram, P.; Narain, S. Dynamic link scheduling in wireless networks through fuzzy-enhanced deep learning. *IEEE Open J. Commun. Soc.* **2024**, *5*, 6832–6848. [\[CrossRef\]](#)
36. Zhao, Z.; Verma, G.; Rao, C.; Swami, A.; Segarra, S. Link scheduling using graph neural networks. *IEEE Trans. Wirel. Commun.* **2022**, *22*, 3997–4012. [\[CrossRef\]](#)
37. Zhao, Z.; Verma, G.; Swami, A.; Segarra, S. Delay-oriented distributed scheduling using graph neural networks. In Proceedings of the ICASSP 2022—2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022; pp. 8902–8906.

38. Zou, C.; Haas. Optimal resource allocation for UWB wireless ad hoc networks. In Proceedings of the 2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, Berlin, Germany, 11–14 September 2005; Volume 1, pp. 452–456.
39. Zou, C.; Haas, Z.J.; Zou, S. Throughput maximization in UWB-based ad-hoc networks. *Sci. China Inf. Sci.* **2010**, *53*, 2538–2547. [[CrossRef](#)]
40. Moscibroda, T.; Wattenhofer, R. The complexity of connectivity in wireless networks. In Proceedings of the INFOCOM 2006: 25th Annual Joint Conference of the IEEE Computer and Communications Societies, Barcelona, Spain, 23–29 April 2006.
41. Yu, J.; Huang, B.; Cheng, X.; Atiquzzaman, M. Shortest link scheduling algorithms in wireless networks under the SINR model. *IEEE Trans. Veh. Technol.* **2016**, *66*, 2643–2657. [[CrossRef](#)]
42. Wan, P.J.; Frieder, O.; Jia, X.; Yao, F.; Xu, X.; Tang, S. *Wireless Link Scheduling Under Physical Interference Model*; IEEE: Piscataway, NJ, USA, 2011.
43. Blough, D.M.; Resta, G.; Santi, P. Approximation algorithms for wireless link scheduling with SINR-based interference. *IEEE/ACM Trans. Netw.* **2010**, *18*, 1701–1712. [[CrossRef](#)]
44. Xu, X.; Tang, S.; Wan, P.J. Maximum weighted independent set of links under physical interference model. In *Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 68–74.
45. Wang, C.; Yu, J.; Yu, D.; Huang, B.; Yu, S. An improved approximation algorithm for the shortest link scheduling in wireless networks under SINR and hypergraph models. *J. Comb. Optim.* **2016**, *32*, 1052–1067. [[CrossRef](#)]
46. Huang, B.; Yu, J.; Cheng, X.; Chen, H.; Liu, H. SINR based shortest link scheduling with oblivious power control in wireless networks. *J. Netw. Comput. Appl.* **2017**, *77*, 64–72. [[CrossRef](#)]
47. Liu, X.; Haenggi, M. Throughput analysis of fading sensor networks with regular and random topologies. *EURASIP J. Wirel. Commun. Netw.* **2005**, *2005*, 397962. [[CrossRef](#)]
48. Teng, Y.; Liu, M.; Yu, F.R.; Leung, V.C.; Song, M.; Zhang, Y. Resource allocation for ultra-dense networks: A survey, some research issues and challenges. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 2134–2168. [[CrossRef](#)]
49. Islam, T.; Kwon, C. Survey on the state-of-the-art in device-to-device communication: A resource allocation perspective. *Ad Hoc Netw.* **2022**, *136*, 102978. [[CrossRef](#)]
50. Jaramillo, J.J.; Srikant, R. Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
51. Zhang, W.; Miller, M.J. Baseband equivalents in digital communication system simulation. *IEEE Trans. Educ.* **1992**, *35*, 376–382. [[CrossRef](#)]
52. Karp, R.M. On the computational complexity of combinatorial problems. *Networks* **1975**, *5*, 45–68. [[CrossRef](#)]
53. Caprara, A.; Fischetti, M.; Toth, P. A heuristic method for the set covering problem. *Oper. Res.* **1999**, *47*, 730–743. [[CrossRef](#)]
54. Blelloch, G.E.; Peng, R.; Tangwongsan, K. Linear-work greedy parallel approximate set cover and variants. In Proceedings of the Twenty-Third Annual ACM Symposium on PARALLELISM in Algorithms and Architectures, San Jose, CA, USA, 4–6 June 2011; pp. 23–32.
55. Sekar, S.; Reggia, J. Parallel set covering algorithms. In Proceedings of the Fourth Conference on Artificial Intelligence Applications, San Diego, CA, USA, 14–18 March 1988; pp. 274–275.
56. Zhang, X.Y.; Zhang, J.; Gong, Y.J.; Zhan, Z.H.; Chen, W.N.; Li, Y. Kuhn–Munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks. *IEEE Trans. Evol. Comput.* **2015**, *20*, 695–710. [[CrossRef](#)]
57. Koufogiannakis, C.; Young, N.E. Distributed and parallel algorithms for weighted vertex cover and other covering problems. In Proceedings of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, 10 August 2009; pp. 171–179.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.