A Novel Method for Scheduling of Wireless Ad Hoc Networks in Polynomial Time

Alper Köse, Hakan Gökcesu, Noyan Evirgen, Kaan Gökcesu, and Muriel Médard, *Fellow, IEEE*

***Abstract*— In this article, we address the scheduling problem in wireless ad hoc networks by exploiting the computational advantage that comes when scheduling problems can be represented by claw-free conflict graphs where we consider a wireless broadcast medium. It is possible to formulate a scheduling problem of broadcast transmissions as finding the maximum weighted independent set (MWIS) in the conflict graph of the network. Finding the MWIS of a general graph is NP-hard leading to an NP-hard complexity of scheduling. In a claw-free conflict graph, MWIS may be found in polynomial time leading to a throughput-optimal scheduling. We show that the conflict graphs of certain wireless ad hoc networks are claw-free. In order to obtain claw-free conflict graphs in general networks, we suggest introducing additional conflicts (edges) with the aim of keeping the decrease in MWIS size minimal. To this end, we introduce an iterative optimization problem to decide where to introduce edges and investigate its efficient implementation. We conclude that the claw breaking method by adding extra edges can perform very close to optimal scenario and better than the polynomial time maximal independent set scheduling benchmark under the necessary assumptions.**

***Index Terms*— Scheduling in polynomial time, wireless ad hoc networks, conflict graph, Claw-free graph, maximum weighted independent set.**

I. Introduction

WE STUDY the scheduling problem in wireless ad hoc networks. In a wireless broadcast medium, networks are usually interference limited and hence, interfering transmissions cannot be made simultaneously. On the other hand, it is desirable to maximize the overall volume of simultaneous transmissions in order to obtain a high throughput in the network. This trade-off forces us to do scheduling which aims to maximize the volume of non-interfering simultaneous transmissions in the considered time slot.

Manuscript received December 14, 2019; revised April 4, 2020 and July 29, 2020; accepted September 9, 2020. Date of publication September 28, 2020; date of current version January 8, 2021. The associate editor coordinating the review of this article and approving it for publication was Y. Cui. *(Corresponding author: Alper Köse.)*

Alper Köse is with the Electrical and Electronics Engineering Department, Bogazici University, 34470 Istanbul, Turkey (e-mail: alper.kose@boun.edu.tr).

Hakan Gökcesu is with the Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey (e-mail: hgokcesu@ee.bilkent.edu.tr).

Noyan Evirgen is with the Electrical and Computer Engineering Department, University of California, Los Angeles, CA 90095 USA (e-mail: nevirgen@ucla.edu).

Kaan Gökcesu and Muriel Médard are with the Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: gokcesu@mit.edu; medard@mit.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2020.3025448

Arikan [1] proves that the scheduling problem is NP-complete for packet radio networks, which are the earliest version of wireless networks. Ephremides and Truong [2] study the problem of scheduling broadcast transmissions in a multihop interference limited wireless network while aiming to optimize throughput. They show that the problem is NP-complete. Sharma *et al.* [3] also consider the problem of throughput optimal scheduling in wireless networks subject to interference constraints where they assume no two links within K hops can successfully transmit in the same time slot. They conclude that the problem can be solved in polynomial time for $K = 1$ whereas it is NP-hard for K>1. Hajek and Sasaki [4] give polynomial time algorithms for link scheduling in a spread spectrum wireless network where each node is allowed to converse with only one other node at a time. Our modeling of possible transmissions in interference limited network setup and approach using conflict graphs are the same with Traskov *et al.*'s work [5]. Therefore, in our case, scheduling has an NP-hard complexity as in their study [5] for general conflict graphs.

Due to the complexity of the scheduling problem, the common approach is to propose an approximate solution which depends on maximal set scheduling [5]–[9]. For example, Traskov *et al.* [5] propose an approach that greedily chooses the maximal independent sets instead of finding the maximum independent set since the complexity of the latter is NP-hard despite providing the optimal solution. Another example is Bao and Garcia-Luna-Aceves [10] who propose a sub-optimal interference scheme where two nodes within two hops cannot transmit simultaneously. As known, there is no optimal solution for the scheduling problem in polynomial time, and in this study, we propose to change the perspective and investigate this problem from another angle as explained in the following paragraphs.

According to our assumptions and the Protocol model that Gupta and Kumar [11] define, we construct the conflict graph of a given network where we model possible transmissions as the vertices of the conflict graph. We define the neighbors of a transceiver as the set of transceivers this transceiver can transmit to. To model possible transmissions, we first find the neighbors of each transceiver, then we assign each transceiver as the sender and every possible combination of its neighbors as its possible receivers, implicitly meaning that there is a directed hyperedge from the sender to its possible receivers in the network setup. In the end, the number of possible transmissions for a transceiver is equal to the number of the

subsets of its set of neighbors except the zero set. An edge between two vertices of a conflict graph means that it is not possible to schedule these two transmissions for the same time slot since they interfere with each other. To find the edges between the modeled vertices in the conflict graph, again, we use our assumptions with the Protocol model and give the interference relationships between possible transmissions. Note that our setup is different than link-based scheduling due to the broadcast modeling approach.

Our work builds upon the ideas of network coding. The network coding concept is introduced by Ahlswede *et al.* [12] and may be used to improve the performance of networks. Ho *et al.* [13], [14] study the random linear network coding approach and show that it can achieve capacity in certain multisource multicast networks. We implicitly consider random linear network coding over a considered wireless network in a bandwidth limited regime as in a previous study [5], and thus, our conflict graph construction accounts for this. The ultimate aim is to compute an optimal network coding subgraph and a schedule that can support it. We require network coding subgraph to lie in the independent set polytope of the conflict graph so that the subgraph can be partitioned into a combination of valid schedules. Although this optimization has an NP-hard complexity in general, it can be achieved in polynomial time for claw-free conflict graphs. So, the conflict graph contains the combinatorial difficulty of the scheduling problem. In this study, we extend a previous work [15] by concentrating on the scheduling problem and considering the graph-theoretical side to get claw-freeness in networks.

Scheduling may be modeled as a maximum weighted independent set (MWIS) problem in the conflict graph [5], [16]–[18]. Therefore, the scheduling complexity is equivalent to the complexity of finding MWIS in the derived conflict graph of a given wireless network as shown by Traskov *et al.* [5]. Since there are algorithms [19]–[22] that can find MWIS in polynomial time in claw-free graphs, we can perform polynomial time scheduling if we can get a claw-free conflict graph.

Note that there are other types of graphs besides claw-free ones, e.g. apple-free, fork-free, chair-free, etc., the MWIS of which can be identified in polynomial time [23], [24]. The reason we pay our attention on claws is that the structure of a claw is the simplest (defined by a total of four vertices) among the other similar types of sub-graphs, i.e. we can modify the original graph just by focusing on the local attributes and a claw is one of the structures that can be identified very easily. There are also perfect graphs which enjoy polynomial time MWIS algorithms, however, it is not easy to decide to which kind of perfect graph a conflict graph should be transformed. Even if this can be done without any problem, a large number of changes are likely needed to transform a conflict graph into a perfect graph. Due to such reasons, we are interested in claw-freeness property in this article.

We investigate certain families of networks, which have claw-free conflict graphs, including line networks and tree networks. Since there are many limitations to construct networks which have claw-free conflict graphs, we are able to set them up only under specific assumptions. Typical wireless networks usually do not have claw-free conflict graphs; thus, we propose to add edges to their conflict graphs to reach claw-freeness. Note that, introducing only a few edges can break all claws and hence perform very well as confirmed with simulations. Another advantage of this method is that we are able to automatically decide between which nodes the edges must be introduced. Thus, we are able to incorporate an optimization problem which breaks all the claws by adding edges between nodes so that the decrease in the optimal scheduling throughput, i.e., the weight of the MWIS, is minimal. We also discuss how physical modifications can also be made in the network whereas such modifications require network flexibility. Moreover, there must be an autonomous system to immediately propose the modifications that should be made.

In short, our main contributions are two-fold. We first introduce certain families of networks which can be scheduled in polynomial time. Following that, we generalize to any broadcast network by providing a novel polynomial time scheduling method based on adding new edges to a conflict graph without any intervention to the network setup.

The remainder of the paper is organized as follows. In Section II, we provide details on conflict graph construction and present different scenarios, which are on line, tree and diamond networks for which the conflict graphs are claw-free. In Section III, we explain the methods that may be used in order to make a general network suitable for polynomial time scheduling. In Section IV, we give details on the claw-breaking strategy that we use in the conflict graph. We present the simulation results and evaluate the quality of the claw breaking strategy in Section V. In Section VI, we discuss the possibilities for the directions of future work. We conclude the paper in Section VII.

II. Constructions for Polynomial Time Scheduling

Definition 1 (Claw-Free Graph): A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is claw-free if none of its vertices $\mathcal{V}$ has three pairwise nonadjacent neighbors [25].

Definition 2 (Independent Set): Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a subset of vertices $\mathcal{S} \subseteq \mathcal{V}$ is an independent set if $\forall i, j \in \mathcal{S}$, $\{i, j\} \notin \mathcal{E}$ is satisfied.

Definition 3 (Maximum Weighted Independent Set): Given an undirected and weighted graph $\mathcal{G} = (\mathcal{V}, w_{\mathcal{V}}, \mathcal{E})$, where the weights are assigned on vertices $w_{\mathcal{V}} : \mathcal{V} \rightarrow \mathbb{R}$, a subset of vertices $\mathcal{S} \subseteq \mathcal{V}$ constitute a maximum weighted independent set if $\forall i, j \in \mathcal{S}$, $\{i, j\} \notin \mathcal{E}$ and there is no other independent set having a greater weight sum $\sum_{i \in \mathcal{S}} w_{\mathcal{V}}(i)$.

Throughout the scenarios, we use the Protocol model [11] and K-hop interference model [3] with small variations to represent networks instead of the Physical model [11], which takes $SINR$ levels into account.

A. Scenario I - Line Networks

We have a wireless ad hoc network with n transceivers with the following assumptions: A transceiver can receive from at

most one transceiver in a time slot, omnidirectional antennas are deployed, and time division duplex transceivers are used.

We model interference between transmissions with the Protocol model. Assume transceiver i_a is transmitting to transceiver i_b while transceiver i_c is transmitting to another one. Then, the transmission between i_a and i_b will be successful if and only if following inequalities are satisfied:

$$|P_{i_a} - P_{i_b}| \leq r_T \tag{1}$$

$$|P_{i_b} - P_{i_c}| \geq (1+\Delta)|P_{i_a} - P_{i_b}| \tag{2}$$

where P_{i_a} is the position of transceiver i_a. Inequality (1) means that the transceivers have a maximum range of transmission r_T. Inequality (2) means that, in a communication pair, among all transmitting nodes, the receiver of this pair must be the closest to its transmitter with a guard zone Δ. Without loss of generality $\Delta > 0$, and it can be chosen arbitrarily small for simplicity.

The conflict graph, as the name suggests, is the graph of transmissions which conflict with each other. We need to represent the potential conflicts among the transmissions since they are directly related to scheduling. In a conflict graph, transmissions are represented by vertices, and conflicts are represented by edges. A conflict exists under certain conditions which depend on the assumptions on the network. After identifying all of the potential conflicts among the transmissions, it is possible to check if the conflict graph is claw-free.

In the construction of the conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ is the set of possible transmissions, and $\mathcal{E}$ is the set of conflicts. The vertices and edges of the conflict graph are found as follows. Let us assume $\mathcal{T}$ is the set of transceivers, $\{i_1, i_2, \ldots, i_n\} \in \mathcal{T}$. There is a set $N(i_x)$ for all $i_x \in \mathcal{T}$ whose elements are neighbors of i_x. Then, we are able to define the set of possible receivers. There is a set $Y_x = P(N(i_x))$ for all $i_x \in \mathcal{T}$ if $|N(i_x)| \geq 1$, where $P(\{\cdot\})$ is the power set of $\{\cdot\}$ without the empty set. We find the possible transmission by defining i_x as the sender and each element of Y_x as the receiver set, respectively, and we do this for all $i_x \in \mathcal{T}$. In the end, we can symbolize possible transmissions as $S_k = (i_{t'}, J_k)$ for $k = \{1, \ldots, m\}$ where $m = \sum_{t=1}^{n} |Y_t|$ and $J_k \in Y_{t'}$. Say $S_1, S_2 \in \mathcal{V}$, then $\{S_1, S_2\} \in \mathcal{E}$ if any of the following conditions hold for $S_1 = (i_1, J_1)$ and $S_2 = (i_2, J_2)$, which means they cannot be scheduled for the same time slot:

C2.1.1 $i_1 = i_2$.
C2.1.2 $(i_1 \in J_2)$ or $(i_2 \in J_1)$.
C2.1.3 $J_1 \cap J_2 \neq \emptyset$.
C2.1.4 $\exists j \in J_1,\ |i_2 - j| \leq (1+\Delta)|i_1 - j|$.
C2.1.5 $\exists j \in J_2,\ |i_1 - j| \leq (1+\Delta)|i_2 - j|$.

In the network, we suppose there is a bound on the number of receivers each transceiver can transmit to, i.e. $|N(i_x)| \leq d$ for all $i_x \in \mathcal{T}$. With this, we ensure that we can complete the conflict graph setup in a reasonable time. The computational complexity of creating vertices to model the possible transmissions becomes $O(n2^d)$ since we have n transceivers in the network, and each transceiver can lead to at most $2^d - 1$ possible transmissions in the conflict graph. Since there may be at most $n(n-1)/2$ edges in a graph with n vertices, the complexity of adding the necessary edges between vertices to model the conflicts is $O(n^2 2^{2d})$. Overall, they imply a polynomial time complexity if we satisfy $|N(i_x)| \leq d$ for all $i_x \in \mathcal{T}$ and $d \ll n$, specifically $d \in O(\log n)$.

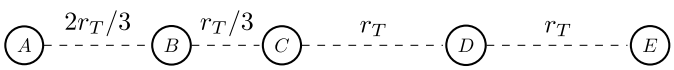


Fig. 1. A possible physical arrangement of a wireless network which leads to a claw-free conflict graph for Scenario I.

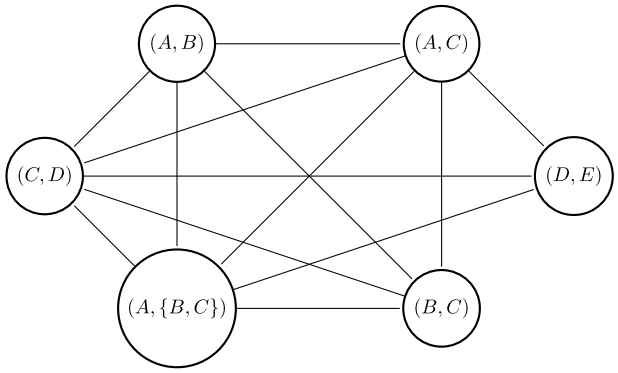


Fig. 2. Conflict graph of the network seen in Fig. 1.

Example I: A possible arrangement of wireless nodes to have a claw-free conflict graph is shown in Fig. 1. The source and the sink can be thought as the nodes A and E, respectively, and we are interested in the transmissions in the direction of the sink. Let the maximum possible transmission distance be r_T and Δ be very small. Then, we can model the conflict graph of this network as seen in Fig. 2. The independent sets of the conflict graph are $\{\}$, $\{(A, B)\}$, $\{(A, C)\}$, $\{(D, E)\}$, $\{(B, C)\}$, $\{(A, \{B, C\})\}$, $\{(C, D)\}$ $\{(A, B), (D, E)\}$ and $\{(B, C), (D, E)\}$.

We can generalize Example I by realizing that the conflict graphs of the line networks are claw-free provided that there is enough distance between the nodes to make a 3-node away transmission impossible and they are sufficiently spread out, that is, we should physically satisfy $r_T < \min_{i \in \{1,2,\ldots,n-3\}} |P_i - P_{i+3}|$ and $\max_{i \in \{1,2,\ldots,n-1\}} |P_i - P_{i+1}| \leq \min_{j \in \{1,2,\ldots,n-2\}} |P_j - P_{j+2}|$, where P_i denotes the position of the ith node in the network. These inequalities can be easily satisfied by many different positioning scenarios, so, for simplicity, we can use a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ to represent our model, where $\mathcal{N}$ denotes the nodes, and $\mathcal{A}$ denotes the hyperedges to symbolize valid transmissions between wireless nodes. The hypergraph of a line network, which has a claw-free conflict graph, changes depending on the number of nodes to which a node i can transmit, which can be 1 or 2 for all $i \in \{1, 2, \ldots, n-2\}$ and 1 for $i \in \{n-1\}$, since the transmissions are unidirectional. We have 2^{n-2} different possible hypergraphs of a line network, with n nodes, all of which lead to claw-free conflict graphs. For instance, the hypergraph of the wireless network given in Fig. 1 can be seen in Fig. 3. In Example I, the first node is able to transmit up to the next two neighbors, whereas the other nodes are only able to transmit to the next node. Transmissions (A, B) and (C, D) are not simultaneously possible, because we use the

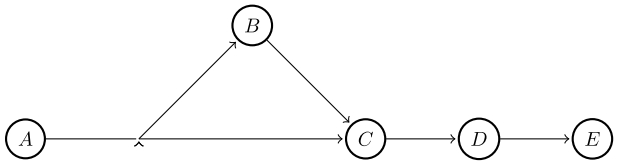


Fig. 3. Hypergraph of the network seen in Fig. 1.

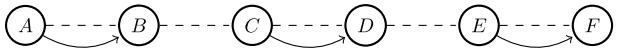


Fig. 4. Illustration for proof of Theorem 1.

Protocol model to decide on interference relations. Receiver B is closer to C, a transmitter of another transmission, than to A, and this violates the constraint (2). Directed antennas, which we do not assume in our scenario, could be used to avoid the interference.

Theorem 1: Under the assumptions given in Scenario I, which lead to C2.1.1-C2.1.5, the conflict graph of a line network where a transceiver is able to transmit to at most 2 hops, the minimum 2-hop distance is greater than the maximum 1-hop distance, and all transceivers convey information in the direction from source to sink is guaranteed to be claw-free.

Proof: Let us prove this theorem by contradiction. To have a claw in the conflict graph, a transmission v_1 should have conflicts with three other transmissions v_2, v_3, v_4, whereas those three should not have any conflicts between them. Let us use Fig. 4 for ease of understanding. Since transmissions are from the source to the sink, assume that a node only transmits to nodes that are located closer to the sink in terms of the hop distance. To this end, assume v_1 as the central node of a possible claw, $v_1 = (C, D)$. Then, we can have one interfering transmission from the source side of C, say $v_2 = (A, B)$, and one from the sink side of D, $v_3 = (E, F)$ not to have interference between v_2 and v_3, B and E are chosen to be as far as possible. In such a situation, the transmitting nodes C and E cause interference on the receiving nodes B and D, respectively. Now, we have to place the transmitter and the receiver of the last transmission. This one has to interfere with (C, D) without interfering with (A, B) and (E, F) to induce a claw in the conflict graph of the network. If we place the transmitter on the source side of C, this leads to an interference with (A, B) which will break the claw, so, this option is not possible. Additionally, we cannot place the receiver in the sink side of D since this leads to an interference with (E, F) which will, again, break the claw. Therefore, since the receiver must be on the sink side relative to the transmitter, the only remaining option is to place both the transmitter and the receiver between C and D. However, this option makes C able to transmit to 3 different nodes where we assume each node is able to transmit to at most 2 nodes. ■

B. Scenario II - Tree Networks

We can also have other network topologies that lead to claw-free conflict graphs. One of them is a tree representation of the network, but since it is harder to get claw-freeness with the same assumptions for the line topology model, we propose a new set of assumptions: A transceiver can receive from at most one transceiver in a time slot. Time division duplex transceivers are used. Nodes are arranged as a tree topology. We directly work on the hypergraph model without any consideration on the physical locations of nodes and assume an interference model based on hops instead of the Protocol model. Transmissions are in the direction from the root node to the leaves of the tree. Directed antennas are used, so, interference can only occur in the forward direction along the tree. A transmitting node does not lead to any interference to the receivers which are 3-hops or further away from it. Only one node in every level can have children.

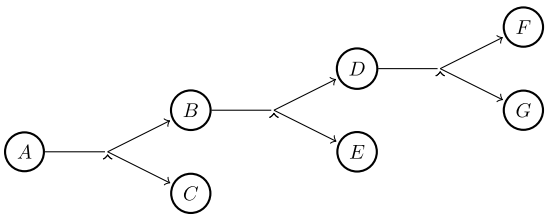


Fig. 5. Example tree network with claw-free conflict graph for Scenario II.

In the construction of the conflict graph, let $v_1, v_2 \in \mathcal{V}$ be in the conflict graph.$\{v_1, v_2\} \in \mathcal{E}$ if any of the conditions below holds:

C2.2.1 $i_1 = i_2$.
C2.2.2 $(i_1 \in J_2)$ or $(i_2 \in J_1)$.
C2.2.3 $J_1 \cap J_2 \neq \emptyset$.
C2.2.4 (i_1 is a child of i_2) or (i_2 is a child of i_1).

Theorem 2: Under the assumptions given in Scenario II, which lead to C2.2.1-C2.2.4, the conflict graph of a wireless network is guaranteed to be claw-free.

Proof: Let us assume that the root node belongs to level 1, and a node, which has a distance k to the root in terms of hyperedge number, belongs to level $k + 1$. Now, consider the transmission v_1, from a node i_k in level k to its children that reside in level $k + 1$. We have $2^{N_k} - 1$ different interfering transmissions to v_1 which are from level $k - 1$ to k where N_k is the number of children of i_k's parent. Since only one of these transmissions can be scheduled in one time slot, they induce a complete subgraph in the conflict graph. Therefore, we can only choose one transmission, say v_2, from level $k-1$ to k which has interference with v_1 because the cardinality of the maximum independent set of the complete graph is 1. Assuming that we have the node i_{k+1}, which belongs to level $k+1$ and has children, in the receiver set of the transmission v_1 (otherwise, it is easier to say that we will not have a claw.), we have another complete subgraph which contains the transmissions from the node i_{k+1} to its children in level $k+2$. One of these transmissions can be selected for a possible claw, say v_3. Since, it is not possible to find another independent transmission v_4, we have a claw-free conflict graph for the network. ■

An example of a tree network and its conflict graph may be seen below.

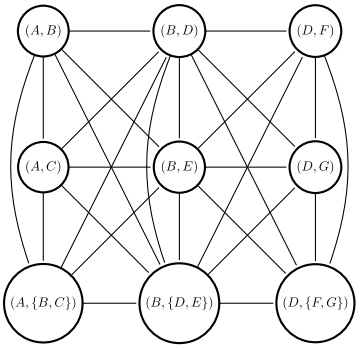


Fig. 6. Conflict graph of the network seen in Fig. 5.

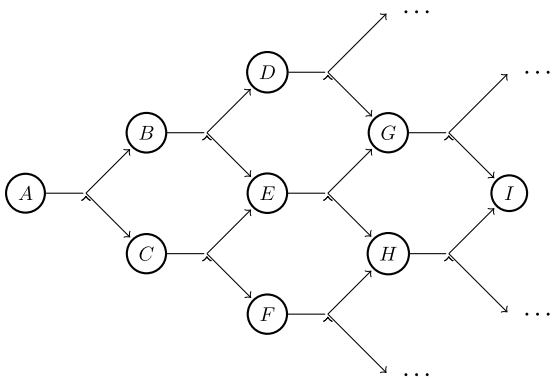


Fig. 7. An example of a diamond network that leads to a claw-free conflict graph under the modified assumptions of Scenario II.

Scenario II can be changed in order to relax the topology by letting every node have children as follows. The transmission and interference schemes are less restricted: We only allow for full-duplex transceivers and assume interference to nodes which are 2 hops away is not possible. Then, $\{v_1, v_2\} \in \mathcal{E}$ if any of the conditions below holds:

C2.3.1 $i_1 = i_2$.
C2.3.2 $J_1 \cap J_2 \neq \emptyset$.

Within these assumptions, we can also introduce a family of networks called diamond networks which are claw-free except having a tree topology. Diamond networks are scalable dense networks which exploit directed antennas. A typical diamond network may be seen in Fig. 7. In a diamond network, every transceiver can transmit to at most two transceivers and can receive from at most two transceivers.

III. Reaching Claw-Freeness in General Networks

After some trials to see when we get a reasonable number of claws under different sets of conditions, it turns out that the following assumptions are both feasible and beneficial to be used in simulations:

- We use the Protocol model, therefore inequalities (1) and (2) are still valid.
- A transceiver can listen to at most one transceiver at the same time.
- Full duplex transceivers are used where we ignore self-interference.

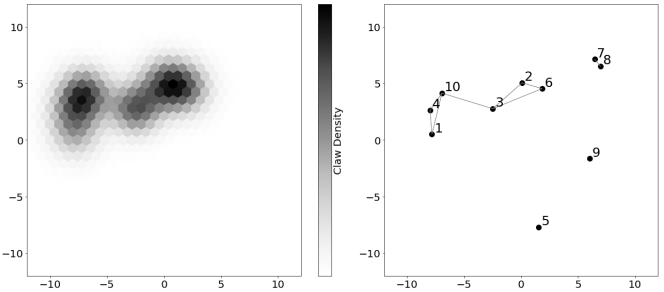


Fig. 8. Left figure shows claw density represented with a heat map. Right figure shows the network which consists of transceivers and their connections.

- 60-degree directional antennas are deployed with all having the same principal direction.

Under these assumptions, the conditions for building the conflict graph change. Denote the physical locations of each node S_i as (x_i, y_i) with the principal direction of the antennas being the x-axis without loss of generality. Say $S_1, S_2 \in \mathcal{V}$, then $\{S_1, S_2\} \in \mathcal{E}$ if any of the following conditions holds for $S_1 = (i_1, J_1)$ and $S_2 = (i_2, J_2)$, which means they cannot be scheduled for the same time slot:

C3.1 $i_1 = i_2$.
C3.2 $J_1 \cap J_2 \neq \emptyset$.
C3.3 $\exists j \in J_1$, $|i_2 - j| \leq (1 + \Delta)|i_1 - j|$ and $\left|\arctan\left(\frac{y_j - y_{i_2}}{x_j - x_{i_2}}\right)\right| < \frac{\pi}{6}$.
C3.4 $\exists j \in J_2$, $|i_1 - j| \leq (1 + \Delta)|i_2 - j|$ and $\left|\arctan\left(\frac{y_j - y_{i_1}}{x_j - x_{i_1}}\right)\right| < \frac{\pi}{6}$.

To assess introduction of claws into the conflict graph, we set up a 2-D coordinate system and randomly assign coordinates to n transceivers in an area of n^2 where we set $n = 10$. Simulations are repeated 100 times for each r_T value and averaged in the end. The results may be seen in Table I. We observe that we almost always get an unconnected network in cases that we have very low number of claws on average as seen for $r_T = 7, 8, 9$. According to the varying transmission range r_T, we observe a rapid increase in the number of claws when connected networks start to appear for $r_T = 10$. Note that, the intersection ratio of connectedness and claw-freeness increases when r_T increases, although the claw-freeness ratio decreases. On the other hand, the average number of claws appearing in a conflict graph increases with r_T. Obviously, there is an important trade-off between connectedness and claw-freeness. We observe that, in a scenario where transceivers have a low transmission range r_T, we usually get a claw-free conflict graph in the expense of the connectedness of a given network. In case of a relatively higher transmission range r_T, we can easily get a connected network, but while also having plenty of claws in the conflict graph. In such a case, it may seem impractical to get rid of a high number of claws without having much effect on MWIS. However, even one action like a very small position change or an introduction of a conflict edge to the conflict graph can break tens of claws and lead to satisfying results as it will also be seen in Section V.

TABLE I

SIMULATION RESULTS FOR RANDOMLY GENERATED TRANSCEIVER LOCATIONS ($n = 10$). RATIO OF INTERSECTION BETWEEN CONNECTEDNESS AND CLAW-FREENESS IS $r_T = 10 \rightarrow 0.01$, $r_T = 11 \rightarrow 0.01$, $r_T = 12 \rightarrow 0.03$, $r_T = 13 \rightarrow 0.05$, $r_T = 14 \rightarrow 0.07$

Transmission range	Average number of claws in the conflict graph	Connectedness of the network	Claw-freeness of the conflict graph	Average number of transmissions in the network
$r_T = 7$	0.19	Unconnected	Claw-free in 96% of trials	5.47
$r_T = 8$	0.99	Connected in 1% of trials	Claw-free in 95% of trials	6.89
$r_T = 9$	3.13	Connected in 1% of trials	Claw-free in 84% of trials	11.43
$r_T = 10$	15.61	Connected in 4% of trials	Claw-free in 83% of trials	11.38
$r_T = 11$	24.44	Connected in 4% of trials	Claw-free in 68% of trials	15.69
$r_T = 12$	15.75	Connected in 10% of trials	Claw-free in 68% of trials	18.49
$r_T = 13$	23.30	Connected in 17% of trials	Claw-free in 56% of trials	23.23
$r_T = 14$	39.43	Connected in 19% of trials	Claw-free in 53% of trials	25.39

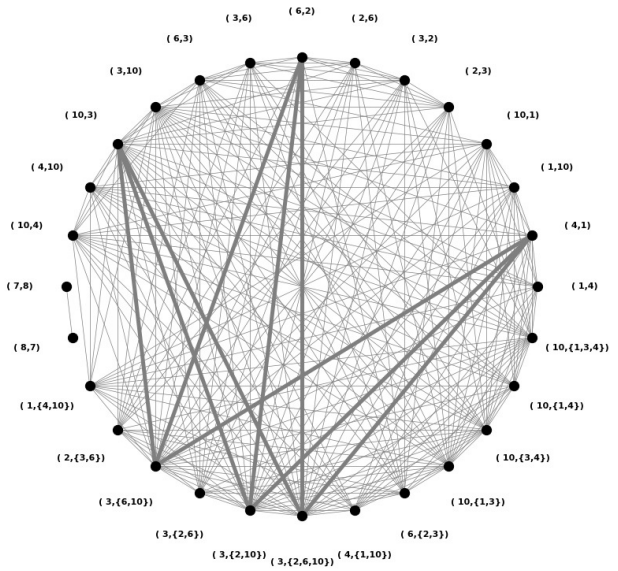


Fig. 9. Conflict graph of the network given in Fig. 8.

For the sake of this example, let us simplify our assumptions. Here, the network has $n = 10$ transceivers as seen in Fig. 8. In this illustration, to better visualize the claw-breaking behavior, we deploy omnidirectional antennas with time division duplex transceivers. In the right part of Fig. 8, the locations of the transceivers are shown in a 2-D coordinate system. An edge between two transceivers i_a and i_b means that $|P_{i_a} - P_{i_b}| \leq r_T$, where P_{i_a} denotes the position of transceiver i_a in the coordinate system. As seen, we do not have a path between every pair of transceivers, so we have an unconnected network in this example. A claw has 4 different vertices which can be represented with (i_t, J_t). In the left figure, the location based claw density of the network is represented as a heat map.

In Fig. 9, we provide the conflict graph of the network seen in Fig. 8. In this conflict graph, we have 28 possible transmissions and 3 different induced claws. We assume that, if there are M pairwise independent vertices connected to a central vertex, there are $\binom{M}{3}$ different claws in this induced subgraph. The induced claws may be seen in Fig. 9 with bold lines.

We can conclude that a small change in the conflict graph may determine whether the said graph has no claws or many of them. Since our purpose is to make a given ad hoc network suitable for polynomial time scheduling, we want the network's conflict graph to be claw-free. What can we do to achieve this goal? Directly breaking claws on the conflict graph by adding edges with the aim that the change in the graph is minimal. Our algorithm is detailed in the next section. Note that, this algorithm is independent of the set of assumptions previously made on the network setup, so that the proposed algorithm can be used for all kinds of conflict graphs.

Before continuing with the proposed method, we remind the reader that we consider the conflict graphs with weighted vertices, even though in our examples, we have simplified weights. Notice that a number of things could have been translated as weights in the conflict graph like the number of receivers in a broadcast, the importance of a particular broadcast and so on. Thus, it is of utmost importance to design our algorithm with the flexibility for an arbitrary transmission weighting scheme.

IV. BREAKING CLAWS ON THE CONFLICT GRAPH

In this section, we propose an approach that modifies the conflict graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, without making any changes in the network configuration. What we do is to add edges to the conflict graph to make it claw-free. In other words, we pretend that some pair of transmissions, say $S_x = (i_x, J_x)$ and $S_y = (i_y, J_y)$, cannot be scheduled for the same time slot even if they do not interfere. So, we modify the conflict graph such that $\{S_x, S_y\} \in \mathcal{E}$ where $S_x, S_y \in \mathcal{V}$.

Given a conflict graph $G = (V, E)$, we denote the set of all missing edges as $\tilde{E}$. We borrow the intuition behind the steepest descent optimization and construct the following greedy algorithm. Starting with G, at each iteration, we identify the edge $e \in \tilde{E}$ such that the action $E \leftarrow E \cup \{e\}$ causes a decrease in the quantity of G's claws as much as possible per decrease in the G's MWIS weighted size. If there exists more than one such edge e, we sample amongst them randomly and conclude with $E \leftarrow E \cup \{e\}$, $\tilde{E} \leftarrow \tilde{E} \backslash \{e\}$. We continue until all the claws in G are eliminated.

A. Substitute for MWIS Weighted Size

The proposed algorithm requires calculating how addition of an edge decreases the MWIS weighted size. However, to our knowledge, the only way is to identify the MWIS both before and after the addition of said edge. Since the motivation of

the algorithm is to avoid the actual calculation of the MWIS in non-claw-free graphs, we need to utilize a meaningful substitute.

Our goal is to eliminate the number of claws with few edge introductions. Hence, we assume that the structure of the graph, and thus the histogram of the maximal independent set weighted sizes, largely maintain their initial form. Through this thinking process, we propose to substitute the expected maximal independent set weighted size, which we denote as S_I, for the actual MWIS weight. Note that this expected value is also a lower bound for MWIS weight. Next, we explain how to compute this to some extent, i.e. a further lower bound.

Procedure 1 Generation of a Maximal Independent Set via an Ordered Vertex Set

Input : $U = \{u_1, \ldots, u_N\}$
Output: A Maximal Independent Set S_U

Initialize $S_U = \emptyset$.
for $i = 1, \ldots, N$ **do**
  if u_i *is not connected to any* $s \in S_U$ **then**
    $S_U \leftarrow S_U \cup \{u_i\}$
  end
end
Output S_U.

First, we identify a given ordering of the vertex set V as $U = \{u_1, \ldots, u_N\}$ where $N = |V|$. Then, Procedure 1 outputs a maximal set S_U heavily dependent on the ordering provided by U. Finally, we calculate $S_I = \mathbf{E}_U[|S_U|_w]$. Here, the expectation is uniformly over all possible Us for a given vertex set V, and the set size operation $|\cdot|_w$ accounts for the vertex weights. This makes S_I the expected maximal independent set weighted size. Note that, Procedure 1 is never actually realized, it only serves a purpose as the insight into the computation of S_I.

Another way to write S_I is as follows:

$$S_I = \mathbf{E}_U\left[|S_U|_w\right] = \mathbf{E}_U\left[\sum_{v\in V} w(v)\cdot \mathbb{1}_{v\in S_U}\right] = \sum_{v\in V} w(v)\cdot \mathbf{E}_U\left[\mathbb{1}_{v\in S_U}\right] = \sum_{v\in V} w(v)\cdot \mathbf{P}\left(v\in S_U\right), \quad (3)$$

where $w : V \to \mathbb{R}$ is the weight mapping for the vertices, and $\mathbb{1}_{v\in S_U}$ is the indicator function for the random event of vertex v being included in S_U, i.e. $\mathbb{1}_{v\in S_U} = 1$ if $v \in S_U$, and $\mathbb{1}_{v\in S_U} = 0$ otherwise. The randomness originates from choosing a random permutation U of the set V to generate S_U. Lastly, the expected value of an indicator function becomes the probability of occurrence for the corresponding event.

To solve for $\mathbf{P}(v \in S_U)$, we define the neighborhood of v as N_v such that each $u \in N_v$ shares an edge with v (no loops, i.e., v does not share an edge with itself). We also define the degree of vertex v as $d_v = |N_v|$.

Let us consider the indexing function $i_u : V \to \{1, \ldots, |V|\}$ for an ordered vertex set U. $i_u(\cdot)$ is such that $v = u_{i_u(v)} \in U$. Given the set U, the vertex v is sure to be chosen for S_U if $i_u(v) < i_u(v')$, for every $v' \in N_v$. For the other placements of v in U, the effect on S_I is not trivial to decompose in terms of the vertex weights and degrees. Consequently, we neglect such additional components contributing to S_I.

Since we have considered all possible Us as equally probable, the probability of $i_u(v) < i_u(v')$ for every $v' \in N_v$ is $1/(d_v + 1)$. Consequently, using Eq. 3,

$$S_I \geq \sum_{v\in V} \frac{w(v)}{d_v + 1}. \quad (4)$$

In Eq. 4, the lower-bound on right-hand side is actually a vertex weighted version of the Caro-Wei bound which is, to our knowledge, the unique quantity amongst both lower and upper bounds for the maximum independent set size easily decomposable into individual contributions from the vertices. Consequently, we conclude this part by setting the following expected maximum independent set weighted size contributions S_v,

$$S_v = \frac{w(v)}{d_v + 1} \quad (5)$$

B. Claw-Freeing Algorithm

Returning to our claw-freeing algorithm, for an edge $e \in \tilde{E}$, we identify its endpoints v and v', i.e. $e = (v, v')$. Let us denote the expected maximal independent set weighted size contributions after the operation $E \leftarrow E \cup \{e\}$, $\tilde{E} \leftarrow \tilde{E}\backslash\{e\}$ as S_v^+ for each $v \in V$. Let us denote the total change from $\sum_{v\in V} S_v$ to $\sum_{v\in V} S_v^+$ as M_e, such that,

$$\begin{aligned} M_e &= \sum_{v\in V} S_v - \sum_{v\in V} S_v^+ = \left(S_v - S_v^+\right) + \left(S_{v'} - S_{v'}^+\right) \\ &= \frac{w(v)}{d_v+1} - \frac{w(v)}{d_v+2} + \frac{w(v')}{d_{v'}+1} - \frac{w(v')}{d_{v'}+2} \\ &= \frac{w(v)}{(d_v+1)(d_v+2)} + \frac{w(v')}{(d_{v'}+1)(d_{v'}+2)} \end{aligned} \quad (6)$$

Let us denote the claw counting function as $C(\cdot)$ where, for a graph G, $C(G)$ is the number of distinct claws present in the said graph.

As a reminder, given 4 vertices from the vertex set V of the graph G, we claim they induce a claw only if their induced sub-graph is a $K_{1,3}$ complete bipartite graph.

Now, we define the quantity Δ_e as the decrease in the quantity of claws after addition of $\{e\}$ into the graph $G = (V, E)$, i.e.,

$$\Delta_e = C((V, E)) - C((V, E \cup \{e\})). \quad (7)$$

Consequently, since we would like to add the missing edge $e \in \tilde{E}$, maximizing the decrease in claw per reduction in the expected maximal independent set weighted size, we choose e as follows, similar with the approach of the steepest descent algorithm.

$$e = \underset{e'\in\tilde{E}}{\operatorname{argmax}} \frac{\Delta_{e'}}{M_{e'}}. \quad (8)$$

Assuming strictly positive vertex weights, for the special case when there is no $e \in \tilde{E}$ such that $\Delta_e > 0$, then it means whichever edge we choose to add, our claw count will not

decrease. In that case, we choose edge e which eliminates the highest number of claws currently on graph $G = (V, E)$ even though it introduces more claws than it erases. This strategy is analogous to local optima escape tactics employed in iterative optimization problems.

C. Computational Cost Analysis of the Algorithm

Throughout the claw-freeing algorithm, the quantities we need to keep track of are d_v, S_v for each vertex $v \in V$, and M_e, Δ_e for each missing edge $e \in \tilde{E}$.

We calculate d_v for every $v \in V$ in $O(\sum_{v \in V} \ (1 + d_v))$ computational time. In terms of the total number of nodes $N = |V|$, this becomes $O(N^2)$. After the introduction of an edge $e = (v_1, v_2)$, we only increment d_{v_1} and d_{v_2}. Since we can at most introduce $O(N^2)$ new edges, the total computational time spent on d_v calculation is still $O(N^2)$.

Calculation of S_v, for all $v \in V$, requires $O(N)$ time overall after the calculation of all d_v. Like d_v, the computation time per edge introduction is constant for updating all S_v, thus, the total computational time spent on all S_v is also $O(N^2)$.

After calculating every d_v and S_v, the calculation of all M_e, for $e \in \tilde{E}$, is $O(N^2)$. However, unlike before, updating M_e quantities has computational cost $O(N - d_{v_1} + N - d_{v_2})$ after the introduction of $e = (v_1, v_2)$ since we need to update M_e for every $e \in \tilde{E}$ and the number of neighbors vertex v has in the complement graph $\tilde{G} = (V, \tilde{E})$ is $(N - d_v)$. As a result, the overall computational cost of calculating every M_e is $O(N^3)$.

Calculation of the quantities Δ_e for each missing edge $e \in \tilde{E}$ requires identifying each unique claw in the graph. Hence, the computational cost is $O(\#Claws)$. As we have observed, the claw, i.e., $K_{1,3}$, is a three-pronged structure where we have a ternary tree of 4 nodes with 1 parent (root) and 3 children. For every node v, the identification of a $K_{1,3}$ can be achieved in $O({d_v}^3)$ time. Consequently, the overall time is $O(\sum_{v \in V} \ {d_v}^3)$.

Furthermore, the calculation of Δ_e also requires identifying unique instances of another structure we shall call a "pre-claw". It occurs when out of 4 vertices, 3 form the two-pronged version of a claw, $K_{1,2}$, while the other is disconnected from the first three. The computational cost of finding the pre-claws is $O(\sum_{v \in V} \ {d_v}^2 |V|)$. The difference of changing one d_v multiplier with $|V| = N$ in comparison to distinct claw identification results from the fact that after identifying a $K_{1,2}$, we also need to find a fourth vertex disconnected from the first three.

In terms of the total number of nodes N, the overall computational time -claw and pre-claw identifications combined- is at worst $O(N^4)$. In the worst-case, we may add $O(N^2)$ edges to eliminate the claws resulting in the overall computational time of $O(N^6)$ for obtaining a claw-free graph.

The computational cost attributed to d_v, S_v and M_e together is $O(N^3)$. However, the cost attributed to Δ_e is $O(N^6)$. Since the initial calculation cost of Δ_e is also larger than the cost of d_v, S_v and M_e with $O(N^4)$, we conclude that the overall computational cost of the algorithm is dominated by the cost of computing Δ_e and currently is $O(N^6)$.

Algorithm 2 Claw-Freeing Algorithm - Initialization

Input : Initial Conflict Graph $G = (V, E)$
Weight Mapping $w : V \to \mathbb{R}$
Output: Claw-Free Graph G_c

Set $N = |V|$.
Initialize $\Delta_e = 0$ for every $e \in \tilde{E}$.
Initialize the number of claws $C = 0$.
Initialize claw elimination counts $\Delta_e^* = 0$.
NOTE: (u, v) and (v, u) are the same undirected edge.
foreach $v \in V$ **do**
  Identify neighbor set N_v.
  Calculate vertex degree $d_v = |N_v|$.
  Calculate size contribution $S_v = w(v)/(d_v + 1)$.
  foreach $\{u_1, u_2, u_3\} \subset N_v$ **do**
    Set $E_c = \{(u_1, u_2), (u_1, u_3), (u_2, u_3)\}$.
    if $\forall e \in E_c$, $e \notin E$ *(i.e., a claw)* **then**
      foreach $e \in E_c$ **do**
        $\Delta_e \leftarrow \Delta_e + 1$.
        $\Delta_e^* \leftarrow \Delta_e^* + 1$.
      $C \leftarrow C + 1$.
  foreach $\{u_1, u_2\} \subset N_v$ **do**
    if $(u_1, u_2) \notin E$ **then**
      foreach $v_2 \in V \backslash (N_v \cup N_{u_1} \cup N_{u_2})$ **do**
        $\Delta_e \leftarrow \Delta_e - 1$, for $e = (v, v_2)$.
foreach $\{v, v'\} \subset V$ *s.t.* $(v, v') \notin E$ **do**
  $e = (v, v')$.
  $M_e = S_v/(d_v + 2) + S_{v'}/(d_{v'} + 2)$.

Despite being polynomial, this computational time is higher than the cost of state-of-the-art algorithms for finding the MWIS in claw-free graphs. The pseudo-code for this part of the initial calculations may be found in Algorithm 2. Our claw-freeing algorithm introduces a substantial bottleneck if Δ_e is calculated from scratch after each edge introduction. Therefore, we propose the following approach to efficiently calculate Δ_e.

D. Iterative Calculation of Δ_e

Introduction of a new edge $e \in \tilde{E}$, eliminates existing claws and introduces new claws only if the two out of four vertices involved in the eliminated or introduced claws are the endpoints of our new edge e. Therefore, it should be possible to reduce the per new edge computational cost of updating M_e to $O(N^2)$, effectively resulting in a total update cost of $O(N^4)$. Thus, even including the initial calculation cost of $O(N^4)$, the overall computational cost becomes $O(N^4)$. We will now detail how this can be achieved.

At each iteration of the edge introduction algorithm, we need to identify unique instances conforming to one of the following five types. Consider the newly introduced edge as $e = (v_1, v_2)$. After the identification of these instances, we follow with the provided update for Δ_e.

1) Type 1 - Claw Before: Before the introduction of edge e, we had a claw such that v_1 and v_2 were two of the three children. This means, we had a root u_1 which was a neighbor of both v_1 and v_2. Furthermore, we had another child u_2 which was a neighbor of u_1 but not v_1 and v_2.

After introducing $e = (v_1, v_2) \in \tilde{E}$, the claw is broken. Hence, $\Delta_{e'}$ for the missing edges $e' = (v_1, u_2)$ and $e' = (v_2, u_2)$ are both decremented by 1.

2) Type 2 - Claw After: Before the edge e, we had a pre-claw $K_{1,2}$ such that u_1 and u_2 are the children of $K_{1,2}$ while either v_1 or v_2 is the parent. This means u_1 and u_2 are disconnected, and only one of the endpoints of e is neighbors with both, while the remaining endpoint is disconnected from the other three.

After adding edge $e = (v_1, v_2)$, a new claw is formed. We determine which of the endpoints (v_1 or v_2) is the root (parent) and denote the other as v_*. Afterwards, $\Delta_{e'}$ for $e' = (u_1, v_*)$, $e' = (u_2, v_*)$ and $e' = (u_1, u_2)$ are incremented by 1.

3) Type 3 - Pre-Claw Before With Children Endpoints: Before the edge e, we had a pre-claw $K_{1,2}$ where v_1 and v_2 are the children, while u_1 is the parent. Note, $u_1 \in N_{v_1}$ and $u_1 \in N_{v_2}$. We also had u_2 which is disconnected with the other three.

After the edge e, the pre-claw is eliminated as v_1, v_2, and u_1 form a triangle. Therefore, $e' = (u_1, u_2) \in \tilde{E}$ no longer introduces a new claw. Hence, $\Delta_{e'}$ is incremented by 1 to neutralize a previous reduction for claw introduction for when e' are to be introduced.

4) Type 4 - Pre-Claw Before With a Non-Child Endpoint: Before the edge e, we had a pre-claw $K_{1,2}$ where one of the children is v_1 or v_2, and the parent is u_1. The remaining endpoint, temporarily denoted by v_*, is disconnected from all the previous vertices involved in $K_{1,2}$.

After the addition of edge e, the pre-claw is eliminated as the 4 vertices currently form a path. Thus, $e' = (u_1, v_*) \in \tilde{E}$ no longer introduces a new claw, and $\Delta_{e'}$ is incremented by 1 as in Type 3.

5) Type 5 - Pre-Claw After: Before the edge e, we have a structure consisting of 4 vertices $\{v_1, v_2, u_1, u_2\}$ and 1 edge which connects one of the endpoints, temporarily denoted as v_*, with one of its neighbors u_1. u_2 is disconnected from the other three, similarly for the remaining endpoint.

After the edge e, we obtain a pre-claw with v_* at the root. Thus, $\Delta_{e'}$ for $e' = (v_*, u_2)$ is decremented by 1.

To sum up, the iterative calculation of Δ_e decreases the computational cost of calculating Δ_e from $O(N^6)$ to $O(N^4)$. This, in turn, reduces the overall computational cost of the claw-freeing algorithm to $O(N^4)$ since the calculation cost of Δ_e still dominates. The pseudo-code for the general run-time is displayed in Algorithm 3.

Algorithm 3 Claw-Freeing Algorithm - Run-Time

while $C > 0$ **do**
  $e = \text{argmax}_{e' \in \tilde{E}} \Delta_{e'}/M_{e'}$.
  if $\Delta_e \leq 0$ **then**
    $e = \text{argmax}_{e' \in \tilde{E}} \Delta^*_{e'}$.
  $E \leftarrow E \cup \{e\}$. $C \leftarrow C - \Delta_e$.
  Identify endpoints $(v, v') = e$.
  foreach $v^* \in \{v, v'\}$ **do**
    $d_{v^*} \leftarrow d_{v^*} + 1$.
    $S_{v^*} \leftarrow S_{v^*} \cdot d_{v^*}/(d_{v^*} + 1)$.
    foreach $u \in V \backslash \{v^*\}$ *s.t.* $(v^*, u) \notin E$ **do**
      $M_{e'} = S_{v^*}/(d_{v^*} + 2) + S_u/(d_u + 2)$, for $e' = (v^*, u)$.
  foreach $u' \in V \backslash (N_v \cup N_{v'})$ **do**
    foreach $u \in N_v \cap N_{v'}$ **do**
      if $(u, u') \in E$ **then**
        $\Delta_e \leftarrow \Delta_e - 1$ for $e \in \{(v, u'), (v', u')\}$.
        $\Delta^*_e \leftarrow \Delta^*_e - 1$ for $e \in \{(v, u'), (v', u')\}$.
      else
        $\Delta_e \leftarrow \Delta_e + 1$ for $e = (u, u')$.
    foreach $\{v^*, v^-\} \in \{\{v, v'\}, \{v', v\}\}$ **do**
      foreach $u \in N_{v^*} \backslash N_{v^-}$ **do**
        if $(u, u') \in E$ **then**
          $\Delta_e \leftarrow \Delta_e + 1$ for $e = (v^-, u)$.
        else
          $\Delta_e \leftarrow \Delta_e - 1$ for $e = (v^*, u')$.
  foreach $\{v^*, v^-\} \in \{\{v, v'\}, \{v', v\}\}$ **do**
    foreach $\{u, u'\} \subset N_{v^*} \backslash N_{v^-}$ **do**
      if $(u, u') \notin E$ **then**
        $\Delta_e \leftarrow \Delta_e + 1$ for $e \in \{(u, u'), (v^-, u), (v^-, u')\}$
        $\Delta^*_e \leftarrow \Delta^*_e + 1$ for $e \in \{(u, u'), (v^-, u), (v^-, u')\}$
Output $G_c = (V, E)$.

V. Simulation Results

We conduct simulations over randomly located n transceivers in a 2-D coordinate system with an area of 20×20. We assign the weights of the vertices in the conflict graph as the number of the receivers in the transmission. Thus, the weight of the resulting independent set and network throughput are proportional, and we consider the MWIS weight as the maximum network throughput in the figures. We compare the performance of claw breaking to the optimal performance and to the maximal set scheduling method which is a commonly used method where the maximum independent set is required and cannot be found due to its NP-hard nature. To find a maximal independent set, we initially assign all vertices to an undecided state and set maximal independent set as the empty set, $MIS = \emptyset$. From the conflict graph, we choose an undecided vertex v and control its neighbors $N(v)$. If none of its neighbors are in our set, we add this vertex to our independent set, $v \in MIS$, and assign all of its neighbors to an unavailable state. This way, we are able to get a maximal independent set in the conflict graph in linear time [7]. Also, to set a more meaningful benchmark for comparison, we repeat maximal independent set scheduling until this new polynomial time method matches the complexity

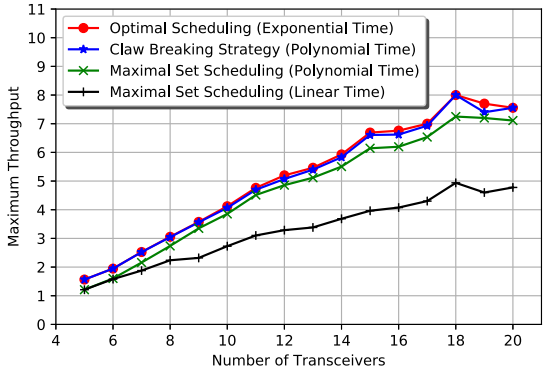


Fig. 10. Performance analysis of Claw Breaking Strategy with respect to number of transceivers in the network. ($r_T = 8, 9, 10, 11$).

of the claw breaking strategy, i.e. both maximal set scheduling and claw breaking method runs in $O(N^4)$ where N is the number of vertices in the conflict graph.

Before continuing with the results, we remind the reader that our algorithm is capable of working with any type of conflict graph. Nevertheless, we keep the specifications of the network as described earlier, in Section III.

We may see the change in MWIS with respect to the number of transceivers in the network in Fig. 10. For $n < 15$, claw breaking performs almost optimally, and actually, this is because we do not observe a high number of claws for a low number of transceivers. As a reminder, we have 11 claws on average for $n = 10$ and $r_T = 8, 9, 10, 11$ as seen in Table I. However, even for $n > 15$, where we have a denser network and observe a higher number of claws, our strategy performs very close to optimal one and clearly outperforms the maximal independent set scheduling which has the same complexity with the proposed claw breaking method. In average, from $n = 5$ to $n = 20$, claw breaking method achieves 8.3% more throughput than the polynomial time maximal independent set scheduling benchmark. In addition to this satisfying result, claw breaking strategy is only, on average, 1% away from the optimal throughput which can only be found in exponential time.

Then, we fix the number of the transceivers to $n = 10$, and we make random choices among $r_T = 10, 11, 12, 13$ in every iteration to get enough samples of conflict graphs with claws between 1 and 200. The performance evaluation may be seen in Fig. 11a. As expected, while performing very close to optimal when the number of claws is nearly zero, we observe a slight decrease in the throughput performance with increasing number of claws. Interestingly, this decrease does not prevail with the increasing number of claws and the proposed method continues to provide satisfying results. To quantify, claw breaking performs 2.4% better than polynomial time maximal set scheduling and it is 2.6% away from being optimal. For $n = 10$, we observe that our method is approximately, on average, in the middle of the polynomial time benchmark and the optimal output in terms of the throughput performance. The average number of edges introduced to reach claw-freeness may be seen in Fig. 11c. Since we are able to break hundreds of claws by just introducing a few edges in the conflict graph, our strategy also performs well for connected networks which have higher numbers of claws.

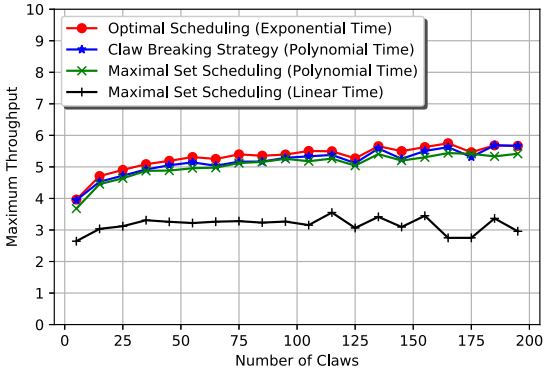
As a next step, we increase the number of the transceivers and fix it to $n = 15$ and make random choice among $r_T = 8, 9, 10, 11$ in every iteration. This time, we evaluate the cases with claws until 400. Above this value, we do not get enough samples since the networks in this setup do not tend to have more than 400 claws in their conflict graphs. The performance of our strategy may be seen in Fig. 11b. We observe that the margin between claw breaking and optimal performance is higher than the case in Fig. 11a. In other words, claw breaking method performs 3.8% inferior to optimal result, on the other hand, it performs 3.1% superior to maximal set benchmark. Again, it can be said to be approximately in the middle way of the optimal performance and the benchmark we set. Besides, the connectedness ratio is significantly lower in this case. This is an expected result since we have a higher number of transceivers, and therefore, we decrease r_T not to have an unnecessarily high number of possible transmissions in the system. The decrease in the connectedness ratio is due to the randomness of the transceiver locations and can be compensated by a different placing method, for instance, random placement on a grid. The number of edges introduced in order to get rid of claws may be seen in Fig. 11d. We observe from Fig. 11c and Fig. 11d that the number of edges introduced for claw-freeing behaves as a concave function of claw number which motivates us to use the strategy in networks with high numbers of claws.

Finally, we have a set of simulations in Fig. 12 investigating the throughput performance with respect to the average number of receivers that a transceiver have in the network. Here, we evaluate the proposed strategy in terms of a network property instead of its conflict graph's property as we have done in Fig. 11. This highlights the performance of the algorithm clearer since it is easier to setup a network by directly using its property than by approximating the properties of its conflict graph. For both cases, our strategy performs very close to optimal, having only 0.6% ($n = 10$) and 1% ($n = 15$) lower output on average, and outperforms the polynomial time maximal set scheduling benchmark by 4% ($n = 10$) and 4.9% ($n = 15$).

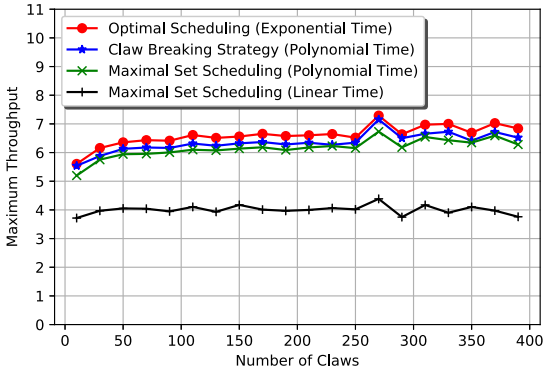
In overall, as seen from the simulation results, claw breaking strategy is a very strong candidate for scheduling in wireless ad hoc networks to get better results in terms of sum throughput.

VI. Discussion and Future Directions

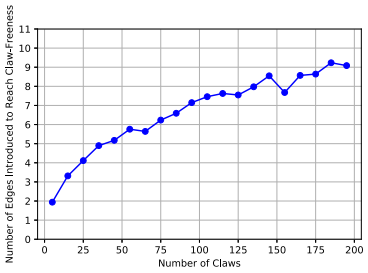
From the experiments, we observe that our claw breaking strategy performs very well for various numbers of transceivers, up to a limited number of connections with directed antennas. In the experiments, we opted to utilize directed antennas instead of omnidirectional ones for primarily the following reasons. First, the deployment of omnidirectional antennas almost always increases the number of neighbors a transceiver has, which in turn increases the tendency of a network to break the limited receiver set rule of $|N(i)| \leq K$ for each transmitter i. Secondly, with omnidirectional antennas, as the transmission range of the transceivers, i.e. the number



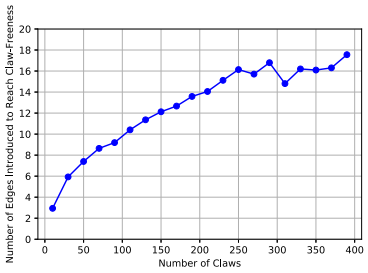
(a) $n = 10$ and $r_T = random(10, 11, 12, 13)$



(b) $n = 15$ and $r_T = random(8, 9, 10, 11)$

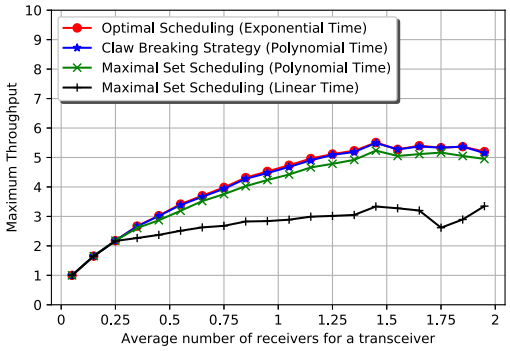


(c) $n = 10$ and $r_T = random(10, 11, 12, 13)$

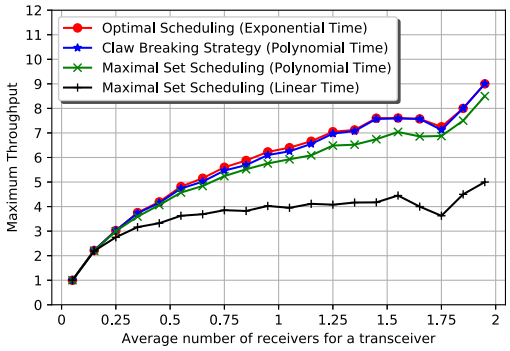


(d) $n = 15$ and $r_T = random(8, 9, 10, 11)$

Fig. 11. Performance analysis of Claw Breaking Strategy under the assumptions given in Section V.



(a) Performance comparison of Claw Breaking Strategy and Maximal Set Scheduling. $n = 10$.



(b) Performance comparison of Claw Breaking Strategy and Maximal Set Scheduling. $n = 15$.

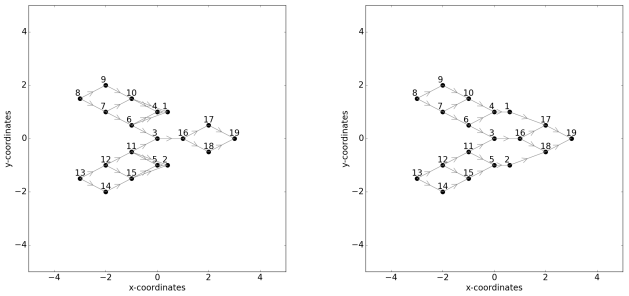
Fig. 12. Performance comparison of Claw Breaking Strategy with Maximal Set Scheduling under the assumptions of Sec. V.

of receivers they can connect, increases, the number of claws in the conflict graph shows a very rapid increase in response and becomes very high. Thus, the choice of deploying directed antennas is crucial both for construction of the conflict graph and for the better performance of the claw breaking strategy.

Besides the antenna choice, there may also be other additional possibilities for getting claw-freeness -or at least low numbers of claws- in the conflict graph. An idea is to propose some physical modifications which include position adjustment, transmission range adjustment and/or antenna orientation adjustment in the network configuration. These possible interventions may lead to changes in the set of possible transmissions and interference between them.

Let us consider the network given in Fig. 13a. We observe transceivers as the black dots and possible transmissions as the directed gray edges. The transmission range r_T is assumed to be equal for all transceivers. The considered network has 32 claws in its conflict graph with the current connections. By a minor change in the transceiver locations in the 2-D coordinate system, $x_1 = x_1 + 0.2$ and $x_2 = x_2 + 0.2$ where x_i is the x-coordinate of transceiver i in figure, we reach a configuration seen in Fig. 13b. By shifting the locations of two transceivers, we lose transmissions $(6, 1)$, $(10, 1)$, $(6, \{1, 4\})$, $(10, 1, 4)$, $(11, 2)$, $(15, 2)$, $(11, \{2, 5\})$, $(15, \{2, 5\})$, whereas $(1, 17)$ and $(2, 18)$ show up as new possible transmissions in the conflict graph. New configuration has a claw-free conflict graph.

However, making physical modifications in a network is usually not a feasible way if there are strict requirements on the locations of nodes, or there is no possibility or resources to make such changes. Even if there are not such limitations, we are currently unable to offer a practical method that works for all kinds of networks in general. A possible direction for the future work may be to tackle this problem and offer an automated approach.



(a) An example network (b) Fig. 13a with adjustments

Fig. 13. An example of a physical position modification.

On another note, in this study, we solely focus on modifying the conflict graphs to make them claw-free. In a similar manner, it is also possible to consider other types of graphs, such as fork-free graphs, also for which the MWIS problem has polynomial time complexity. Moreover, it may also be worthy to investigate an algorithm that decides on-the-fly to which type of graph it needs to transform the conflict graph to obtain polynomial time scheduling capability in return of the minimum reduction in the weight of the MWIS.

VII. Conclusion

Scheduling in wireless networks is an NP-Hard problem. We show that scheduling can be completed in polynomial time given that the network structure conforms to the set of assumptions presented in Section II. Since these assumptions are limiting factors for the general use of the polynomial time scheduling, we extend the scope of our work to all kinds of networks.

To address scheduling of general wireless broadcast networks when such networks do not conform to the set of rules we present, we provide an algorithm to break the claws on the conflict graph by introducing new edges without any intervention with the network setup. This is obviously a sub-optimal approach, but after proper optimization on the selection of new edges, the decrease in the throughput can be kept very low. From the simulations, we observe that our strategy performs significantly better than the common method for scheduling in wireless networks.

Another possibility to get a claw-free conflict graph is to make physical modifications in the considered network to change the connections and/or interference relationships. A disadvantage of physical modifications is that it does not seem possible to provide an automatized method to implement them. Such modifications may only be done by relating the current network to ones studied in Section II.

References

[1] E. Arikan, "Some complexity results about packet radio networks (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 4, pp. 681–685, Jul. 1984.

[2] A. Ephremides and T. V. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Trans. Commun.*, vol. 38, no. 4, pp. 456–460, Apr. 1990.

[3] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proc. 12th Annu. Int. Conf. Mobile Comput. Netw.*, 2006, pp. 227–238.

[4] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 5, pp. 910–917, Sep. 1988.

[5] D. Traskov, M. Heindlmaier, M. Medard, and R. Koetter, "Scheduling for network-coded multicast," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1479–1488, Oct. 2012.

[6] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 709–720, Jun. 2010.

[7] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar, "Throughput and fairness guarantees through maximal scheduling in wireless networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 2, pp. 572–594, Feb. 2008.

[8] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 3, Mar. 2005, pp. 1804–1814.

[9] V. Aggarwal, A. S. Avestimehr, and A. Sabharwal, "On achieving local view capacity via maximal independent graph scheduling," *IEEE Trans. Inf. Theory*, vol. 57, no. 5, pp. 2711–2729, May 2011.

[10] L. Bao and J. J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for ad hoc networks," in *Proc. 7th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2001, pp. 210–221.

[11] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[12] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[13] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, 2003, vol. 41. no. 1, pp. 11–20.

[14] T. Ho *et al.*, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[15] A. Köse and M. Médard, "Scheduling wireless ad hoc networks in polynomial time using claw-free conflict graphs," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–7.

[16] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[17] J. K. Sundararajan, M. Médard, R. Koetter, and E. Erez, "A systematic approach to network coding problems using conflict graphs," in *Proc. UCSD Workshop Inf. Theory Appl.*, 2006, pp. 1–5.

[18] J. K. Sundararajan, M. Médard, M. Kim, A. Eryilmaz, D. Shah, and R. Koetter, "Network coding in a multicast switch," in *Proc. 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 1145–1153.

[19] G. J. Minty, "On maximal independent sets of vertices in claw-free graphs," *J. Combinat. Theory, B*, vol. 28, no. 3, pp. 284–304, Jun. 1980.

[20] D. Nakamura and A. Tamura, "A revision of minty's algorithm for finding a maximum weight stable set of a claw-free graph," *J. Oper. Res. Soc. Jpn.*, vol. 44, no. 2, pp. 194–204, 2001.

[21] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin, Germany: Springer, 2003.

[22] Y. Faenza, G. Oriolo, and G. Stauffer, "Solving the weighted stable set problem in claw-free graphs via decomposition," *J. ACM*, vol. 61, no. 4, p. 20, 2014.

[23] V. V. Lozin and M. Milanič, "A polynomial algorithm to find an independent set of maximum weight in a fork-free graph," *J. Discrete Algorithms*, vol. 6, no. 4, pp. 595–604, Dec. 2008.

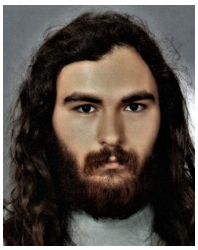
[24] D. Lokshantov, M. Vatshelle, and Y. Villanger, "Independent set in, P_5-free graphs in polynomial time," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2014, pp. 570–581.

[25] M. Chudnovsky and P. D. Seymour, "The structure of claw-free graphs," *Surv. Combinatorics*, vol. 327, pp. 153–171, Jul. 2005.

Alper Köse received the B.Sc. degree (Hons.) in electrical and electronics engineering from Bogazici University, Istanbul, Turkey, in 2015, and the M.Sc. degree in electrical and electronics engineering along with a minor in computer science from EPFL, Lausanne, Switzerland. He completed his M.Sc. thesis at the Research Laboratory for Electronics, MIT, Cambridge, USA, in 2017. He is currently pursuing the Ph.D. degree with Bogazici University, focusing on communications and information theory. He has several research articles in fields of machine learning, communications, and networking.

Hakan Gökcesu received the B.Sc. degree (Hons.) in electrical and electronics engineering from Bilkent University in 2015 and the M.Sc. degree in communication systems with data analytics specialization from the School of Computer and Communication Sciences, EPFL, in 2018. He is continuing his academic studies with Bilkent University, with a focus on online learning. He is also affiliated with Turkcell Technologies as a Telecommunications Engineer, where his work is focused on research and development for 5G and beyond. He was part of many collaborations, with public or private funding, which conducted Research and Development on distributed machine learning, sensor analytics, smart systems, reinforcement learning, prediction and estimation, data science, and wireless communications. He has authored or reviewed many articles in a number of respected academic journals. His research interests include machine learning and optimization with limiting scenarios. His research interests also include signal processing, learning systems, information theory, decision theory, graph theory, big data, and applied mathematics.

Noyan Evirgen received the B.Sc. degree in electrical and electronics engineering from Bilkent University in 2015 and the M.Sc. degree in information technology and electrical engineering from ETH Zurich in 2018, where he focused on computer vision. He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of California at Los Angeles, where he focuses on machine learning and human computer interface. In 2018, he was a Research Intern with the Schlumberger Doll Research Center, Math and Modeling Team, where he developed a novel unsupervised learning algorithm for unconventional reservoirs. His work in Schlumberger is published and patented. He also did a research internship at Apple with the System Engineering and Machine Learning Department with the Scene Understanding Team, where he focused on novel conditional computation and multitask learning algorithms in convolutional neural networks. He has authored nine articles in various subjects, including machine learning, wireless communications, oilfield, and space. His current research interests include explainable artificial intelligence and conditional computation. He received the Academic Excellence Award for his B.Sc. degree.



Kaan Gökcesu received the B.Sc. degree (Hons.) in electrical and electronics engineering and the M.Sc. degree (Hons.) from Bilkent University, Turkey, in 2015 and 2017, respectively, where his thesis focused on online learning and anomaly detection. He is currently pursuing the Ph.D. degree in electrical engineering with MIT, USA. He has founded several companies and research groups, both government and private funded, that focused on machine learning, sensor analytics, smart systems, data analysis, and optimization. He has authored numerous papers in highly respected academic journals and conference proceedings. His research interests include computational learning, optimization, sequential prediction, data science, information theory, decision theory, adaptive filtering, communication systems, statistical signal processing, big data, and mathematical finance.

Muriel Médard (Fellow, IEEE) has co-founded three companies to commercialize network coding, CodeOn, Steinwurf, and Chocolate Cloud. She is currently the Cecil H. Green Professor with the Electrical Engineering and Computer Science (EECS) Department, MIT, where she also leads the Research Laboratory for Electronics, Network Coding, and Reliable Communications Group. She is a member of the National Academy of Engineering and the National Academy of Inventors. She received the 2002 IEEE Leon K. Kirchmayer Prize Paper Award, the 2009 IEEE Communication Society and Information Theory Society Joint Paper Award, the 2009 William R. Bennett Prize in the field of communications networking, and the 2018 ACM SIGCOMM Test of Time Paper Award; and several conference paper awards. She was a co-winner of the MIT 2004 Harold E. Edgerton Faculty Achievement Award, received the 2013 EECS Graduate Student Association Mentor Award and has served as a Housemaster for seven years. In 2007, she was named as a Gilbreth Lecturer by the U.S. National Academy of Engineering. She received the 2016 IEEE Vehicular Technology James Evans Avant Garde Award, the 2017 Aaron Wyner Distinguished Service Award from the IEEE Information Theory Society, and the 2017 IEEE Communications Society Edwin Howard Armstrong Achievement Award. She has served as the Technical Program Committee Co-Chair of many of the major conferences in information theory, communications, and networking. She was the President of the IEEE Information Theory Society in 2012, and has served on its board of governors for 11 years. She has served as an Editor for many publications of the Institute of Electrical and Electronics Engineers (IEEE), of which she was elected as a fellow. She has served as the Editor-in-Chief for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.