International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5, Issue No: 1, September 2019

Complexity Reduction of the Optimal Link Scheduling Algorithm in Wireless Networks through Topology Control

Azham Hussain[1] Mazniha Berahim[2]
[1]School of Computing, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia
[2]Center for Diploma Studies, Universiti Tun Hussein Onn Malaysia.

Abstract: Concurrent transmissions at different links can interfere with each other in single channel wireless networks. A scheduling algorithm is needed to select a subset of links for data transmission to improve system performance. Optimal connection scheduling discipline throughput is usually an NP-hard issue. We use the concept of line graph in this paper and extend it to line multi graph to deal with the complexity issue of the algorithm of maximum weight scheduling (MWS). The required and sufficient conditions are derived in terms of network topology to reduce the complexity of MWS. We prove that eLehot's complexity is polynomial time as long as there are no seven derived prohibited graphs as induced sub graphs in the conflict graph. We also propose an eLehot algorithm to detect if a graph is a line multigraph and to output its root graph. The findings of this paper introduce a new approach to regulation of wireless topology where the aim is to reduce complexity.

Keywords: Wireless network scheduling; line graph; line multigraph; root graph; dispute graph; topology command.

I. INTRODUCTION

Concurrent transmission of simultaneous slot and separate links can interfere with each other in single channel wireless networks. Using graph concepts is a general approach for modeling the network and interference relationship between nodes and edges. An undirected and connected graph tt(V, E) can model an underlying wireless network in which V is the set of vertices and E is the set of edges. Every network node is represented in graph tt by a vertex. If they are within each other's contact distance, two vertices are adjacent. Another graph, called the conflict graph, is introduced to deal with interference. A given graph tt(V, E)'s conflict graph is $tt^c(E, L)$. The vertex in tt^c corresponds to an edge in tt, and if its corresponding edges in tt overlap with edges, two vertices in tt^c are adjacent. We define the notion of interfering with ties in the near future. In this approach, when a link is ready for transmission, it is necessary to consider as interfering links only a subset of links which are called the interference set associated with that link. A link scheduling discipline is required to select a subset of non-interfering links at each time slot for data transmission to mitigate the adverse effects of interference in wireless networks. Note that if l_1 interferes with l_2 then l_2 also interferes with l_1. Finding a set of non-interference links in tt is the same as finding an autonomous set in tt^c. An independent set in a graph is a collection of vertices so that between them there are no edges.

We refer to more terminologies of graph theory that we use throughout the paper in Appendix A. We rephrase Sharma et al. (2006)'s M-hop interference model based on the concept of line graph to achieve a rigorous definition of conflict graph. Two edges l_1 and l_2 are interfering edges under this general interference design ifd(l_1, l_2) M. The conflict diagram can therefore be defined as follows,

$$tt^c(E, L) = [L(tt)]^M, \quad M \geq 1, \tag{1}$$

Where L(tt) is tt's line graph. This interference model applies to a wide range of practical applications like Bluetooth, FH-CDMA, wireless LAN (IEEE 802.11 standard), etc. (Sharma et al., 2006; Yi and Chiang, 2008). In the RTS / CTS scheme IEEE 802.11 wireless LAN network, two edges that are either adjacent or both incidental to a common edge interfere with the edges. The conflict graph can then be constructed using equation (1) and setting M = 2 while the conflict graph is the same as L(tt) in Bluetooth networks, which is derived by setting M = 1 in equation (1).

Connection scheduling algorithms influence the network bandwidth directly. One of the well-known NP-Hard issues (Sharma et al., 2006) is the throughput optimal link scheduling algorithm, called maximum weight scheduling and its counterpart in conflict graph maximum weight independent set (MW IS). Because of the algorithm's high computational complexity, several investigations were carried out to resolve this problem. We are following this line of research in this paper. We note that if the conflict graph is a line graph, finding M W IS in ttc is the same as finding total matching weight (M W M) in the root graph (see Figure 1(a)). Because there are polynomial time complexity algorithms for the MW M problem (Lawler, 2001), this finding makes the overall solution much simpler.

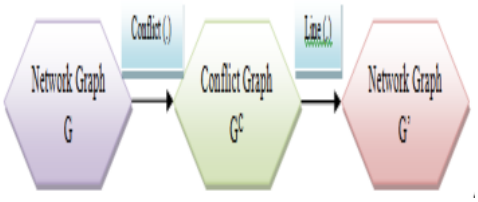
Line graphs are graph category well defined. In Beineke (1968) it is proved that a graph tt is a line graph of a

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5, Issue No: 1, September 2019

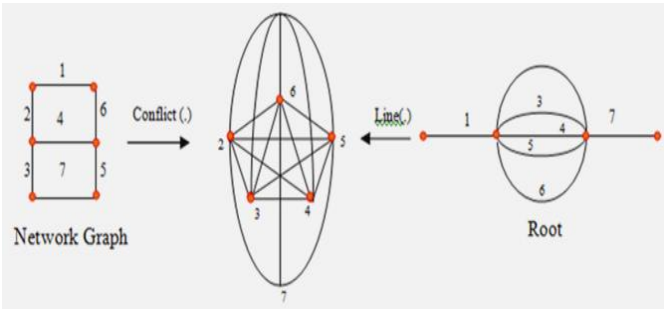
simple graph tt′ if and only if tt does not contain any of the nine graphs forbidden, as shown in Figure 2, as a subgraph induced. An induced graph subgraph is a subset of graph vertices with edges that have both endpoints in this subset. Whitney proved that the structure of tt′ can be fully recovered from its line graph with two exceptional cases (triangle and star with three branches, tt1 in Figure 2) (West, 2000).

It should be noted that Lehot has developed an optimal algorithm that can be executed in linear time to detect whether a graph is line graph and beget its root graph (Lehot, 1974). The algorithm, however, only considers simple graphs as a root graph. The key point that catches our attention here is that there is no need for the root graph to be a simple graph. If the root graph is multigraph, holding the heaviest edge between several edges and eliminating the other edges is necessary before running M W M algorithm.

In this paper we introduce a generalization of line graph to line multigraph, i.e. line graph for which its root graph is multigraph, following our observation that allows the root graph to be multigraph. Instead we extend the Lehot algorithm to line multigraphs and propose an algorithm of low complexity, known as extended Lehot (eLehot), to detect whether a graph is line multigraph and output its root graph. Accordingly, we loosen the restriction shown in Figure 2 by allowing the root graph to be multigraph to seven minimally forbidden graphs. This result is of great importance in the dynamics of wireless networks. Not only is the number of forbidden graphs increasing, it is much easier to avoid them in the topology construction of a standard wireless network as they are smaller in the number of vertices and edges. By increasing or decreasing the nodes ' transmission power, we can easily add / remove edges to / from the conflict graph. Such topics are subject to wireless network topology regulation. The findings of this paper introduce a new approach to wireless network topology control algorithms where the ultimate goal is to reduce complexity. It complements the original motivation of topology control disciplines which attempted to reduce energy consumption while maintaining network graph connectivity (Santi, 2005; Wang, 2008). Then, the results of this paper can add a new design dimension to topology control algorithms. As a consequence, based on available polynomial time complexity algorithms for M W M problem (Lawler, 2001) and due to the Lehot algorithm's linear time complexity (Lehot, 1974), we are designing a polynomial time complexity approach for the general M-hop interference model for the graph category that their conflict graphs are line multigraphs.



(a)



(b)

Figure 1. Main concept illustration: (a) network, conflict and root graph relationship and (b) example (see online color version)

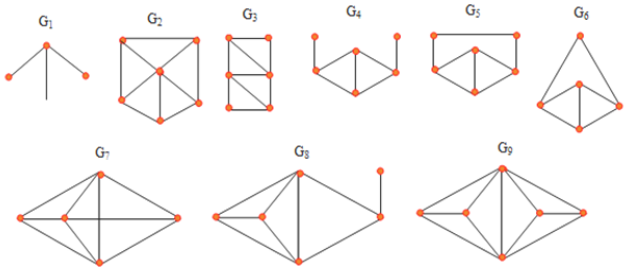


Figure 2. Nine minimum graphs prohibited

In addition to topology control algorithms, the results of this paper can be used as a guideline for network engineers when they want to manually layout a stationary wireless network topology, such as positioning a wireless mesh network (WMN) routers / gateways. If they use this paper's results, then running the optimum link scheduling algorithm in large networks becomes feasible. Then the network's overall performance is clearly promoted.

We show an example in Figure 1(b) in which 2-hop interference model (similar to IEEE 802.11) was implemented to explain the proposed concept. Under this model, all connections are interfering except links to numbers 1 and 7. As is evident, only links 1 and 7 can be selected simultaneously in the root graph (matching). The root graph can be viewed as a link-reorganized version of the network graph on which the matching algorithm can generate the desired set of non-interfering links. The necessary and sufficient conditions for the root graph's existence and the method for generating it were formulated in this paper.

This paper's structure is as follows. In Section 2, the related works are reviewed. We propose our algorithm in Section 3, which is called eLehot and derives prohibited graphs. In Section 4, we analyze the algorithm of eLehot. In Section 5, we demonstrate that there is a rare appearance in the network of derived prohibited graphs. Finally, in Section 6, we conclude.

II. LITERATURE REVIEW

Because of their effect on network performance, connection scheduling algorithms are of interest. An empirical model is designed in Almotairi and Shen (2015) to assess the network's performance in terms of bandwidth. A distributed multi-channel scheduling protocol was also introduced in line with IEEE 802.11 strategy. Brzezinski et al., 2008; Gupta and Shroff, 2010; Chen et al., 2009; Tassiulas and Ephremides, 1992; Yi et al., 2008; Zussman et al., 2008; Chaporkar and Proutiere, 2013; Jiang and Walrand, 2010). Assume that each link is associated with a queue and that packets are queued before being transmitted via the channel. A well-known bandwidth optimal and hierarchical link scheduling algorithm is to find (M W IS) where each vertex's weight is specified at each time slot as the queue length of its corresponding link in the network graph. M W IS, one of the well-known problems with NP-Hard (Sharma et al., 2006). Many simpler algorithms (Chaporkar et al., 2008; Joo, 2008) defined the delay performance of (M W IS) and its approximations (Le et al., 2009; Neely, 2008; Gupta and Shroff, 2011; Ghiasian et al., 2012) due to their favorable characteristics in terms of performance and delay. We use different approaches in this paper to cope with (M W IS) complexity. In our approach, in addition to graph theory phenomena, we use the topology control capacity of wireless networks to extract conditions to reduce the algorithm's complexity.

In this article, the proposed algorithm is a hierarchical scheme. We assume there is a central management node in the network that can perform the task of controlling topology. Several research and practical papers have focused on this type of network, for example, Muruganathan et al. (2005) and Song et al. (2007).

In addition, wireless network management systems have two general approaches,

• Centralized scheme

• Decentralized scheme, each with advantages and disadvantages.

The base station is responsible for collecting information from network nodes and performing management duties in centralized control systems such as Sympathy, MOTE VIEW, BOSS and SNMS (Lee et al., 2006). The hierarchical schemes that incur a high control packet overhead to collect information from all the nodes, making the network's scalability a challenging problem. Executing complicated management tasks with a huge amount of resources on a single server, however, can reduce the processing jobs of wireless nodes and then prolong the life of the network. In addition, the central server (sink) with global system knowledge can make more accurate decisions to control the network and drive the system to its best performance. While distributed management systems have lower overhead communication, their algorithms for resource-constrained wireless network nodes such as wireless sensor networks (WSN) are more complex.

III. IMPLEMENTATION OF THE SCHEME PROPOSED

A stationary wireless network's topology can be managed in two ways

- by putting nodes in a specific position (positioning)
- by manipulating the node's transmission power (Ramanathan and Rosales-Hain, 2000).

There are various applications including smart home design, indoor and outdoor environmental monitoring, transport systems, home networking, etc. where wireless network topology can be managed. Most wireless sensors and actuators are attached to each other in smart homes, for example, and control activities are carried out by a central station. The location of the wireless nodes in those buildings is set and predetermined. Then the network topology is fully controlled (Arampatzis et al., 2005). As a practical example, we refer to the U.C Department of Electrical Engineering and Computer Sciences experiences of the Sensors and Buildings Engineering Research Center (SABER). Berkeley, where 50 Smartdust Motes were installed by researchers to monitor light and temperature to optimally manage the building's energy consumption (Arampatzis et al., 2005).

As an example of outdoor monitoring, we refer to the GDI project, where for habitat monitoring purposes a WSN consisting of 32 nodes was deployed on Great Duck Island. Arampatzis et al. (2005) can find more practical examples in this regard.

In most WMN implementations, such as broadband home networking, public networking, business networking, transportation systems, and metropolitan networking, the router / gateway location is set and defined by network engineers (Akyildiz et al., 2005). In addition, topology is under command in safety and monitoring systems where network cameras and fire / gas detectors can be introduced by wireless means. The topology of the network is completely controllable in all the above-mentioned applications and hence the suggested scheme of this paper is applicable.

Algorithm of eLehot

Suppose the tt^c graph is given and we want to find the tt′ root graph so that L(tt′) = ttc (if tt′ exists) can be a multigraph. When they are adjacent and their neighbourhoods are the same, two vertices u and v are called true twins. If two vertices that are not adjacent

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5, Issue No: 1, September 2019

have identical neighborhoods, they are called false twins. We mean true twin vertices in the rest of the paper wherever we use the term twin vertices. The following observation shows that the vertices of a clique are k mutually real twin vertices in tt^c. A clique is a graph's complete subgraph.

Observation 1: If vertices u_1 and u_2 are twins, and if twins are both u_2 and u_3, then twins are u_1 and u_3. If you do, u_1, u_2, there are twin vertices of each other, then the vertices of the K_t clique, $t > 0$.

We describe the following eLehot algorithm.

Algorithm 1 (eLehot)

Input: tt^c
Step 1. Mark all edges in tt^c that are twins in their end vertices. Then all the labeled edges are contracted. Label vertices with the number of incidents involving contract edges. Finally, consider the weighted simple graph obtained from the vertex as graph H.
Step 2. Run the algorithm of Lehot on graph H.
Step 3. If Lehot algorithm creates the root graph, name it H′ Then add multiple edges to the corresponding edge in H′ equal to the weights of each vertex in H. The graph that results is tt′.

Output: tt′

Note that the individuality of tt′ doesn't matter. Figure 3 shows that the last six graphs of the nine prohibited subgraphs (Figure 2) are line multigraphs using the eLehot algorithm. The labelled edges are indicated in the figure by the symbol' //.' We illustrated all the steps in the algorithm in detail in the first graph, but only the final results were shown for the others.

IV. REVIEW OF ELEHOT ALGORITHMS

We provide the necessary and sufficient conditions for the eLehot algorithm to have an output in this section via three key theorems, prove the correctness of the algorithm and examine its complexity.

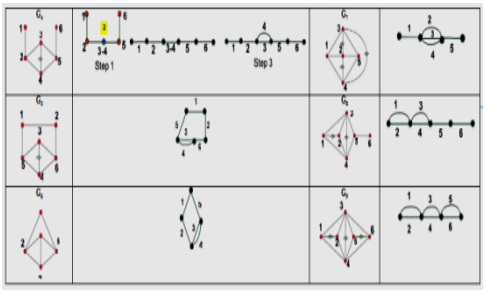


Figure 3. On the last six graphs in Figure 2 running eLehot algorithm (see online version for colors)

Lemma 1: No true twin vertices will remain in the resulting graph after running Step 1 of the eLehot algorithm.

Proof: To see this fact, it is sufficient to show that there is a vertex u ' adjacent to u, which is not adjacent to v, for every two adjacent vertices u and v in the resulting graph H.
Since contraction operation does not create new edges, edge uv is present in tt^c and the vertices u and v are not twin in tt^c. Thus, a vertex u′ exists in tt^c adjacent to u and not adjacent to v. Therefore, vertex u′ is the ideal vertex in H.
Preposition 1 indicates that it is sufficient to run one contraction round (Step 1). This property is necessary in the eLehot algorithm's complexity analysis.
Lemma 2: There is a twin less induced subgraph in ttc isomorphic to F for each induced subgraph F in H and vice versa.
Proof: Assume that F is an induced subgraph of H in order to see this. Next, we demonstrate that the main graph tt^c contains a subgraph F. - vertex of F with multiplicity t, according to Observation 1, is symbolic of a clique, Kt in tt^c.
Now, to create a subgraph F in tt^c, it is necessary to pick one vertex from the cliques corresponding to the vertices of F and make the adjacence between these vertices the same as the adjacence of the vertices in F (Figure 4 clarifies this approach). The subgraph obtained in ttc is isomorphic to F, because in the contraction the adjacency and non-adjacency relationship of the corresponding vertices is maintained.
Likewise, it is possible to use the vice versa of this cycle to get the desired twin less induced ttc subgraph in H.
Theorem 1: If and only if tt^c contains no induced subgraph F1, F2, ..., F7 shown in Figure 5, the eLehot algorithm has an output.
Proof: First, it is easy to see that if and only if the Lehot algorithm on H has an output, the eLehot algorithm on tt^c will have an output. By Beineke's theorem (Beineke, 1968), on the other hand, it is understood that the Lehot algorithm has an output if and only if the input graph does not contain a subgraph{ tt_1, tt_2,..., tt_9} shown in Figure 2.

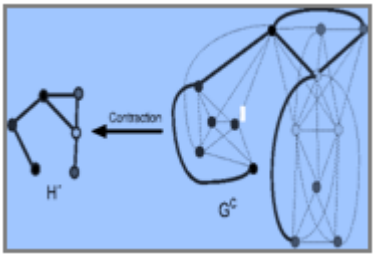


Figure 4. H induced subgraphs can be found in tt^C (colored vertices are contracted) as induced subgraphs (see online color version)

Therefore, to prove the theorem, it is necessary to see that graph H contains an induced subgraph tt_1, tt_2, ..., tt_9 if and only if there is an induced subgraph F_1, F_2, ..., F_7.
Note that by Lemma 1, after running Step 1 of the eLehot algorithm, the resulting graph H removes all twin vertices

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5, Issue No: 1, September 2019

in tt^c and does not create new twin vertices. H is therefore a less graphic twin. Furthermore, if F is a subgraph of H induced by Lemma 2, then tt^c contains an isomorphic subgraph induced by F and vice versa.

They divide the graphs of Figure 2 into two groups, E = {tt_1, tt_2, tt_3,…} say twin less graphs, and ′E = {tt_4, ..., tt_9} as shown in Figure 3.

Next assume that H comprises one of the {tt_1,..., tt_9}. The key point that H is a twin less graph leads us to look at graphs in E′ one by one and see how adding new neighboring vertex (or vertices) for one of the twin vertices could render the graph without twin vertices for each of them. The minimal twin minus subgraphs extracted is the new graphs that are prohibited. This process shows that in Figure 5, these new prohibited graphs are four F_4, F_5, F_6, F_7 graphs.

Remember that the above statement is not in class of graphs. Because these graphs have no twin vertices, graph H can be one of them, and then they are only minimally forbidden graphs.

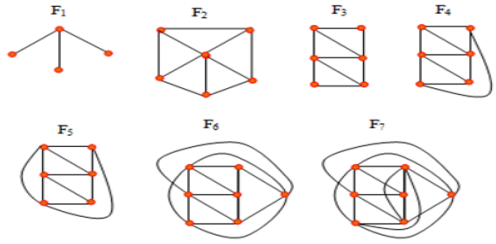


Figure 5 Theorem 1's seven banned graphs

In what follows, we consider each of six category graphs separately to see how by adding the minimum number of vertices we can make them twin less. Each time we come across one of the recognized prohibited (induced) subgraphs, we end up and go to the next case. To see the process of building the prohibited subgraphs, we refer to Figure 6.

i) Consider the tt_4 graph

See Figure 6(A1). The 3 and 4 vertices are real twins. There should be another vertex x adjacent to either 3 or 4 to eliminate twin property. Because of tt_4's symmetry, we assume x is adjacent to 4. The following choices are possible to adjust x to other nodes. Remember that the graph's symmetry allows us to delete similar cases and hold only one for investigation.

When x is only adjacent to the vertex 4, the four vertices {4, 2, 5, x} make F1 (claw) regardless of x to 1 and 6 adjacency. The four vertices {5, 3, x, 6} create a claw if x is adjacent to the vertices 4 and 5. If x is adjacent to vertices 4, 5 and 6 as shown in Figure 6(A1), the resulting graph contains graph F3 as the subgraph caused (by deleting vertex 1). If x is identical to 4, 5, 6 and 2, then {2, 1, 3, x} is a claw. If x is adjacent to vertices 4, 5, 6 and 1, then {x, 1, 4, 6} is a claw regardless of x being adjacent to vertex 2. If x is adjacent to 4, 5, 6, 2 and 1, the claw is {x, 1, 4, and 6}. (Figure 6, section A2).

The above investigations show that tt_4 can be removed from the list of prohibited graphs as F_1 and F_3 prevention yields the same result.

(ii) Take into account tt_5

See Figure 6(B1). The 3 and 4 vertices are identical. There should be another vertex x adjacent to either 3 or 4 to eliminate twin property. Because of tt5's symmetry, we assume x is adjacent to 4. The following choices are possible to adjust x to other vertices. Remember that the graph's symmetry lets us exclude similar cases and hold for investigation only one of them.

When x is only adjacent to vertex 4, then {4, 2, 5, x} is a claw regardless of x to 1 and 6 adjacency. The four vertices {5, 3, x, 6} create a claw if x is adjacent to the vertices 4 and 5. If x is adjacent to vertices 4, 5 and 6, the resulting graph contains graph F3 as the subgraph caused (by deleting vertex 1, in Figure 6(B1)). If x is adjacent to 4, 5, 6 and 2 verices, then 2, 1, 3, x is a claw. If x is adjacent to the 4, 5, 6 and 1 vertices, the resulting graph includes F3 as induced subgraph (by deleting vertex 1). If x is adjacent to vertices 4, 5, 6, 2 and 1, the resulting graph includes graph F2 as the subgraph induced (by deleting vertex 3 in Figure 6(B2)). The above investigations show that tt5 can be removed from the list of prohibited graphs as F1, F2 and F3 prevention yields the same result.

(iii) Take the tt_6 graph

See Figure 6(C1). The 3 and 4 vertices are identical. There should be another vertex x adjacent to either 3 or 4 to eliminate twin property. Because of tt_4's symmetry, we assume x is adjacent to 4. The following options are possible to adjust x to the other vertices.

When x is just adjacent to 4, then {4, 1, 2, x} is a claw. If x is adjacent to the 4 and 2, then {2, 2, 5, 3, x} is a claw. If x is adjacent to vertices 4, 2 and 5, the obtained subsection is shown in Figure 6(C1) and should be added to the list of prohibited line multigraph graphs as it is a new minimal graph that includes one of Beineke's prohibited graphs (tt_6) as an induced subsection and has no twin vertices. In Figure 7, we call this graph F_4.

If vertex x is adjacent to vertices 4, 2, 5 and 1, the obtained subsection is shown in Figure 6(C2) and should be added to the list of prohibited graphs for line multigraphs, since it is a new minimal graph containing one of Beineke's prohibited graphs (tt_6) as induced subsection and has no twin vertices. In Figure 7, we call this graph as F_5.

(iv) Take the tt_7 diagram

There are three mutual twin vertices in this graph. So we need two additional vertices to say x and y to eliminate the graph's twin property. The following addresses all the adjacency possibilities that make the graph twin less. Remember that the twin vertices 3, 4 and 5 symmetry and

the vertices 1 and 2 symmetry in Figure 6(D1) allow us to explain as follows the possible options.

We should take three cases into consideration.

Case 1. Vertex x adjoins vertex 5 and vertex y adjoins vertex 3.

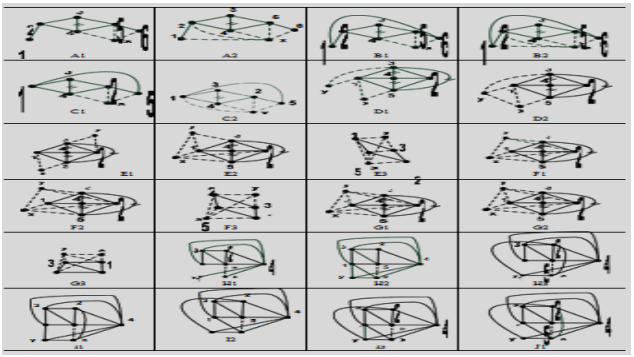
If the vertex x is adjacent to the vertex 5 only, then the claw is {5, x, 1, 2}. The same occurs if vertex y is adjacent only to one of the 3 or 4 twin vertices. Therefore, x and y vertices should be adjacent to more than one tt_7 vertex.

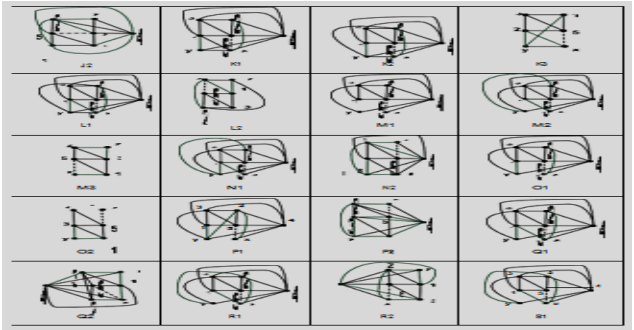
If x is adjacent to 5 and 1 vertex, and y is adjacent to 3 and 1 vertices, then 1,x, y, 4 is a claw (Figure 6(D1)). If x is adjacent to vertices 5 and 1; y is adjacent to vertices 3 and 1; and x and y are adjacent, graph F3 is an induced subgraph shown in Figure 6(D2) (remove vertex 3) of the obtained graph.

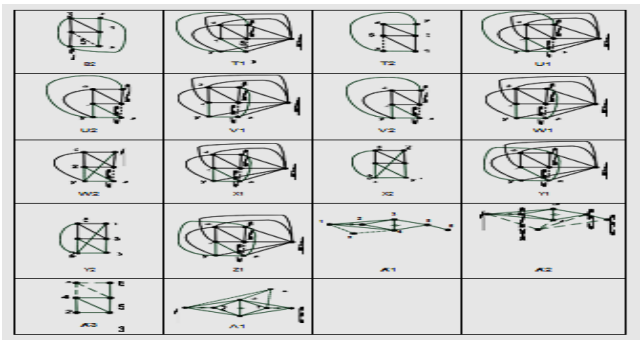
If x is adjacent to vertices 5 and 1; y is adjacent to vertices 3 and 2; then F3 is an induced subsection of the obtained graph shown in Figure 6(E1) (remove vertex 4).

If x is adjacent to vertices 5, 1 and 2; y is adjacent to vertices 3 and 1; x and y are adjacent, then graph F5 as induced as shown in Figure 6(E2) is contained in the constructed graph.

Figure 6 Construct forbidden graphs







Upon deleting vertex 4, the induced graph F5 is achieved and has been redrawn for clarity in Figure 6(E3).

If x is adjacent to vertices 5, 1 and 2; y is adjacent to vertices 3 and 2; x and y are adjacent (Figure 6(F1)), map F5 as induced subgraph as shown in Figure 6(F2) is embedded in the graph. Upon deleting vertex 4, the induced graph F5 is obtained and has been redrawn for clarity in Figure 6(F3).

If x is adjacent to vertices 5, 1 and 2; y is adjacent to vertices 3, 1 and 2; x and y are adjacent (Figure 6(G1)), map F5 as induced graph as shown in Figure 6(G2) is found in the graph. Through deleting vertex 5, the induced graph F5 is achieved and was redrawn for clarification in Figure 6(G3).

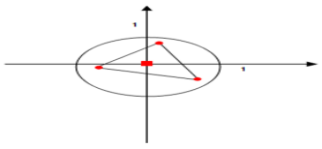


Figure 7 Simulation scenarios showing that F1 happens rarely

Case 2. Vertex x adjoins vertices 5 and 4; vertex y adjoins vertex 4.

If x is adjacent to vertices 5 and 4; y is adjacent to vertex 4 (Figure 6 (H1)),{ 4, x, y, 2,}{ 4, x, y, 3,}{ 4, x, y, 1,}{ 4, 2, 1, x,}{ 4, 2, 1, 1, y} and{ 5, 1, 2, x} are different induced claws. Since in this case none of x and y could be adjacent to vertex 3, x to y adjacency is compulsory to prohibit claw {4, x, y, 3,} otherwise this claw exists in all scenarios. Thus, we assume that x and y are adjacent in other situations under Case 2. Additionally, to preclude claws {4, 2, 1, x} and {4, 2, 1, y}, x and y should be adjacent to vertices 1 and/or 2. Such findings bring us to the situations that follow.

If vertex x is adjacent to vertices 5, 4 and 1; y is adjacent to vertices 4, 1; x and y are adjacent (Figure 6(H2)), then by eliminating vertex 4, the graph constructed includes F_3 as induced subgraph. If x is adjacent to vertices 5, 4 and 2; y is adjacent to vertices 4 and 2; x and y are adjacent (Figure 6(H3)), then the graph constructed includes F_3 as a subgraph caused by removing vertex 4 shown in the same figure.

If x is adjacent to vertices 5, 4 and 2; y is adjacent to vertices 4 and 1; x and y are adjacent (Figure 6(I1)), then the graph constructed contains F_2 as a subgraph induced by removing vertex 5 shown in the same figure. If x is adjacent to vertices 5, 4 and 1; y is adjacent to vertices 4 and 2; x and y are adjacent (Figure 6(I2)), then the graph constructed includes F_2 as a subgraph caused by removing vertex 5 shown in the same figure.

If x is adjacent to vertices 5, 4 and 1; y is adjacent to vertices 4, 2 and 1; x and y are adjacent (Figure 6(I3)), then by eliminating vertex 4, the graph constructed includes F_4 as induced subgraph.

If x is adjacent to vertices 5, 4, 1 and 2; y is adjacent to vertices 4 and 2; x and y are adjacent (Figure 6(J1)), this is a new graph which does not contain any of the forbidden graphs previously found and should then be added to the list of forbidden graphs. As shown in Figure 6(J2), we rearrange its illustration and call it F_6.

If x is adjacent to the vertices 5, 4, 1 and 2; y is adjacent to the vertices 4 and 1; x and y are adjacent to the vertices (Figure 6(K1)), the diagram is the same as the F_6. If x is adjacent to vertices 5, 4 and 2; y is adjacent to vertices 4, 1 and 2; x and y are adjacent (Figure 6(K2)), then by eliminating vertex 4 (Figure 6(K3)) the graph constructed includes F_4 as induced subgraph.

If x is adjacent to vertices 5, 4, 1 and 2; y is adjacent to vertices 4, 1 and 2; x and y are adjacent to each other (Figure 6(L1)), then the graph constructed includes F_5 as caused by removing vertex 4 as shown in Figure 6(L2).

Case 3. Vertex is adjacent to 5 and 4 vertices; vertex y is adjacent to 3 and 4 vertices.

If vertex x is adjacent to vertices 5 and 4; y is adjacent to vertices 3 and 4 (Figure 6(M1)), then { 4, x, y, 2 }, { 4, x, y, 1 }, { 4, 2, 1, x }, { 4, 2, 1, y }, { 5, 1, 2, x } and { 3, y, 1, 2 } are caused by separate claws. We investigate the scenarios where there is no such thing as the described claws. We first analyze the alternatives that are not adjacent to x and y and then look at the neighboring cases of x and y.

If vertex x is adjacent to vertices 5, 4, 1; y is adjacent to vertices 3, 4, 2 (Figure 6(M2)), the graph built contains F_3 as induced by removing vertex 4 as shown in Figure 6(M3).

If vertex x is adjacent to vertices 5, 4, 1; y is adjacent to vertices 3, 4, 2, 1 (Figure 6(N1)), then the graph being constructed is F6 as shown in Figure 6(N2) by rearranging the vertical position.

If vertex x is adjacent to vertices 5, 4, 2; y is adjacent to vertices 3, 4, 1 (Figure 6(O1)), the graph built contains F_3 as induced by removing vertex 4 as shown in Figure 6(O2).

If vertex x is adjacent to vertices 5, 4, 2; y is adjacent to vertices 3, 4, 1, 2 (Figure 6(P1)), otherwise, as shown in Figure 6(P2), the graph constructed is isomorphic to F_6.

If vertex x is adjacent to vertices 5, 4, 1, 2; y is adjacent to vertices 3, 4, 1 (Figure 6(Q1)), then, as shown in Figure 6(Q2), the graph constructed is isomorphic to F_6.

If vertex x is adjacent to vertices 5, 4, 1, 2; y is adjacent to vertices 3, 4, 2 (Figure 6(R1)), otherwise, as shown in Figure 6(R2), the graph constructed is isomorphic to F_6.

If vertex x is adjacent to vertices 5, 4, 1, 2; y is adjacent to vertices 3, 4, 1, 2 (Figure 6(S1)), F_5 as induced subgraph as shown in Figure 6(S2) is embedded in the graph.

If vertex x is adjacent to vertices 5, 4, 1; y is adjacent to vertices 3, 4, 2 and x is adjacent to y (Figure 6(T1)), then the graph constructed includes F_4 as the subgraph caused by removing vertex 4 as shown in Figure 6(T2).

If vertex x is adjacent to vertices 5, 4, 1; y is adjacent to vertices 3, 4, 2, 1 and x adjacent to y (Figure 6(U1)), then the graph constructed includes F_5 as the induced subsection shown in Figure 6(U2) by removing vertex 4.

If vertex x is adjacent to vertices 5, 4, 2; y is adjacent to vertices 3, 4, 1 and x is adjacent to y (Figure 6(V1)), then the graph created contains F_4 as caused by removing vertex 4 as shown in Figure 6(V2).

If vertex x is adjacent to vertices 5, 4, 2; y is adjacent to vertices 3, 4, 1,2 and x is adjacent to y (Figure 6(W1)), then the graph constructed contains F_5 as a subgraph shown in Figure 6(W2).

If vertex x is adjacent to vertices 5, 4, 1, 2; y is adjacent to vertices 3, 4, 1 and x is adjacent to y(Figure 6(X1)), then the graph constructed includes F_5 as a subgraph shown in Figure 6(X2).

If vertex x is adjacent to vertices 5, 4, 1, 2; y is adjacent to vertices 3, 4, 2 and x is adjacent to y (Figure 6(Y1)), then the graph constructed contains F_5 as a subgraph shown in Figure 6(Y2).

If vertexx is adjacent to vertices 5, 4, 1, 2; y is adjacent to vertices 3, 4, 1, 2 and is adjacent to y (Figure 6(Z1)), this is a new graph which does not contain any of the previously discovered prohibited graphs and should then be added to the list of prohibited graphs. We call it a F_7 graph.

(V) Take the tt^8 graph

This graph contains two twin vertices, but one additional vertex says that is enough to delete the graph's twin property. It's because, unlike the tt7 map, the twin vertices do not share any shared vertex and are couples that are completely separated. In the meantime, because of the graph's symmetry, there is only one possible solution to remove the graph's twin property shown in Figure 6(Θ1).

The graph obtained is not a new graph, as it contains {4, 5, x, 2} claws. Indeed any of the vertices 3 or 4 adjacent to x (vertex 4 in this figure) except for one vertex from the set of vertices 1, 2 not adjacent to vertex x (vertex 2 in the figure) in addition to vertex 5 and x always make a claw. We also consider the case that vertex x is adjacent to vertex 5 to preclude the resulting claw. Then there will be a new paw, {5, 6, 3, x}. The only way this claw can be forbidden is to have vertex x adjacent to vertex 6.

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5, Issue No: 1, September 2019

The graph generated was shown in Figure 6(Θ2). The result is graph F3 shown in Figure 6(Θ3) by rearranging Figure 6(Θ2) diagram.

(vi) Take the tt^9 graph

This graph contains three twin vertices, but one additional vertex says x is enough to delete the graph's twin property. It's because the twin vertices don't share any common vertex and are couples that are completely separated. In the meantime, because of the graph's symmetry, there is only one possible solution to remove the graph's twin property shown in Figure 6(Π_1). The graph obtained is not a new graph as it contains the {3, 6, x, 2} claws. Remember that by connecting vertex x to neither vertex 6, nor vertex 2, this claw could not be prohibited otherwise a twin couple will be formed again.

Yes, any of the vertices 3 or 4 (vertex 3 in this figure) that, in addition to one vertex from other twin vertices not adjacent to x (vertices 6 and 2 in the figure), always make a claw. Consequently, graph tt9 does not result in a new graph that is forbidden.

To complete the proof of Theorem 1, note that the construction of graphs F_1, F_2,..., F_7 shows that each graph F_i, $1 \leq I \leq 7$, is a twin less graph containing one of the subgraphs tt1, tt2,..., tt9 induced. So if tt^c contains one of the induced subgraphs F_i, $1 \leq I \leq 7$, then tt_9 preserves in graph H by Lemma 2 its induced subgraph tt_1, tt_2,.

We prove in the following theorem that the eLehot algorithm generates the root graph of the tt^c conflict map.

Theorem 2: If the graph tt′ is the result of the eLehot algorithm on the tt^c conflict graph, then tt′ is the root graph of the tt^c conflict graph, i.e. $L(tt') = tt^c$.

Proof: Assume that tt′ is the eLehot algorithm's output and H′ is a simple graph obtained by tt′ after removing tt″s multiple edges while retaining one edge. Remember that we render the multiple edges according to the vertices tag in H= L(H′) in step 3 of the eLehot algorithm. As a label of its corresponding vertex in H, we keep the multiplicity of each edge. The line graph of tt′ is therefore a graph obtained from H by replacing each vertex with a clique of its label's length. This graph is the original tt^c graph as desired, according to Step 1.

We have the following corollars from Theorems 1 and 2.

Corrolary 1: If the tt^c conflict graph does not contain an induced subgraph F_1, F_2, ..., F_7, then ttc is the graph tt′ line multigraph, where tt′ is an eLehot algorithm output.

Corrolary 2: If and only if eLehot algorithm has an output for it, a given graph tt is a line multigraph.

Proof: The requirement is the outcome of Theorem 2. Assume that G is a line multigraph to see the sufficiency. It has been shown in Bermond and Meyer (1973) and Hemminger (1972) that there are no subgraphs of F_1, F_2, ..., F_7 as induced subgraphs in a line multigraph. In addition, we know from Theorem 1 that if G does not contain graphs of F_1, F_2, ..., F_7 as a subgraph induced, then eLehot algorithm will have an output.

In the following theorem, we analyze the complexity of the algorithm.

Theorem 3: eLehot algorithm's time complexity is $O(|E|^3)$.

Proof: The time complexity of building tt^c (E, L) from tt (V, E) is O(E 2) according to equation (1). Running step 1 of the algorithm has the complexity of O(L. E) or O(E 3), given that in the worst case the limit of L could be up to $|E|(|E|-1)$. Phase 2 has the complexity of $O(|E|^2)$ + O(|E|) (Lehot, 1974), depending on the time complexity of the Lehot algorithm. Step 3 of the proposed algorithm is O(|E|) complex. Consequently, O $(|E|^3)$ has polynomial time complexity in the overall process of buildingtt′.

V. DISCUSSION

We want to show in this chapter that the proposed algorithm does not place as many burdens on the layer of network management. To this end, we show that in a randomly generated network, the presence of forbidden graphs is indeed unusual. Therefore, there is not much overhead for the network due to the restriction provided by the proposed scheme. We focus on graph F_1, which due to its simple topology is most likely to occur. Suppose all nodes have the same range of interference. The following simulation scenario has been set up. We placed a typical receiver in the center of the plane (xy) surrounding a circle with radius 1 as the area of interference. The prohibited graph F_1 is created when there are at least 3 other nodes inside the circle (as transmitter) to interfere with the receiver, but the distance between transmitters is greater than 1. Then all three transmitters are nodes that conflict with each other (see Figure 7). In addition, these transmitters ' corresponding receivers are presumed to be outside the circle and then the conflict graph is F_1.

To demonstrate that such a scenario rarely occurs in reality, we randomly generated 3 nodes within the unit circle ($x^2 + y^2 < 1$) for 10,000,000 times. We found that the distance between each two randomly generated nodes is greater than 1 in only 7.3 percent of the generated scenarios, say that the prohibited F_1 graph is formed in 7.3 percent of the generated cases. This simulation shows that in practice, the occurrence of the conditions that the management layer should take care of will not happen too much.

VI. CONCLUSIONS AND FUTURE WORK

We have generalized the concept of line graphs to line multigraphs in this paper and applied it for link scheduling purposes to the conflict graph of stationary wireless networks. The application of M W M algorithm on the root graph of the conflict graph is shown to be equal to the connection scheduling in the network graph under the general M-hop interference design. We also suggested an algorithm to detect if the conflict graph is

line multigraph and output the root graph. Although applying the optimal connection scheduling algorithm throughput in general is an NP-Hard problem, our proposed overall method results in a polynomial time algorithm of low complexity, given that the conflict graph is line multigraph. It was shown that by prohibiting the construction of seven forbidden graphs in the conflict graph, network designers can meet the derived conditions through topology control of the network. We believe that the results of this paper can be used as a guideline for network designers to plan a stationary wireless network's topology in such a way that the required conditions hold and the optimal algorithm can then be run in much less time. We aim to create a topology control algorithm based on the results of this paper as a future plan.

It is interesting that the proposed algorithm applies not only to single-channel wireless networks, but also to multi-channel wireless networks. We could literally divide the entire network into smaller sub-networks based on radio frequencies (channels) in the case of multi-radio multi-channel networks. The radios with the same frequency can create a sub-network that interferes. Multi-channel network can simply be perceived as a single-channel sub-network unit. Therefore, it is possible to apply the eLehot algorithm to each sub-network (sub-graph). Due to the decrease in the number of interfering links, the complexity would be reduced.

ANNEX TO THE STUDY

We summarize in this appendix some graph theory notes used in this article. The length of a shortest path between u and v in tt is the distance between two vertices u and v in a tt map, denoted by $d_G(u, v)$. The distance between two edges is defined as a function d: (E, E) $\longrightarrow$N, so d(u_1u_2 and v_1v_2, $d(u_1u_2, v_1v_2) = \min_{\{i, j\}\in\{1,2\}} d_G(u_i, v_j)$. A graph tt's power, denoted by tt^t, t N, ϵ is a graph with the same set of vertices as tt where two vertices u and v are adjacent in tt^t if and only if $d_G(u, v) \leq t$. An edge that connects a vertex to itself is a loop in a graph.

A match in a graph is a group of edges that do not have common end vertices.

The line graph of a graph tt = (V, E), denoted by L(tt), is a graph with vertex set E, where two vertices of L(tt) are adjacent when their corresponding edges in tt are adjacent, i.e. they have a common end vertex. In this case, L(tt's) root graph is called graph tt. If there is a root graph tt′ such that tt = L(tt′) is considered a line graph. For other graphical observations and terminologies not defined in this paper, we refer to West (2000).

REFERENCES

[1] Akyildiz, I.F., Wang, X. and Wang, W. (2005) 'Wireless mesh networks: a survey', Computer Networks, Vol. 47, No. 4, pp.445–487.

[2] Almotairi, K. and Shen, X. (2015) 'A distributed multi-channel mac protocol for ad hoc wireless networks', IEEE Transactions on Mobile Computing, Vol. 14, No. 1, pp.1–13.

[3] Arampatzis, T., Lygeros, J. and Manesis, S. (2005) 'A survey of applications of wireless sensors and wireless sensor networks', Proceedings of the 2005 IEEE International Symposium on Intelligent Control, 2005, Mediterrean Conference on Control and Automation, Limassol, pp.719–724.

[4] Beineke, L. (1968) 'Derived graphs of digraphs', Beitrage zur Graphentheorie, pp.17–33.

[5] Bermond, J-C. and Meyer, J.C. (1973) 'Graphe représentatif des arêtes d'un multigraphe', J. Math. Pures Appl., Vol. 52, No. 9, pp.299–308.

[6] Brzezinski, A., Zussman, G. and Modiano, E. (2008) 'Distributed throughput maximization in wireless mesh networks via pre- partitioning', IEEE/ACM Trans. Netw., Vol. 16, pp.1406–1419.

[7] Chaporkar, P. and Proutiere, A. (2013) 'Optimal distributed scheduling in wireless networks under sinr interference model', 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2013, Allerton Park & Retreat Center, University of Illinois at Urbana-Champaign, pp.1372–1379.

[8] Chaporkar, P., Kar, K., Luo, X. and Sarkar, S. (2008) 'Throughput and fairness guarantees through maximal scheduling in wireless networks', IEEE Transactions on Information Theory, Vol. 54, No. 2, pp.572–594.

[9] Chen, Q., Zhang, Q. and Niu, Z. (2009) 'A graph theory based opportunistic link scheduling for wireless ad hoc networks', IEEE Transactions on Wireless Communications, Vol. 8, No. 10, pp. 5075–5085.

[10] Ghiasian, A., Saidi, H. and Behdadfar, M. (2012) 'Delay analysis of randomized algorithms for link scheduling in wireless networks', Int. J. Ad Hoc and Ubiquitous Computing, Vol. 13, No. 1, pp.59–72.

[11] Gupta, G.R. and Shroff, N.B. (2010) 'Practical scheduling schemes with throughput guarantees for multi-hop wireless networks', Computer Networks, Vol. 54, No. 5, pp.766–780.

[12] Gupta, G.R. and Shroff, N.B. (2011) 'Delay analysis and optimality of scheduling policies for multihop wireless networks', IEEE/ACM Trans. on Networking, Vol. 19, pp.129–141.

[13] Hemminger, R. (1972) Characterizations of the Line Graph of a Multigraph, Department of Mathematics, Vanderbilt University. Jiang, L. and Walrand, J. (2010) 'A distributed csma algorithm for throughput and utility maximization in wireless networks', IEEE/ACM Transactions on Networking, Vol. 18, No. 3, pp.960–972.

[14] Joo, C. (2008) 'A local greedy scheduling scheme with provable performance guarantee', Proceedings of ACM MobiHoc, NY, USA, pp.111–120.

[15] Lawler, E. (2001) Combinatorial Optimization: Networks and Matroids, Dover Publications, Mineola, New York.

[16] Le, L.B., Jagannathan, K. and Modiano, E. (2009) 'Delay analysis of maximum weight scheduling in wireless ad hoc networks', 43rd Annual Conference on Information Sciences and Systems, 2009. CISS 2009, Baltimore, Maryland, pp.389–394.

[17] Lee, W.L., Datta, A. and Cardell-Oliver, R. (2006) 'Network management in wireless sensor networks', Handbook of Mobile Ad Hoc and Pervasive Communications, American Scientific Publishers, pp.1–20.

[18] Lehot, P.G.H. (1974) 'An optimal algorithm to detect a line graph and output its root graph', J. ACM, Vol. 21, No. 4, pp.569–575. Muruganathan, S.D., Ma, D.C., Bhasin, R.I. and Fapojuwo, A. (2005)

[19] 'A centralized energy-efficient routing protocol for wireless sensor networks', Communications Magazine, IEEE, Vol. 43, No. 3, pp.S8–13.

[20] Neely, M. (2008) 'Delay analysis for max weight opportunistic scheduling in wireless systems', 46th Annual Allerton Conference on Communication, Control, and Computing, 2008, Allerton Park and Retreat Center, University of Illinois Monticello, IL, USA, pp.683–691.

[21] Ramanathan, R. and Rosales-Hain, R. (2000) 'Topology control of multihop wireless networks using transmit power adjustment', Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2000, IEEE, Vol. 2, pp.404–413.

[22] Santi, P. (2005) 'Topology control in wireless ad hoc and sensor networks', ACM Comput. Surv., Vol. 37, No. 2, pp.164–194.

[23] Sharma, G., Mazumdar, R.R. and Shroff, N.B. (2006) 'On the complexity of scheduling in wireless networks, Proceedings ACM MobiCom, NY, USA, pp.227–238.

[24] Song, J., Han, S., Mok, A.K., Chen, D. and Nixon, M. (2007) 'Centralized control of wireless sensor networks for real-time applications', 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems, FeT'2007, Toulouse, pp.25–32.

[25] Tassiulas, L. and Ephremides, A. (1992) 'Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks', IEEE Trans. on Automatic Control, Vol. 37, No. 12, pp.1936–1948.

[26] Wang, Y. (2008) 'Topology control for wireless sensor networks', Wireless Sensor Networks and Applications, Springer, USA, pp.113–147.

[27] West, D. (2000) Introduction to Graph Theory, 2nd ed., Prentice-Hall, Pearson, USA.

[28] Yi, Y. and Chiang, M. (2008) 'Wireless scheduling algorithms with o(1) overhead for m-hop interference model', Proceedings of IEEE ICC, Beijing, China, pp.3105–3109.

[29] Yi, Y., Proutière, A. and Chiang, M. (2008) 'Complexity in wireless scheduling: impact and tradeoffs', Proceedings of ACM MobiHoc, Hong Kong, China, pp.33–42.

[30] Zussman, G., Brzezinski, A. and Modiano, E. (2008) 'Multihop local pooling for distributed throughput maximization in wireless networks', IEEE INFOCOM, Phoenix AZ, USA, pp.1139–1147.

