A generalization of line graphs via link scheduling in wireless networks

Ali Ghiasian\textsuperscript{a}, Behnaz Omoomi\textsuperscript{b} and Hossein Saidi\textsuperscript{a}

\textsuperscript{a}Department of Electrical and Computer Engineering
\textsuperscript{b}Department of Mathematical Science
Isfahan University of Technology
84156-83111, Isfahan, Iran

Abstract

In single channel wireless networks, concurrent transmission at different links may interfere with each other. To improve system throughput, a scheduling algorithm is necessary to choose a subset of links at each time slot for data transmission. Throughput optimal link scheduling discipline in such a wireless network is generally an NP-hard problem. In this paper, we develop a polynomial time algorithm for link scheduling problem provided that network conflict graph is line multigraph. (i.e., line graph for which its root graph is multigraph). This result can be a guideline for network designers to plan the topology of a stationary wireless network such that the required conditions hold and then the throughput optimal algorithm can be run in a much less time.

Keywords: Link scheduling, Wireless network, Line graph, Line multigraph, Root graph, Conflict graph.

1 Introduction

The underlying wireless network is shown by an undirected and connected graph \(G(V,E)\) in which \(V\) is the set of vertices and \(E\) is the set of edges.

\textsuperscript{1}This work was supported by Iran Telecom Research Center (ITRC)
Every node of the network is represented by a vertex in graph $G$. Two vertices are adjacent if they are within communication range of each other. We assume that time is slotted. In single channel wireless networks, concurrent transmission at the same time slot and different links (edges) may interfere with each other. Therefore, a scheduling discipline is necessary to choose a subset of links at each time slot such that packets do not corrupt due to the interference.

Depending on the method used to deal with interference in such a radio network, different models have been introduced in the literature. A general approach to deal with interference is to consider a conflict graph. The conflict graph of a given graph $G(V,E)$ is graph $G_c(E,L)$. Each vertex in $G_c$ is corresponding to an edge in $G$, and two vertices in $G_c$ are adjacent whenever their corresponding edges in $G$ are interfering edges. In this approach, when a link is ready for transmission, only a subset of links which are called the interference set needs to be considered as interfering links. In other words, each link is associated with an interference set such that the link can be scheduled only if no other link in its interference set is scheduled. Note that if link $l_1$ interferes with link $l_2$ then $l_2$ interferes with $l_1$ as well. Finding a set of non interfering links in $G$ is the same as finding an independent set in $G_c$. An independent set in a graph is a collection of vertices such that there are no edges between them.

We shortly describe how the so called conflict graph can be constructed based on general $M$-hop interference model. First, we refer to some more terminologies of graph theory which we use throughout the paper. The distance between two vertices $u$ and $v$ in a graph $G$, denoted by $d_G(u,v)$, is the length of a shortest path between $u$ and $v$ in $G$. The distance between two edges is defined as a function $d : (E,E) \rightarrow \mathbb{N}$, such that for every two edges $u_1v_2$ and $v_1v_2$, $d(u_1v_2,v_1v_2) = \min_{\{i,j\} \in \{1,2\}} d_G(u_i,v_j)$. The power of a graph $G$, denoted by $G^t$, $t \in \mathbb{N}$, is a graph with the same set of vertices as $G$ in which two vertices $u$ and $v$ are adjacent in $G^t$ if and only if $d_G(u,v) \leq t$. A loop in graph is an edge that connects a vertex to itself. Multiple edges are two or more edges that are incident to the same two vertices. A multigraph is a graph with multiple edges. A simple graph is a graph without loops and/or multiple edges. An edge contraction is an operation which removes an edge from a graph while simultaneously merging its end vertices. We refer to [16] for other graphical notations and terminologies not described in this paper.

The line graph of a graph $G = (V,E)$, denoted by $L(G)$, is a graph with vertex set $E$, where two vertices of $L(G)$ are adjacent if their corresponding
edges in $G$ are adjacent, i.e. they have a common end vertex. In this case, we call graph $G$ the root graph of $L(G)$. A graph $G$ is called a line graph if there is a root graph $G'$ such that $G = L(G')$.

Following the definition of line graph, we now introduce $M$-hop interference model \cite{13} which is mostly used to construct the conflict graph. Under this general interference model, two edges $l_1$ and $l_2$ are interfering edges if $d(l_1, l_2) \leq M$. Therefore, the conflict graph can be defined as follows,

$$G^c(E, L) = [L(G)]^M, \quad M \geq 1. \quad (1)$$

This general interference model is applicable for extensive number of practical applications such as Bluetooth, FH-CDMA systems, Wireless LAN (IEEE 802.11 standard), etc. \cite{13, 17}. More details about different interference models can be found in \cite{12}. For example, in Bluetooth and FH-CDMA systems, two adjacent edges are interfering edges. Link scheduling in these networks results in finding a matching in graph $G$. A matching in a graph is a set of edges with no common end vertices.

In IEEE 802.11 wireless LAN network under the RTS/CTS scheme, two edges that are either adjacent or are both incident on a common edge are interfering edges. Link scheduling in this network results in finding a strong matching in graph $G$. A matching is called strong matching if no edge connects two edges of the matching \cite{4}.

Note that in IEEE 802.11 wireless LAN networks, the conflict graph can be constructed by using Eq. (1) and setting $M = 2$ while in Bluetooth networks the conflict graph is the same as $L(G)$ which is derived by setting $M = 1$ in Eq. (1).

Link scheduling algorithms are of interest due to their impact on the network throughput. Throughput optimal algorithms have been studied extensively in the literature \cite{3, 4, 5, 9, 10, 14, 18, 19}. Let assume that associated to each link is a queue and packets are queued before they are transmitted over the link. A well known throughput optimal link scheduling algorithm is to find maximum weight independent set (MWIS) at each time slot in the conflict graph, where the weight of each vertex is defined as the queue length of its corresponding link in the network graph.

Finding the MWIS is one of the known NP-Hard problems in graph theory \cite{13}. However, if the conflict graph is line graph, then finding MWIS in $G^c$ equals finding maximum weight matching (MWM) in its root graph.
Since there are polynomial time complexity algorithms for $MWM$ problem \cite{7}, then the overall solution is much simpler under this assumption. The key point here that catches our attention is that the root graph does not required to be simple graph. If the root graph is multigraph, it is enough to keep the heaviest edge among multiple edges and remove the others before running $MWM$ algorithm.

Following to this motivation for studying the line graphs, we explore these kind of graphs more precisely. Line graphs are well characterized class of graphs. In \cite{1} it is proved that a graph $G$ is a line graph of a simple graph $G'$ if and only if $G$ does not contain any of the forbidden nine graphs, depicted in Figure 1, as an induced subgraph. An induced subgraph of a graph is a subset of vertices of the graph with edges whose endpoints are both in this subset. Whitney proved that with two exceptional case (triangle and star with three branches, $G_1$ in Figure 1) the structure of $G'$ can be recovered completely from its line graph \cite{16}.

It is worth mentioning that Lehot has developed an optimal algorithm which can be run in linear time to detect whether a graph is line graph and beget its root graph \cite{8}. Lehot algorithm considers only simple graphs as root graphs.

In this paper, following our motivation that allows the root graph to be multigraph, we introduce a generalization of line graph to line multigraph, i.e., line graph for which its root graph is multigraph. Then, we extend Lehot algorithm to line multigraphs and propose a low complexity algorithm, termed as extended Lehot ($e$Lehot), for detecting whether a graph is line
multigraph and output its root graph. Accordingly, by allowing the root graph to be multigraph, we relax the constraint shown in Figure 1, to seven minimal forbidden graphs. Not only the number of forbidden graphs are reduced, prevention of them in the topology construction of the graph is much simpler because they are larger in the number of vertices and edges.

The results of this paper introduce a new approach in topology control algorithms in wireless networks where the final target is complexity reduction. It complements the original motivation of topology control disciplines which tries to minimize energy consumption while the connectivity of network graph is guaranteed [11, 15]. Then, a new design dimension can be added to topology control algorithms by the results of this paper. As a result, based on available polynomial time complexity algorithms for MWM problem [7] and due to the linear time complexity of Lehot algorithm [8], we develop a polynomial time complexity approach for link scheduling algorithm under general M-hop interference model for the class of graphs that their conflict graphs are line multigraphs. In addition to topology control algorithms, the results of this paper can be used as a guideline for network designers when they want to design the topology of a stationary wireless network, e.g., positioning the routers/gateways of a wireless mesh network (WMN). If they prohibit the construction of derived forbidden graphs in the network’s conflict graph, the throughput optimal link scheduling algorithm can be run in the network in much less time. Then the overall performance of the network is obviously promoted.

Let us consider Figure 2 which depicts the idea. The number of edges in $G'$ is equal to the number of edges in $G$, both equal to the number of vertices in $G^c$. Note that, since there is a one to one mapping between each vertex of $G^c$ and each edge in $G'$, then there is also a one to one relation between edges in $G$ and edges in $G'$. Also, note that finding a scheduling in $G$ is the
same as finding an independent set in $G^c$, while finding an independent set in $G^c$ is equivalent to finding a matching in $G'$. Thus, we can deploy a policy of edge selection on $G'$ to obtain an interfering free link selection in network graph $G$.

The structure of this paper is as follows. In Section 2, we propose eLehot algorithm as an extension to Lehot algorithm. We analyze eLehot’s algorithm in Section 3. Finally, we conclude in Section 4.

2 eLehot Algorithm

Suppose that the graph $G^c$ is given and we want to find the root graph $G'$ such that $L(G') = G^c$ (if $G'$ exists), where $G'$ may be a multigraph. Two vertices $u$ and $v$ are called true twins if they are adjacent and their neighborhoods are the same. If two non-adjacent vertices have identical neighborhoods they are called false twins. In the rest of the paper, where ever we use the term twin vertices, we mean true twin vertices. The following observation indicates that $k$ mutually true twin vertices in $G^c$ are the vertices of a clique. A clique is a complete subgraph of a graph.

**Observation 1.** If vertices $u_1$ and $u_2$ are twins, and if $u_2$ and $u_3$ are twins as well, then $u_1$ and $u_3$ are twins. If $u_1, u_2, \ldots, u_t$ are mutually twin vertices, then they are the vertices of a clique $K_t$, $t > 0$.

We describe eLehot algorithm in Figure 3.

Note that we do not care about the uniqueness of $G'$. In Figure 4, it is shown that using eLehot algorithm, the last six graphs of the nine forbidden subgraphs (Figure 1) are line multigraphs. The marked edges are denoted by the symbol “//” in the figure. In the first graph, we have plotted all the steps in the algorithm in details, but for the others, the final result has been shown.

3 eLehot Algorithm Analysis

In this section, through three main theorems, we provide the necessary and sufficient condition for the eLehot algorithm to have an output, prove the correctness of the algorithm and analyze it’s complexity. First, we need to prove the following lemmas.
Algorithm eLehot

Input : $G^c$

Step 1. Mark all edges in $G^c$ which their end vertices are twins.
   Then contract all marked edges.
   Label vertices with the number of contracted edges incident on it.
   Finally, consider the obtained simple vertex weighted graph as graph $H$.

Step 2. Run Lehot algorithm on the graph $H$.
   (refer to [8] for the description of Lehot algorithm).

Step 3. If Lehot algorithm outputs the root graph, say $H'$,
   then equal to the weights of each vertex in $H$,
   add multiple edges to the corresponding edge in $H'$.
   The resulting graph is $G'$.

Output : $G'$

Figure 3: eLehot Algorithm.

Lemma 1.. After running Step 1 of eLehot algorithm, no true twin vertices will remain or produce in the resulting graph.

Proof. To see this fact, it is enough to show that for every two adjacent vertices $u$ and $v$ in the resulting graph $H$, there exists a vertex $u'$ adjacent to $u$ which is not adjacent to $v$.

Since contraction operation does not create any new edges, edge $uv$ exists in $G^c$ and the vertices $u$ and $v$ are not twin in $G^c$. Hence, there exists a vertex $u'$ in $G^c$ adjacent to $u$ and not adjacent to $v$. Thus, vertex $u'$ is the desired vertex in $H$. \qed

Preposition 1 shows that running one round of contraction (Step 1) is sufficient. This property is required in the complexity analysis of eLehot algorithm.

Lemma 2.. For every induced subgraph $F$ in $H$, there exists a twin less induced subgraph in $G^c$ isomorphic to $F$ and vice versa.

Proof. To see this, suppose that $F$ is an induced subgraph of $H$. First, we show that there exists a subgraph $F'$ in the main graph $G^c$. According to
Figure 4: Running eLehot algorithm on the last six graphs in Figure 1.

Observation 1, each vertex of $F$ with multiplicity $t$, is representative of a clique, $K_t$ in $G^c$. Now to construct a subgraph $F$ in $G^c$, it is sufficient to select one vertex from the cliques corresponding to the vertices of $F$ and make the adjacency between these vertices the same as the adjacency of the vertices in $F$ (Figure 5 clarifies this approach). The obtained subgraph in $G^c$ is induced subgraph isomorphic to $F$, since the adjacency and non adjacency relation of the corresponding vertices are preserved in the contraction.

Similarly, the vice versa of this process can be used to obtain the desired twin less induced subgraph of $G^c$ in $H$. □
Figure 5: Induced subgraphs of $H$ can be found as induced subgraphs in $G^c$ (Colored vertices are contracted ones).

Figure 6: Seven forbidden graphs of Theorem 1.

**Theorem 1.** The eLehot algorithm has an output if and only if $G^c$ contains no induced subgraph $\{F_1, F_2, ..., F_7\}$ shown in Figure 6.

**Proof.** First, it is easy to see that the eLehot algorithm on $G^c$ has an output if and only if the Lehot algorithm on $H$ has an output. On the other hand, by Beineke’s theorem \[1\] it is known that Lehot algorithm has an output if and only if it’s input graph contains no induced subgraph $\{G_1, G_2, ..., G_9\}$ shown in Figure 1.

Therefore, to prove the statement it is enough to see that graph $H$ contains an induced subgraph $\{G_1, G_2, ..., G_9\}$ if and only if the conflict graph $G^c$ contains an induced subgraph $\{F_1, F_2, ..., F_7\}$.
Note that, by Lemma 1, the resulting graph $H$ after running Step 1 of eLehot algorithm, removes all twin vertices in $G^c$ and don’t produce new twin vertices. Hence, $H$ is a twin less graph. Also, by Lemma 2, if $F$ is an induced subgraph of $H$, then $G^c$ contains an induced subgraph isomorphic to $F$ and vice versa.

We divide the graphs of Figure 1 to two classes, $\mathcal{E} = \{G_1, G_2, G_3\}$, say twin less graphs, and $\mathcal{E}' = \{G_4, ..., G_9\}$ that all have twin vertices as shown in Figure 4.

First assume that $H$ contains one of the induced subgraphs $\{G_1, ..., G_9\}$. The key point that $H$ is a twin less graph leads us to examine graphs in $\mathcal{E}'$ one by one and for each of them show that how insertion of new neighbor vertex (or vertices) for one of the twin vertices can make the graph without twin vertices. The extracted minimal twin less subgraphs are the new forbidden graphs. This process shows that these new forbidden graphs are four graphs $F_4, F_5, F_6, F_7$ in Figure 6.

Note that the above argument does not hold for graphs in class $\mathcal{E}$. Since these graphs do not have any twin vertices, graph $H$ can be any of them and then they are still minimal forbidden graphs. Therefor the minimal forbidden graphs for $H$ are three graphs of twin less class, $\mathcal{E}$, which are shown in Figure 6 by $F_1, F_2$ and $F_3$ in addition to four graphs that are derived from graphs in class $\mathcal{E}'$ and is depicted in Figure 6 by $F_4, F_5, F_6, F_7$.

In what follows, we consider each of six graphs of class $\mathcal{E}'$ separately to see how we can make them twin less by adding the minimum number of vertices to twin vertices. Whenever we encounter to one of the known forbidden (induced) subgraph, we terminate and go to the next case. We refer to Figure 7 to see the process of constructing the forbidden subgraphs.

i) **Consider the graph $G_4$**

Look at Figure 7-A1. The vertices 3 and 4 are true twins. To remove twin property, there should be another vertex $x$ that is adjacent to either 3 or 4. Due to the symmetry of $G_4$, we suppose that $x$ is adjacent to 4. The following options for adjacency of $x$ to other nodes are possible. Note that the symmetry of the graph helps us to eliminate similar cases and keep only one of them for investigation.

If $x$ is only adjacent to the vertex 4, then the four vertices $\{4, 2, 5, x\}$ make graph $F_1$ (claw) regardless of adjacency of $x$ to 1 and 6. If $x$ is adjacent to the vertices 4 and 5, then the four vertices $\{5, 3, x, 6\}$ make a claw. If $x$ is adjacent to the vertices 4, 5 and 6 as shown in Figure 7-A1, then the resulting graph includes graph $F_3$ as induced subgraph (by deleting vertex
1). If \( x \) is adjacent to the vertices 4, 5, 6 and 2, then \( \{2,1,3,x\} \) is a claw. If \( x \) is adjacent to the vertices 4, 5, 6 and 1, then \( \{x,1,4,6\} \) is a claw regardless of adjacency of \( x \) to vertex 2. If \( x \) is adjacent to the vertices 4, 5, 6, 2 and 1, then \( \{x,1,4,6\} \) is a claw. (Figure 7-A2)

The above investigations show that \( G_4 \) can be removed from the list of forbidden graphs since prevention of \( F_1 \) and \( F_3 \) provides the same result.

ii) Consider the graph \( G_5 \)

Look at Figure 7-B1. The vertices 3 and 4 are twin. To remove twin property, there should be another vertex \( x \) that is adjacent to either 3 or 4. Due to the symmetry of \( G_5 \), we suppose that \( x \) is adjacent to 4. The following options for adjacency of \( x \) to other vertices are possible. Note that the symmetry of the graph helps us to eliminate similar cases and keep only one of them for investigation.

If \( x \) is only adjacent to the vertex 4, then \( \{4,2,5,x\} \) is a claw regardless of adjacency of \( x \) to 1 and 6. If \( x \) is adjacent to the vertices 4 and 5, then the four vertices \( \{5,3,x,6\} \) make a claw. If \( x \) is adjacent to the vertices 4, 5 and 6, then the resulting graph includes graph \( F_3 \) as induced subgraph (by deleting vertex 1, in Figure 7-B1) If \( x \) is adjacent to the vertices 4, 5, 6 and 2, then \( \{2,1,3,x\} \) is a claw. If \( x \) is adjacent to the vertices 4, 5, 6 and 1, the resulting graph includes graph \( F_3 \) as induced subgraph (by deleting vertex 1). If \( x \) is adjacent to the vertices 4, 5, 6, 2 and 1, the resulting graph includes graph \( F_2 \) as induced subgraph (by deleting vertex 3, in Figure 7-B2). The above investigations show that \( G_5 \) can be removed from the list of forbidden graphs since prevention of \( F_1, F_2 \) and \( F_3 \) provides the same result.

iii) Consider the graph \( G_6 \)

Look at Figure 7-C1. The vertices 3 and 4 are twin. To remove twin property, there should be another vertex \( x \) that is adjacent to either 3 or 4. Due to the symmetry of \( G_4 \), we suppose that \( x \) is adjacent to 4. The following options for adjacency of \( x \) to other vertices are possible.

If \( x \) is only adjacent to the vertex 4, then \( \{4,1,2,x\} \) is a claw. If \( x \) is adjacent to the vertices 4 and 2, then \( \{2,5,3,x\} \) is a claw. If \( x \) is adjacent to the vertices 4, 2 and 5, the achieved subgraph is shown in Figure 7-C1 and should be added to the list of forbidden graphs for line multigraphs since it is a new minimal graph that contains one of Beineke’s forbidden graphs (\( G_6 \)) as an induced subgraph and does not have any twin vertices. We call this graph as \( F_4 \) in Figure 6.

If vertex \( x \) is adjacent to the vertices 4, 2, 5 and 1, the achieved subgraph is shown in Figure 7-C2 and should be added to the list of forbidden graphs.
for line multigraphs since it is a new minimal graph that contains one of Beineke's forbidden graphs \((G_6)\) as induced subgraph and does not have any twin vertices. We call this graph as \(F_5\) in Figure 6.

iv) Consider the graph \(G_7\)

This graph contains three mutual twin vertices. Therefore, we need two extra vertices say \(x\) and \(y\) to remove twin property of the graph. All the adjacency possibilities that make the graph twin less are discussed in the following. Note that the symmetry of the twin vertices 3, 4 and 5 and the symmetry of vertices 1 and 2 in Figure 7-D1 helps us to abstract the possible options as follows.

We should consider three cases.

Case 1. Vertex \(x\) is adjacent to vertex 5 and vertex \(y\) is adjacent to vertex 3.

If vertex \(x\) is only adjacent to vertex 5, then \(\{5, x, 1, 2\}\) is a claw. The same occurs if vertex \(y\) is only adjacent to one of twin vertices 3 or 4. Hence, vertices \(x\) and \(y\) should be adjacent to more than one vertex of \(G_7\).

If \(x\) is adjacent to the vertices 5 and 1; and \(y\) is adjacent to the vertices 3 and 1, then \(\{1, x, y, 4\}\) is a claw (Figure 7-D1). If \(x\) is adjacent to the vertices 5 and 1; \(y\) is adjacent to the vertices 3 and 1; and \(x\) and \(y\) are adjacent, then graph \(F_3\) is an induced subgraph of the achieved graph which is shown in Figure 7-D2 (remove vertex 3).

If \(x\) is adjacent to the vertices 5 and 1; \(y\) is adjacent to the vertices 3 and 2; then graph \(F_3\) is an induced subgraph of the achieved graph which is shown in Figure 7-E1 (remove vertex 4). If \(x\) is adjacent to the vertices 5, 1 and 2; \(y\) is adjacent to the vertices 3 and 1; \(x\) and \(y\) are adjacent, then the constructed graph contains graph \(F_5\) as induced subgraph as shown in Figure 7-E2. The induced graph \(F_5\) is achieved by removing vertex 4 and is redrawn in Figure 7-E3 for clarity.

If \(x\) is adjacent to the vertices 5, 1 and 2; \(y\) is adjacent to the vertices 3 and 2; \(x\) and \(y\) are adjacent (Figure 7-F1), then the constructed graph contains graph \(F_5\) as induced subgraph as shown in Figure 7-F2. The induced graph \(F_5\) is achieved by removing vertex 4 and is redrawn in Figure 7-F3 for clarity.

If \(x\) is adjacent to the vertices 5, 1 and 2; \(y\) is adjacent to the vertices 3, 1 and 2; \(x\) and \(y\) are adjacent (Figure 7-G1), then the constructed graph contains graph \(F_5\) as induced graph as shown in Figure 7-G2. The induced graph \(F_5\) is achieved by removing vertex 5 and is redrawn in Figure 7-G3 for clarity.

Case 2. Vertex \(x\) is adjacent to the vertices 5 and 4; vertex \(y\) is adjacent to
the vertex 4.

If $x$ is adjacent to the vertices 5 and 4; $y$ is adjacent to the vertex 4 (Figure 7-H1), then $\{4, x, y, 2\}$, $\{4, x, y, 3\}$, $\{4, x, y, 1\}$, $\{4, 2, 1, x\}$, $\{4, 2, 1, y\}$ and $\{5, 1, 2, x\}$ are different induced claws. Since none of $x$ and $y$ could be adjacent to the vertex 3 in this case, adjacency of $x$ to $y$ is mandatory to prohibit claw $\{4, x, y, 3\}$, otherwise this claw exists in all the scenarios. Thus, in other situations under case 2, we suppose $x$ and $y$ are adjacent. Also, $x$ and $y$ should be adjacent to the vertices 1 and/or 2 to prohibit claws $\{4, 2, 1, x\}$ and $\{4, 2, 1, y\}$. These observations leads us to the following scenarios.

If vertex $x$ is adjacent to the vertices 5, 4 and 1; $y$ is adjacent to the vertices 4, 1; $x$ and $y$ are adjacent (Figure 7-H2), then the constructed graph contains $F_3$ as induced subgraph by removing vertex 4. If $x$ is adjacent to the vertices 5, 4 and 2; $y$ is adjacent to the vertices 4 and 2; $x$ and $y$ are adjacent (Figure 7-H3), then the constructed graph contains $F_3$ as induced subgraph by removing vertex 4 which is shown in the same figure.

If $x$ is adjacent to the vertices 5, 4 and 2; $y$ is adjacent to the vertices 4 and 1; $x$ and $y$ are adjacent (Figure 7-I1), then the constructed graph contains $F_2$ as induced subgraph by removing vertex 5 which is shown in the same figure. If $x$ is adjacent to the vertices 5, 4 and 1; $y$ is adjacent to the vertices 4 and 2; $x$ and $y$ are adjacent (Figure 7-I2), then the constructed graph contains $F_2$ as induced subgraph by removing vertex 5 which is shown in the same figure. If $x$ is adjacent to the vertices 5, 4 and 1; $y$ is adjacent to the vertices 4, 2 and 1; $x$ and $y$ are adjacent (Figure 7-I3), then the constructed graph contains $F_4$ as induced subgraph by removing vertex 4.

If $x$ is adjacent to the vertices 5, 4, 1 and 2; $y$ is adjacent to the vertices 4 and 2; $x$ and $y$ are adjacent (Figure 7-J1), then this is a new graph that does not contain any of previously found forbidden graphs and then should be added to the list of forbidden graphs. We rearrange its illustration as shown in Figure 7-J2 and call it as graph $F_6$.

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 (Figure 7-K1), then the constructed graph is the same as graph $F_6$. If $x$ is adjacent to the vertices 5, 4 and 2; $y$ is adjacent to the vertices 4, 1 and 2; $x$ and $y$ are adjacent (Figure 7-K2), then the constructed graph contains $F_4$ as induced subgraph by removing vertex 4 (Figure 7-K3).

If $x$ is adjacent to the vertices 5, 4, 1 and 2; $y$ is adjacent to the vertices 4, 1 and 2; $x$ and $y$ are adjacent (Figure 7-L1), then the constructed graph contains $F_5$ as induced subgraph by removing vertex 4 which is shown in
Figure 7-L2.

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

If vertex $x$ is adjacent to the vertices 5 and 4; $y$ is adjacent to the vertices 3 and 4 (Figure 7-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 different induced claws. We investigate the scenarios in which the mentioned claws does not exists. We first study the options that $x$ and $y$ are not adjacent and then consider the cases that $x$ and $y$ are adjacent.

If vertex $x$ is adjacent to the vertices 5, 4, 1; $y$ is adjacent to the vertices 3, 4, 2 (Figure 7-M2), then the constructed graph contains $F_3$ as induced subgraph by removing vertex 4 which is shown in Figure 7-M3.

If vertex $x$ is adjacent to the vertices 5, 4, 1; $y$ is adjacent to the vertices 3, 4, 2, 1 (Figure 7-N1), then the constructed graph is $F_6$ as shown in Figure 7-N2 by rearranging the position of vertices.

If vertex $x$ is adjacent to the vertices 5, 4, 2; $y$ is adjacent to the vertices 3, 4, 1 (Figure 7-O1), then the constructed graph contains $F_3$ as induced subgraph by removing vertex 4 which is shown in Figure 7-O2.

If vertex $x$ is adjacent to the vertices 5, 4, 2; $y$ is adjacent to the vertices 3, 4, 1, 2 (Figure 7-P1), then the constructed graph is isomorphic to graph $F_6$ as shown in Figure 7-P2.

If vertex $x$ is adjacent to the vertices 5, 4, 1, 2; $y$ is adjacent to the vertices 3, 4, 1 (Figure 7-Q1), then the constructed graph is isomorphic to graph $F_6$ as shown in Figure 7-Q2.

If vertex $x$ is adjacent to the vertices 5, 4, 1, 2; $y$ is adjacent to the vertices 3, 4, 2 (Figure 7-R1), then the constructed graph is isomorphic to graph $F_6$ as shown in Figure 7-R2.

If vertex $x$ is adjacent to the vertices 5, 4, 1, 2; $y$ is adjacent to the vertices 3, 4, 1, 2 (Figure 7-S1), then the constructed graph contains $F_5$ as induced subgraph as shown in Figure 7-S2.

If vertex $x$ is adjacent to the vertices 5, 4, 1; $y$ is adjacent to the vertices 3, 4, 2 and $x$ is adjacent to $y$ (Figure 7-T1), then the constructed graph contains $F_4$ as induced subgraph by removing vertex 4 which is shown in Figure 7-T2.

If vertex $x$ is adjacent to the vertices 5, 4, 1; $y$ is adjacent to the vertices 3, 4, 2, 1 and $x$ is adjacent to $y$ (Figure 7-U1), then the constructed graph contains $F_5$ as induced subgraph which is shown in Figure 7-U2 by removing vertex 4.
If vertex \( x \) is adjacent to the vertices 5, 4, 2; \( y \) is adjacent to the vertices 3, 4, 1 and \( x \) is adjacent to \( y \) (Figure 7-V1), then the constructed graph contains \( F_4 \) as induced subgraph by removing vertex 4 which is shown in Figure 7-V2.

If vertex \( x \) is adjacent to the vertices 5, 4, 2; \( y \) is adjacent to the vertices 3, 4, 1, 2 and \( x \) is adjacent to \( y \) (Figure 7-W1), then the constructed graph contains \( F_5 \) as induced subgraph which is shown in Figure 7-W2.

If vertex \( x \) is adjacent to the vertices 5, 4, 1, 2; \( y \) is adjacent to the vertices 3, 4, 1 and \( x \) is adjacent to \( y \) (Figure 7-X1), then the constructed graph contains \( F_5 \) as induced subgraph which is shown in Figure 7-X2.

If vertex \( x \) is adjacent to the vertices 5, 4, 1, 2; \( y \) is adjacent to the vertices 3, 4, 1, 2 and \( x \) is adjacent to \( y \) (Figure 7-Y1), then this is a new graph that does not contain any of previously found forbidden graphs and then should be added to the list of forbidden graphs. We call it as graph \( F_7 \).

\textbf{v)} Consider the graph \( G_8 \)

This graph contains two couple of twin vertices, nevertheless one extra vertex say \( x \) suffices to remove twin property of the graph. It is because the twin vertices, unlike the graph \( G_7 \), do not share any common vertex and are completely separated couples. Meanwhile, due to the symmetry of the graph, only one possible solution for removing twin property of the graph exists which is shown in Figure 7-Θ1. The obtained graph is not a new graph since it contains claw \( \{4, 5, x, 2\} \). Indeed any of the vertices 3 or 4 which makes adjacency with \( x \) (vertex 4 in this figure), in addition to one vertex out of the set of vertices \( \{1, 2\} \) which is not adjacent to vertex \( x \) (vertex 2 in the figure) in addition to vertex 5 and \( x \) always make a claw. To prohibit the resulted claw, we consider the case that vertex \( x \) is adjacent to vertex 5 too. Then a new claw, \( \{5, 6, 3, x\} \) is made. Then the only possibility to prohibit this claw is making vertex \( x \) adjacent to vertex 6. The derived graph is shown in Figure 7-Θ2. The result is graph \( F_3 \) which is depicted in Figure 7-Θ3 by rearranging the illustration of Figure 7-Θ2.

\textbf{vi)} Consider the graph \( G_9 \)

This graph contains three couple of twin vertices, nevertheless one extra vertex say \( x \) suffices to remove twin property of the graph. It is because the twin vertices, do not share any common vertex and are completely separated couples. Meanwhile, due to the symmetry of the graph, only one possible
solution for removing twin property of the graph exists which is shown in Figure 7-II1. The obtained graph is not a new graph since it contains the claw \( \{3, 6, x, 2\} \). Note that this claw could not be prohibited by connecting vertex \( x \) to neither vertex 6, nor vertex 2, otherwise a twin couple is constructed again. Indeed any of the vertices 3 or 4 (vertex 3 in this figure) which makes adjacency with vertex \( x \) in addition to one vertex out of each other twin vertices which is not adjacent to \( x \) (vertices 6 and 2 in the figure) always make a claw. Therefore, graph \( G_9 \) does not result in a new forbidden graph.

To complete the proof of Theorem 1, note that regarding to construction of the graphs \( F_1, F_2, ..., F_7 \), it can be seen that every graph \( F_i \), \( 1 \leq i \leq 7 \), is a twin less graph contains one of the induced subgraphs \( G_1, G_2, ..., G_9 \). Thus, if \( G^c \) contains one of the induced subgraph \( F_i \), \( 1 \leq i \leq 7 \), then by Lemma 2 its induced subgraph \( G_1, G_2, ..., G_9 \) preserves in graph \( H \).

In the following theorem, we prove that eLehot algorithm begets the root graph of the conflict graph \( G^c \).

**Theorem 2.** If graph \( G' \) is the output of eLehot algorithm on conflict graph \( G^c \), then \( G' \) is the root graph of conflict graph \( G^c \), i.e. \( L(G') = G^c \).

**Proof.** Assume that \( G' \) is the output of the eLehot algorithm and \( H' \) be a simple graph obtained by \( G' \) after removing the multiple edges of \( G' \) but keeping one edge. Note that in Step 3 of eLehot algorithm, we make the multiple edges according to the label of vertices in \( H = L(H') \). We keep the multiplicity of each edge as a label of its corresponding vertex in \( H \). Therefore, the line graph of \( G' \) is a graph obtained from \( H \) by replacing every vertex by a clique of the size of its label. According to Step 1, this graph is the original graph \( G^c \) as desired.

By Theorems 1 and 2, we have the following corollaries.

**Corollary 1.** If the conflict graph \( G^c \) contains no induced subgraph \( F_1, F_2, ..., F_7 \), then \( G^c \) is the line multigraph of graph \( G' \), where \( G' \) is an output of eLehot algorithm.

**Corollary 2.** A given graph \( G \) is a line multigraph if and only if eLehot algorithm has an output for it.

**Proof.** The necessity is the result of Theorem 2. To see the sufficiency, assume that \( G \) is a line multigraph. In [2, 6] it has been shown that a line
multigraph does not contain any subgraphs of \( \{F_1, F_2, ..., F_7\} \) as induced subgraph. Moreover, by Theorem 1 we know that if \( G \) contains no graphs of \( \{F_1, F_2, ..., F_7\} \) as induced subgraph, then eLehot algorithm has an output.

We analyze the algorithm’s complexity in the following theorem.

**Theorem 3.** The time complexity of eLehot algorithm is \( O(|E|^3) \).

**Proof.** According to Eq. [1], the time complexity of constructing \( G'(E, L) \) from \( G(V, E) \) is \( O(|E|^2) \). Running Step 1 of the algorithm has complexity of \( O(|L| \cdot |E|) \) or \( O(|E|^3) \), considering that in the worse case maximum of \( |L| \) could be up to \( \frac{|E|(|E|-1)}{2} \). According to the time complexity of Lehot algorithm, Step 2 has the complexity \( O(|E|^2) + O(|E|) \) \[8\]. Step 3 of the proposed algorithm has the complexity of \( O(|E|) \). Consequently, the overall process of constructing \( G' \) has polynomial time complexity of \( O(|E|^3) \). \( \square \)

### 4 Conclusions and Future work

In this paper, we have generalized the concept of line graphs to line multigraphs and applied it to the conflict graph of stationary wireless networks for the purpose of link scheduling. It is shown that applying MWM algorithm on the root graph of the conflict graph is equivalent to the link scheduling under general M-hop interference model in the network graph. We have proposed an algorithm to detect whether the conflict graph is line multigraph and output its root graph. The proposed algorithm is an extension of the well known Lehot algorithm to the line multigraphs and is called eLehot.

While applying the throughput optimal link scheduling algorithm in general is an NP-Hard problem, our overall proposed method results in a low complexity polynomial time algorithm, provided that the conflict graph is line multigraph. It was shown that how the derived conditions can be satisfied by network designers through topology control of the network by prohibiting the construction of seven forbidden graphs in the conflict graph. We believe that the results of this paper can be used as a guideline for network designers to plan the topology of a stationary wireless network such that the required conditions hold and then the throughout optimal algorithm can be run in a much less time. As a future plan, we aim to design a topology control algorithm based on the results of this paper.
References

[1] L.W. Beineke. Derived graphs of digraphs. *Beitrage zur Graphentheorie*, pages 17–33, 1968.

[2] J.-C. Bermond and J. C. Meyer. Graphe représentatif des arêtes d’un multigraphe. *J. Math. Pures Appl. (9)*, 52:299–308, 1973.

[3] Andrew Brzezinski, Gil Zussman, and Eytan Modiano. Distributed throughput maximization in wireless mesh networks via pre-partitioning. *IEEE/ACM Trans. Netw.*, 16:1406–1419, December 2008.

[4] Martin Charles Golumbic and Moshe Lewenstein. New results on induced matchings. *Discrete Applied Mathematics*, 101(1-3):157 – 165, 2000.

[5] Gagan Raj Gupta and Ness B. Shroff. Practical scheduling schemes with throughput guarantees for multi-hop wireless networks. *Computer Networks*, 54(5):766 – 780, 2010.

[6] R.L. Hemminger. *Characterizations of the line graph of a multigraph*. Dept. of Mathematics, Vanderbilt University, 1972.

[7] E.L. Lawler. *Combinatorial optimization: networks and matroids*. Dover Publications, 2001.

[8] Philippe G. H. Lehot. An optimal algorithm to detect a line graph and output its root graph. *J. ACM*, 21, October 1974.

[9] Eytan Modiano, Devavrat Shah, and Gil Zussman. Maximizing throughput in wireless networks via gossiping. in *Proceedings of ACM SIGMETRICS*, 34(1):27–38, 2006.

[10] Jian Ni and R. Srikant. Distributed csma/ca algorithms for achieving maximum throughput in wireless networks. In *Information Theory and Applications Workshop*, page 250, feb. 2009.

[11] Paolo Santi. Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.*, 37(2):164–194, June 2005.
[12] Paolo Santi, Ritesh Maheshwari, Giovanni Resta, Samir Das, and Douglas M. Blough. Wireless link scheduling under a graded sinr interference model. In *Proceedings of the 2nd ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing, FOWANC ’09*, pages 3–12. ACM, 2009.

[13] Gaurav Sharma, Ravi R. Mazumdar, and Ness B. Shroff. On the complexity of scheduling in wireless networks. In *Proceedings ACM MobiCom*, pages 227–238, NY, USA, 2006.

[14] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control*, 37(12):1936–1948, 1992.

[15] Yu Wang. Topology control for wireless sensor networks. *Wireless Sensor Networks and Applications*, pages 113–147, 2008.

[16] D. West. *Introduction to Graph Theory, 2nd Edition, Prentice Hall*. 2000.

[17] Yung Yi and Mung Chiang. Wireless scheduling algorithms with o(1) overhead for m-hop interference model. In *Proceedings of IEEE ICC*, pages 3105–3109, 2008.

[18] Yung Yi, Alexandre Proutière, and Mung Chiang. Complexity in wireless scheduling: impact and tradeoffs. In *Proceedings of ACM MobiHoc*, pages 33–42, 2008.

[19] G. Zussman, A. Brzezinski, and E. Modiano. Multihop local pooling for distributed throughput maximization in wireless networks. In *IEEE INFOCOM*, pages 1139–1147, april 2008.
Figure 7: Construction of Forbidden Graphs.