The Minimum Clearing Time Problem in Wireless Networks

By

Swaroop Gopalam

B.Tech, IIT Bombay, 2014.

A thesis submitted to Macquarie University

for the degree of Master of Research

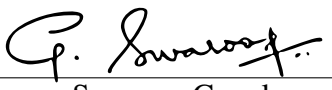
Department of Engineering

10 October 2016

MACQUARIE
University
SYDNEY·AUSTRALIA

Typeset in LaTeX 2_ε.

The work presented in this thesis was carried out at the Department of Engineering, Macquarie University, Sydney, Australia, between January 2016 and October 2016. This work was principally supervised by Prof. Stephen Hanly and co-supervised by Prof. Philip Whiting. Except where acknowledged in a customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any other university or institution other than Macquarie University.



Swaroop Gopalam

Abstract

The thesis focuses on the minimum time clearing objective for link scheduling in wireless networks. We formulate the problem as a deterministic LP. We relate the problem to other works on capacity and scheduling. In general, the problem is NP-complete. We consider special networks where the additional structure makes the problem tractable. We argue that explicitly solving the linear program is implausible. And proceed to show that the problem is amenable to a greedy allocation scheme that only requires local information.

Beyond the special case, we consider simple ring networks where in general, the greedy solution is sub-optimal. We solve the minimum clearing time problem for ring networks under 1-hop interference model. We establish the sufficiency conditions for optimality of a greedy solution. We also provide the solution when these conditions do not hold. For this case, the solution depends on global information, unlike the greedy solution. However, the solution only has a linear complexity in terms of computation.

The minimum clearing time problem can help us understand how the network is constrained under different load distributions. Ultimately, in the future we aim to use these insights to design a simple scheduling policy, with nearly optimal performance in a general wireless network.

Contents

Chapter 1

Introduction

1.1 Introduction

Understanding the capacity of a general wireless network is a problem that has received significant attention from researchers. A general wireless network consists of several transmitter - reciever pairs. Transmission of information from a transmitter to a receiver happens via a wireless channel i.e., transmitter and receiver are connected via a wireless link. Now, given a set of arrival rates of traffic (on each link), is it possible to find a scheduling policy to stabilize the network? This is an important question for which the answer is not straightforward, because of the interference between wireless links. Interference occurs when two or more links use the same frequencies at the same time.

It is clear that interference is an inherent property of wireless networks. However, the power of a wireless signal attenuates rapidly with the distance travelled. Hence, two links that are not in the vicinity (or range) of each other, can transmit simultaneously without significant interference. Therefore in a wireless network, interference imposes constraints as to which links can transmit simultaneously, and the rate at which the transmissions can happen. The capacity region (or stability region) is the set of all the arrival rate vectors for which the network can be stabilized. A system is stable iff traffic does not build up indefinitely at the network nodes. Methods implemented in current wireless networks achieve only a fraction of the capacity region. Example: 802.11 MAC [1]. Finding simple policies that can achieve close to optimal performance is a challenging task, and will enable support of more wireless traffic than what is possible today.

In [2], Tassiulas *et al.* proposed a throughput optimal scheduling policy for general wireless networks. A throughput optimal policy stabilizes the network for any arrival rate vector within the capacity region. The proposed scheduling policy is often referred to as *maximum weighted scheduling* (MWS), as it relies on finding the maximum weighted activation set. In every slot, queue length based maximum weighted activation set is selected for scheduling. Unfortunately, finding the maximum weighted activation set is equivalent to finding the maximum weighted independent set (in the conflict graph), which is known to be NP-hard in general. Hence, maximum weighted scheduling has an exponential complexity, and requires centralized control. As a result, max weight scheduling is seldom implemented in practice.

Following [2], a lot of effort has been put into finding low complexity approximations of max-weight scheduling. The simplest of such algorithms is the *Maximal Scheduling* considered in [3]. The basic idea behind maximal scheduling is to greedily select some maximal activation set, even though it does not have the maximum weight. A maximal set of non-interfering and non-empty queues are selected for scheduling in every slot. A distributed implementation of a similar greedy scheduling scheme was studied in [4]. As one would expect, the simplicity of maximal scheduling comes at a cost in terms of acheivable capacity. Maximal scheduling only achieves a small fraction of the capacity region.

Longest Queue First (LQF) or *Greedy Maximal Scheduling* (GMS) considered in [5–7] is a natural simplification of the max weight scheduling. In LQF, the queues are prioritized for scheduling based on their queue lengths. And a maximal schedule is constructed by selecting non-interfering queues based on the queue length, i.e., longest queue Q_1 is selected first, and then the longest queue from the set of queues that do not interfere with Q_1 is selected, and this process is repeated for maximality. LQF has received a lot of attention in the recent literature, specifically its performance in various scenarios [5–7]. The efficiency of the LQF scheduling is given by the local pooling factor [5, 6]. Notable cases where performance of LQF was analyzed include tree and ring networks. LQF was shown to be throughput optimal in tree networks under the K-hop interference model [5]. LQF is shown to be sub-optimal in ring networks, e.g. local pooling factor in a ring under the 1-hop interference model is $2/3$ [7].

Another approach to the problem of scheduling in wireless networks is studied in [8].

Under the 1-hop interference model, the problem of finding the maximum weighted activation set boils down to the maximum weighted matching problem. Matching is a set of links such that no two links share the same node. The authors in [8] extend this framework to a K-hop interference model, by defining K-valid matching. Unsurprisingly, the hardness of maximum weighted K-valid matching problem was shown to be NP-hard for $K \geq 2$ [8].

One of the earliest works on the capacity of a general wireless network is presented in [9]. The author, Erdal Arikan studied the problem of determining whether a given arrival rate vector lies within the capacity region. The problem was studied as two sub-problems: the FF (TDMA-$\vec{f} - feasibility$) problem and the RF ($\vec{r}$-feasibility) problem. The FF problem deals with the feasibility of a given rate vector under 1-hop traffic, whereas the RF problem also deals with optimal routing decisions involving multi-hop traffic. Both FF and RF problems were shown to be NP-hard. The FF problem is central to the topic of this thesis, we will discuss this further after introducing some context via our problem formulation.

Recent works of Hanly *et al.* [10, 11] on capacity of HetNets, provide an interesting approach to characterizing capacity. They formulated the minimum clearing time problem as a linear program, and used the solution to the linear program to characterize the capacity region. The linear program had special structural properties which were exploited to obtain the solution. Minimum clearing time problem was also recently studied in [12]. They were solving the problem for special wireless networks with structure (referred to as multi cluster cardinality based rates), and they achieved polynomial time solvability for these networks. The aim of this thesis is similar to these works [10–12]: we want to solve the minimum clearing time problem for special networks where the additional structure makes the problem tractable.

1.2 Capacity and Minimum Clearing Time Problem

We model a wireless network as an undirected graph $G(V, E)$, where V represents the set of wireless nodes and E represents the set of links. We model the interference as follows: a link ℓ interferes with all the links in the set $I(\ell) \subseteq E$. We assume that two interfering links cannot transmit at the same time. We consider a slotted time model. We assume that in each slot, a link ℓ can transmit one packet provided that no link in $I(\ell)$ is transmitting in the same

slot. Let $\{e_1, e_2, ...e_{|E|}\}$ denote an enumeration of the links in the set E. Let $A_i(t)$ denote the number of packets arriving at a link e_i at the beginning of slot t. We assume that the arrival process $\{A_i(t)\}_{t=1:\infty}$ is a sequence of i.i.d random variables for $i = 1 : |E|$. We further assume that arrival processes are independent for different links, and they satisfy the conditions for the fluid limits to hold. Let, $\tau(e_i) = \mathbb{E}_t[A_i(t)]$ denote the arrival rate of packets at a link e_i.

Definition 1.2.1. *A feasible set is set $A \subseteq E$ such that $\forall \ell \in A$, $I(\ell) \bigcap A = \phi$.*

From the definition, no two links in a feasible set interfere with each other. Therefore, all the links in a feasible set can be scheduled together.

Definition 1.2.2. *A maximal feasible set is a feasible set that is not a subset of any other feasible set.*

Let $\mathcal{S}$ denote the set of all the maximal feasible sets S for the graph $G(V, E)$. Denote the arrival rate vector as $\vec{\tau} = [\tau(e_1), \tau(e_2), ...\tau(e_{|E|})]$. Consider the vector representation $\vec{r}_S$ of a maximal feasible set S as follows: $\vec{r}_S$ is a vector of length $|E|$, and the i th element of $\vec{r}_S$ is given by

$$\vec{r}_S(i) = \begin{cases} 1 & \text{if } e_i \in S, \\ 0 & \text{o.w.} \end{cases}$$

We refer to $\vec{r}_S$ as a maximal scheduled vector. Let $\mathcal{R}$ denote the set of all the maximal scheduled vectors $\{\vec{r}_S\}_{S \in \mathcal{S}}$. Let $Conv(A)$ denote the convex hull of a set A. Consider the set of arrival rate vectors given by

$$\Gamma = \{\vec{\tau} | \vec{\tau} \leq \vec{\gamma} \text{ for some } \vec{\gamma} \in Conv(\mathcal{R})\}$$

where $\vec{a} \leq \vec{b}$ denotes $\vec{a}$ is element wise less than or equal to $\vec{b}$.

We consider the system to be stable iff a scheduling policy can be found such that the queue lengths do not grow to infinity. It is well known that Γ is the stability region for the considered system [2].

Theorem 1.2.1. *If the arrival rate vector $\vec{\tau} \notin \Gamma$, the system is unstable under any scheduling policy.*

Proof. See Lemma 3.3 in [2]. □

Observe that since $\vec{\gamma} \in Conv(\mathcal{R})$, $\vec{\gamma}$ can be written as $\vec{\gamma} = \sum_{S\in\mathcal{S}} f_S \vec{r}_S$ such that $\sum_{S\in\mathcal{S}} f_S = 1$ and $f_S \geq 0, \forall S \in \mathcal{S}$. Now imagine a policy that picks a maximal feasible set S for f_S fraction of time slots, for scheduling. Under such a policy, if $\vec{\tau} \leq \vec{\gamma}$, each link ℓ receives a service rate higher than its arrival rate $\tau(\ell)$. So for any arrival rate vector $\vec{\tau} \in \Gamma$, this policy can stabilize the network. This is the intuition behind our formulation of the minimum clearing time problem. We formulate the minimum clearing time problem as the following linear program.

$$\begin{aligned} &\min \sum_{S\in\mathcal{S}} f_S \\ &\text{s.t.} \\ &\sum_{S:\ell\in S} f_S \geq \tau(\ell), \forall \ell \in E, \\ &f_S \geq 0, \forall S \in \mathcal{S} \end{aligned} \tag{1.1}$$

where f_S represents time alloted to a maximal feasible set S.

Observe that this linear program is a deterministic problem. For the graph $G(V, E)$, we consider a fixed load (equivalent to the arrival rate on each link) of $\tau(\ell)$ on each link $\ell \in E$. $\tau(\ell)$ is treated as the time required to clear a link ℓ. Now the interference constraint can be interpreted as: no two interfering links can be cleared simultaneously. The objective of the minimum clearing time problem is to clear the network in the least possible time subject to the interference constraints. We will now show the relation between the set Γ and the linear program (1.1).

Claim 1. $\{\vec{\tau} | \sum_{S\in\mathcal{S}} f_S^* \leq 1\} = \Gamma$, *where* $\{f_S^*\}_{S\in\mathcal{S}}$ *is an optimal solution of the LP (1.1).*

Proof. From the definition, for any $\vec{\tau} \in \Gamma$, $\exists \vec{\gamma} \in Conv(\mathcal{R})$ such that$\vec{\gamma} \geq \vec{\tau}$. We can write $\vec{\gamma} = \sum_{S\in\mathcal{S}} f_S \vec{r}_S$ such that $\sum_{S\in\mathcal{S}} f_S = 1$ and $f_S \geq 0, \forall S \in \mathcal{S}$. $\sum_{S\in\mathcal{S}} f_S \vec{r}_S \geq \vec{\tau}$ is equivalent to the constraints $\sum_{S:\ell\in S} f_S \geq \tau(\ell), \forall \ell \in E$. Therefore, $\{f_S\}_{S\in\mathcal{S}}$ is a feasible solution of the LP (1.1). Therefore, $\sum_{S\in\mathcal{S}} f_S^* \leq \sum_{S\in\mathcal{S}} f_S = 1$. Hence, $\Gamma \subseteq \{\vec{\tau} | \sum_{S\in\mathcal{S}} f_S^* \leq 1\}$.

Conversely, given $\vec{\tau}$, suppose that $\sum_{S\in\mathcal{S}} f_S^* \leq 1$. we construct the following solution

$$f_S = \frac{f_S^*}{\sum_{S\in\mathcal{S}} f_S^*}, \forall S \in \mathcal{S}$$

and set $\vec{\gamma} = \sum_{S \in \mathcal{S}} f_S \vec{r}_S$. Observe that $f_S \geq f_S^*, \forall S \in \mathcal{S}$ and $\sum_{S \in \mathcal{S}} f_S = 1$. Therefore, $\vec{\gamma} \geq \vec{\tau}$ and $\vec{\gamma} \in Conv(\mathcal{R})$, which implies $\tau \in \Gamma$. Hence, $\{\vec{\tau} | \sum_{S \in \mathcal{S}} f_S^* \leq 1\} \subseteq \Gamma$. □

From Claim 1, it is clear that solution of the minimum clearing time problem can determine the capacity region of the network. In [9], the minimum clearing time problem was considered in the same context, although Claim 1 is not made. The formulation of Γ is a result of much later works in the literature [2]. Using Claim 1, we establish the equivalence of the two formulations.

In [9], the author considers the problem of determining whether a given arrival rate vector belongs to the capacity region of a packet radio network. The FF (TDMA-$\vec{f}$-feasibility problem) is equivalent to our formulation of the minimum clearing time problem. The complexity of the FF problem was well analyzed in [9]. The author provides algorithms that transforms the FF problem to the CLIQUE problem. The CLIQUE problem is known to be NP-complete in general [9]. Therefore in general, the minimum clearing time problem is NP-complete.

Much of the hardness of the problem can be attributed to constructing the set $\mathcal{S}$. The number of maximal feasible sets grow exponentially with respect to the number of links. Therefore, constructing the set $\mathcal{S}$ is not practical except in case of small networks. The LP itself has exponentially many number of variables due to size of $\mathcal{S}$. Therefore, explicitly solving the LP is not practical. We will discuss more on this topic in Chapter 2.

Chapter 2

Tree Networks

2.1 Introduction

In the previous chapter, we have established that solving the minimum clearing time problem in a general network is NP-hard. So, it is natural to start with graphs with additional structural properties, and see whether the problem is tractable. In this chapter, we focus on solving the minimum clearing time problem in tree networks under the K-hop interference model.

2.2 Network Model and Problem Formulation

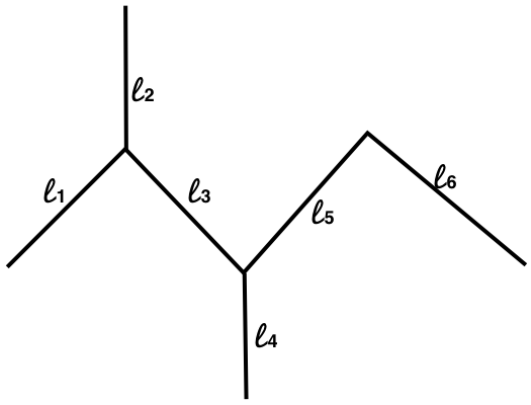


Figure 2.1: An example graph containing 6 links

We model a wireless network as an undirected graph $G(V, E)$, where V represents the set of wireless nodes and E represents the set of links. Furthermore, we assume that $G(V, E)$

is a tree. We consider a K-hop interference model [8], i.e., a link ℓ interferes with all the links within a K-hop distance of ℓ. We assume that two interfering links cannot be cleared at the same time. We define the 1-hop neighbourhood of $\ell \in E$ as the set of all edges in E that share a common node with ℓ. Denote the 1-hop neighbourhood of ℓ as $N_1(\ell)$. For example in Figure 2.1, $N_1(\ell_1) = \{\ell_1, \ell_2, \ell_3\}$ and $N_1(\ell_3) = \{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5\}$. Note that $N_1(\ell)$ also includes ℓ. For a set of edges A, define $N_1(A) = \bigcup_{\ell \in A} N_1(\ell)$. Now we can define l-hop neighbourhood of a link ℓ inductively as follows

Definition 2.2.1. *The l-hop neighbourhood of a link ℓ, denoted by $N_l(\ell)$ is defined as* $N_l(\ell) = N_1(N_{l-1}(\ell))$.

From the definition, $N_K(\ell)$ is the set of all the links within a K-hop distance of ℓ, including ℓ. Hence, ℓ interferes with every link in $N_K(\ell) - \{\ell\}$. Based on this, we can redefine the feasible set as follows

Definition 2.2.2. *A feasible set is a set $A \subseteq E$ such that $\forall \ell \in A$, $N_K(\ell) \bigcap A - \{\ell\} = \phi$.*

From the definition, no two links in a feasible set are within K-hop distance of each other. Therefore, all the links in a feasible set can be scheduled together.

We treat time as an infinitely divisible resource, and clear the network by sharing time among the maximal feasible sets of the given graph $G(V, E)$. Let $\mathcal{S}$ denote the set of all the maximal feasible sets S for the graph $G(V, E)$. Let $\tau(\ell)$ denote the time required to clear a link $\ell \in E$. We formulate the minimum clearing time problem as the following linear program

$$\begin{aligned} &\min \sum_{S \in \mathcal{S}} f_S \\ &\text{s.t.} \\ &\sum_{S:\ell \in S} f_S \geq \tau(\ell), \forall \ell \in E, \\ &f_S \geq 0, \forall S \in \mathcal{S} \end{aligned} \tag{2.1}$$

where f_S represents the time allocated to a maximal feasible set S.

2.3 Lower Bounds

Definition 2.3.1. *A clique set is a set $C \subseteq E$ such that, $\forall \ell \in C$, $N_K(\ell) \supseteq C$.*

Definition 2.3.2. *A maximal clique set is a clique set that is not a subset of any other clique set.*

Observe that a maximal clique set represents a set of mutually interfering links, i.e., every two links in a maximal clique set interfere with each other. Now, we will discuss a few properties of maximal clique sets. To help with the discussion, we introduce some notation. Let $P(\ell_1, \ell_2)$ denote the path joining ℓ_1 and ℓ_2 also including ℓ_1 and ℓ_2. $|P(\ell_1, \ell_2)|$ denotes the length of the path $P(\ell_1, \ell_2)$ in terms of number of links.

Lemma 2.3.1. *Let C be a maximal clique set of the graph $G(V, E)$. If $\ell_1, \ell_2 \in C$, then $P(\ell_1, \ell_2) \subseteq C$.*

Proof. Consider a link $\ell \in C$ and $\ell \neq \ell_1, \ell_2$. Since, ℓ and ℓ_1 interfere, $|P(\ell, \ell_1)| \leq K + 1$. Thus, $N_K(\ell) \supseteq P(\ell, \ell_1)$. Similarly, $N_K(\ell) \supseteq P(\ell, \ell_2)$. And thus $N_K(\ell) \supseteq P(\ell, \ell_1) \bigcup P(\ell, \ell_2)$. Since $G(V, E)$ is a tree, $P(\ell, \ell_1) \bigcup P(\ell, \ell_2) \supseteq P(\ell_1, \ell_2)$ which implies $N_K(\ell) \supseteq P(\ell_1, \ell_2)$. Since this is true $\forall \ell \in C$, every link $\ell \in C$ interferes with every link in the path $P(\ell_1, \ell_2)$. Since $\ell_1, \ell_2 \in C$, $|P(\ell_1, \ell_2)| \leq K + 1$, which implies for every $\ell \in P(\ell_1, \ell_2)$, $N_K(\ell) \supseteq P(\ell_1, \ell_2)$. Therefore, $P(\ell_1, \ell_2)$ is itself a clique set.

Therefore, $P(\ell_1, \ell_2) \bigcup C$ is also a clique set. If $C \not\supseteq P(\ell_1, \ell_2)$, it contradicts the assumption that C is a maximal clique set. Hence, $C \supseteq P(\ell_1, \ell_2)$. □

Claim 2. *Any maximal clique set C of the graph $G(V, E)$ is a tree.*

Proof. It is clear from the definition that C is a sub-graph of $G(V, E)$. Since $G(V, E)$ is a tree, C cannot contain any loops. Lemma 2.3.1 implies that every two links in a maximal clique set are connected. It follows that C is a tree. □

We define $\tau(C) = \sum_{\ell \in C} \tau(\ell)$, for a maximal clique set C. It can be observed that $\tau(C)$ is the minimum time required to clear a maximal clique set C. And since C is a sub-graph of $G(V, E)$, $\tau(C)$ forms a lower bound on clearing time of the graph $G(V, E)$. We formally prove this using duality theory.

Claim 3. *$\tau(C)$ is a lower bound to the minimum clearing time of $G(V, E)$.*

Proof. Consider the dual-program of the minimum clearing time LP (1.1).

$$\max \sum_{\ell \in E} \tau(\ell).x_\ell$$

s.t.

$$\sum_{\ell:\ell \in S} x_\ell \leq 1, \forall S \in \mathcal{S}$$

$$x_\ell \geq 0, \forall \ell \in E \tag{2.2}$$

Observe that there is $1 - 0$ feasible solution to the dual given by a maximal clique set C:

$$x_\ell = \begin{cases} 1 & \text{if } \ell \in C, \\ 0 & \text{o.w.} \end{cases}$$

Since, any two links in a clique set interfere with each other, they cannot be present in the same maximal feasible set. As a result, at most one link $\ell \in C$ can be present in a maximal feasible set S. Thus, for every maximal feasible set $S \in \mathcal{S}$, under the current solution, $\sum_{\ell:\ell \in S} x_\ell \leq 1$. Therefore, the solution is feasible.

The value of the dual objective function under the solution is given by $\sum_{\ell \in C} \tau(\ell).1 = \tau(C)$. Using weak-duality theorem, $\tau(C)$ is a lower bound of the primal (1.1). □

Let $\mathscr{C}$ denote the set of all the maximal clique sets for the graph $G(V, E)$. Let $C^M := \text{argmax}_{C \in \mathscr{C}} \tau(C)$, denote the maximal clique set that has the maximum clearing time. We refer to $\tau(C^M)$ as the *clique time*. Using Claim 3, we can establish that minimum clearing time $\geq \tau(C^M)$. We refer to this bound as the *clique bound*.

2.4 Greedy Scheme

Explicitly solving the linear program (1.1) for the minimum clearing time requires the knowledge of all the maximal feasible sets S. Before even attempting to solve the linear program, one must construct the entire state space $\mathcal{S}$ containing maximal feasible sets for the given graph. It can be observed that the number of maximal feasible sets grows exponentially with number of links, even in the case of trees. e.g. For a linear chain of 30 links, under 1-hop

interference model, there are 4410 maximal feasible sets, the number grows to 1.22×10^6 for 50 links. Moreover, the linear program itself has exponentially many variables, due to the size of the state space. Therefore, traditional methods like the simplex algorithm are not suitable for solving this problem in general.

Due to the above reasons, we are interested in finding an alternate approach for solving the minimum time clearing problem. As it turns out, we can employ greedy methods that exploit the structure of trees and solve the minimum clearing time problem. The greedy scheme uses local information (within a maximal clique set) to allocate time to each link. To broadly describe the scheme, we start with a small tree and greedily allocate time to the links in this tree. Then we gradually start adding new links growing the size of the tree, every time new links are added the time is allocated greedily.

Before proceeding to explain the greedy scheme in context of a general tree. We use a special example, a chain of links, to illustrate key ideas behind the greedy scheme. Consider the chain of N links shown in Figure 2.2. The interference model is considered to be a 2-hop interference model. There is a load of $\tau(\ell_i)$ on a link ℓ_i.

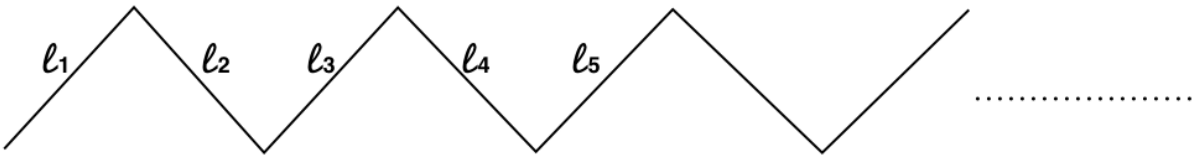


Figure 2.2: Chain of links

For this example, observe that any three consecutive links $\{\ell_i, \ell_{i+1}, \ell_{i+2}\}$ form a maximal clique set. Therefore, there are $N - 2$ maximal clique sets. Let $C_i = \{\ell_i, \ell_{i+1}, \ell_{i+2}\}$ denote a maximal clique set, then $\{C_i\}_{i=1:N-2}$ denotes all the maximal clique sets for Figure 2.2. Let $\mathcal{T}(\ell_i)$ denote the union of time intervals allocated to a link ℓ_i. Under this notation, the length of the time allocation is given by $|\mathcal{T}(\ell_i)| = \tau(\ell_i)$. For a set of links A, define $\mathcal{T}(A) = \bigcup_{\ell \in A} \mathcal{T}(\ell)$.

We describe the greedy scheme as follows. For links in $C_1 = \{\ell_1, \ell_2, \ell_3\}$, assign time as $\mathcal{T}(C_1) = (0, \tau(C_1)]$. Since $\tau(C_1) = \tau(\ell_1) + \tau(\ell_2) + \tau(\ell_3)$, the time allocation is feasible.

For the rest of the links, we define the allocation inductively as follows: Assuming the time allocation for $C_i = \{\ell_i, \ell_{i+1}, \ell_{i+2}\}$ is known, time allocation for $C_{i+1} = \{\ell_{i+1}, \ell_{i+2}, \ell_{i+3}\}$

has to be made.

Since, link ℓ_{i+3} interferes with both ℓ_{i+2} and ℓ_{i+1}, we allocate time to ℓ_{i+3} such that $\mathcal{T}(\ell_{i+3}) \subseteq (0, \tau(C_{i+1})] - \{\mathcal{T}(\ell_{i+1}) \bigcup \mathcal{T}(\ell_{i+2})\}$. We will show that the time allocation is feasible by showing that the time allocation $(0, \tau(C_{i+1})] - \{\mathcal{T}(\ell_{i+1}) \bigcup \mathcal{T}(\ell_{i+2})\}$ is longer than $\tau(\ell_{i+3})$.

$$\begin{aligned} |(0, \tau(C_{i+1})] - \{\mathcal{T}(\ell_{i+1}) \bigcup \mathcal{T}(\ell_{i+2})\}| &\geq \tau(C_{i+1}) - \tau(\ell_{i+1}) - \tau(\ell_{i+2}) \\ &= \tau(\ell_{i+3}) \end{aligned}$$

Thus, the algorithm assigns time to a link ℓ_{i+1} within the interval $(0, \tau(C_{i+1})]$. Thus the time allocation for all the links lies within the interval $(0, \max_{i=1:N-2} \tau(C_i)]$. Therefore, the greedy scheme clears the network in clique time, and hence it is optimal.

Observe that ℓ_i and ℓ_{i+3} do not interfere with each other, so allocating the same time resources to ℓ_i and ℓ_{i+3} does not violate any interference constraints. This kind of time reuse is the key idea behind the greedy scheme.

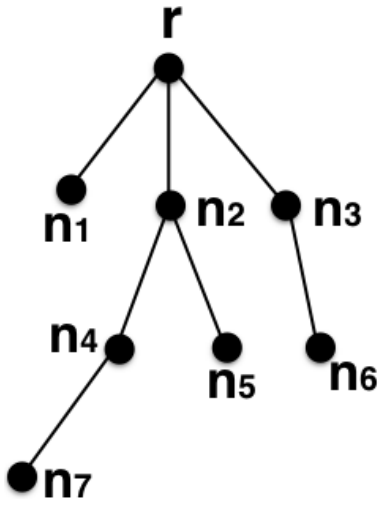


Figure 2.3: A rooted tree with r as the root

2.4.1 Algorithm

In this section, we describe the algorithm that solves the minimum clearing time problem in a tree. We describe the algorithm for disjoint cases, for a K-hop interference model: 1) K is odd and 2) K is even.

2.4.1.1 K is odd

We use the *rooted tree* $T(V, E, r)$ representation for the tree $G(V, E)$. In the rooted tree representation, a node $r \in V$ is chosen as the *root*. We now introduce new terms that will help in the discussion.

If node u is the neighbour of v in the path from v to r, then u is a parent of v and v is a child of u. Since the path from u to r is unique, every node has exactly one parent, expect for the root. For example, in Figure 2.3, n_2 is a parent of n_4 and n_5. The root only has children, it has no parent. We can similarly define an ancestor as follows: every node $u \neq v$ in the path from v to r is an ancestor of v, and v is a descendant of u. For example, in Figure 2.3, n_7 is a descendant of n_2, but not of n_3 or n_1. Observe that the root is an ancestor of every node in the tree, but a descendant of none. Let $V_1(u)$ denote the set of children of node u. For a set of nodes A, define $V_1(A) = \bigcup_{u \in A} V_1(u)$. We can inductively define $V_i(u) = V_1(V_{i-1}(u))$. The set $V_i(u)$ can be interpreted as the ith generation descendants of a node u. For a node u, let $\mathcal{L}(u)$ denote the link connecting u and its parent. For a set of nodes A, define $\mathcal{L}(A) = \bigcup_{u \in A} \mathcal{L}(u)$.

Now, we introduce some necessary notation before decribing the algorithm. $P(\ell, n)$ denotes the path from a link ℓ to a node n, including ℓ. $|P(\ell, n)|$ denotes the length of the path $P(\ell, n)$. $\mathcal{T}(\ell)$ denotes the union of time intervals allocated to a link ℓ. For a set of links A, $\tau(A) = \sum_{\ell \in A} \tau(\ell)$, and $\mathcal{T}(A) = \bigcup_{\ell \in A} \mathcal{T}(\ell)$. Let $\mathscr{C}$ denote the set of all maximal clique sets for the tree T. Define $C^M := \arg\max_{C \in \mathscr{C}} \tau(C)$. We now present Algorithm 1(see page 14).

It can be observed that the algorithm spans the whole tree. In the first step, by adding links $T^{(0)} = \bigcup_{m=1:(K+1)/2} \mathcal{L}(V_m(r))$, we add the first $(K+1)/2$ generation descendant nodes of the root, and the corresponding parental links. By iterating until $T^{(j)} \neq T$, we add all the subsequent generation descendants of the root in the tree T. Under Algorithm 1, the addition of links is exhaustive, and also exclusive in the sense that no link is added twice. Now, we prove some important results that will help in understanding how the algorithm works, and

Algorithm 1 K is odd

$i = 1, j = 1$;

$T^{(0)} = \bigcup_{m=1:(K+1)/2} \mathcal{L}(V_m(r))$;

$\mathcal{T}(T^{(0)}) = (0, \tau(T^{(0)})]$;

while $T^{(j)} \neq T$ **do**

  for $n \in V_i(r)$ **do**

    $T^{(j+1)} \leftarrow T^{(j)} \bigcup A^{(j+1)}$; where $A^{(j+1)} = \mathcal{L}(V_{(K+1)/2}(n))$.

    // $A^{(j+1)}$ is the set of links being newly added.

    Allocate time such that $\mathcal{T}(A^{(j+1)}) \subseteq (0, \tau(B^{(j+1)})] - \mathcal{T}(B^{(j+1)} - A^{(j+1)})$; where $B^{(j+1)} = \{\ell \in T^{(j+1)} : |P(n, \ell)| \leq (K+1)/2\}$.

    // $B^{(j+1)}$ is the maximal clique set containing $A^{(j+1)}$ in the tree $T^{(j+1)}$. This will be shown in Lemmas 2.4.2, 2.4.3.

    // Take any arbitrary union of intervals within $(0, \tau(B^{(j+1)})] - \mathcal{T}(B^{(j+1)} - A^{(j+1)})$ that has a length $\tau(A^{(j+1)})$, and allocate this time to links in $A^{(j+1)}$. In Theorem 2.4.4, we will show that such an allocation is always possible.

    $j \leftarrow j + 1$;

  end for

  $i \leftarrow i + 1$;

end while

also establish its optimality.

Lemma 2.4.1. *$T^{(0)}$ is a clique set.*

Proof. By definition, $T^{(0)} = \bigcup_{m=1:(K+1)/2} \mathcal{L}(V_m(r))$. Therefore, every link $\ell \in T^{(0)}$ satisfies $|P(\ell, r)| \leq (K+1)/2$. For any two links $\ell_1, \ell_2 \in T^{(0)}$, $|P(\ell_1, \ell_2)| \leq |P(\ell_1, r)| + |P(\ell_2, r)|$, equality occurs when node r is in the path $P(\ell_1, \ell_2)$. Therefore, $|P(\ell_1, \ell_2)| \leq K+1$. Therefore, any two links $\ell_1, \ell_2 \in T^{(0)}$ are with in a K-hop distance of each other, and hence interfere with each other. □

Lemma 2.4.2. *Assuming $A^{(j+1)}$ is non-empty, any link $\ell_1 \in A^{(j+1)}$ does not interfere with a link $\ell_2 \in T^{(j+1)} - B^{(j+1)}$, i.e., $|P(\ell_1, \ell_2)| > K + 1$.*

Proof. We divide the tree $T^{(j+1)}$ into two disjoint trees T_1 and T_2 by removing the link between the node n and its parent. T_1 is the tree that contains the node n and its $(K+1)/2$ generations of

descendants. And $T_2 = T^{(j+1)} - \{T_1 \bigcup \mathcal{L}(n)\}$. Since node n has at most $(K+1)/2$ generations of descendants in T_1, for any $\ell \in T_1$, $|P(\ell, n)| \leq (K+1)/2$. Therefore, $B^{(j+1)} \supseteq T_1 \supseteq A^{(j+1)}$.

By definition, $B^{(j+1)} = \{\ell \in T^{(j+1)} : |P(n, \ell)| \leq (K+1)/2\}$. Therefore, for any $\ell_2 \in T^{(j+1)} - B^{(j+1)}$, $|P(\ell_2, n)| > (K+1)/2$. From the previous arguments, we have $B^{(j+1)} \supseteq T_1 \supseteq A^{(j+1)}$. Therefore, $\ell_2 \notin T_1$, and hence $\ell_2 \in T_2$. Now choose any link $\ell_1 \in A^{(j+1)}$, from the definition of $A^{(j+1)}$, $|P(\ell_1, n)| = (K+1)/2$. Now since $\ell_1 \in T_1$, and $\ell_2 \in T_2$, the path $P(\ell_1, \ell_2)$ must contain node n. Therefore, $|P(\ell_1, \ell_2)| = |P(\ell_1, n)| + |P(n, \ell_2)| > K+1$. Hence, any link $\ell_1 \in A^{(j+1)}$ does not interfere with a link $\ell_2 \in T^{(j+1)} - B^{(j+1)}$. □

Lemma 2.4.3. *$B^{(j+1)}$ is a maximal clique set of $T^{(j+1)}$.*

Proof. From definition, $B^{(j+1)} = \{\ell \in T^{(j+1)} : |P(n, \ell)| \leq (K+1)/2\}$. Therefore, for any two links $\ell_1, \ell_2 \in B^{(j+1)}$, $|P(\ell_1, \ell_2)| \leq |P(\ell_1, n)| + |P(\ell_2, n)| \leq K+1$. Therefore, any two links $\ell_1, \ell_2 \in B^{(j+1)}$ are with in a K-hop distance of each other, and hence interfere with each other. Therefore, $B^{(j+1)}$ is a clique set. Now, using Lemma 2.4.2, $B^{(j+1)}$ is a maximal clique set of $T^{(j+1)}$. □

Theorem 2.4.4. *Under Algorithm 1, the time allocation is feasible and lies within the interval $(0, \tau(C^M)]$.*

Proof. We use proof by induction. Using Lemma 2.4.1, $(0, \tau(C^M)] \supseteq (0, \tau(T^{(0)})]$. Since, $\tau(T^{(0)}) = \sum_{\ell \in T^{(0)}} \tau(\ell)$, each link gets enough time. Therefore, the time allocation for $T^{(0)}$ is feasible, and lies within the interval $(0, \tau(C^M)]$.

Assume that the time allocation for the links in $T^{(j)}$ is feasible, and lies within the interval $(0, \tau(C^M)]$. Now, the set of new links $A^{(j+1)}$ are added to the tree $T^{(j)}$, and the tree evolves into $T^{(j+1)} = T^{(j)} \bigcup A^{(j+1)}$. We will show that the time allocation for $A^{(j+1)}$ is feasible and lies within the interval $(0, \tau(C^M)]$, and hence showing that the time allocation for $T^{(j+1)}$ is feasible, and lies within the interval $(0, \tau(C^M)]$.

Upon addition of links in $A^{(j+1)}$, the time for the new links under Algorithm 1 is allocated as $\mathcal{T}(A^{(j+1)}) \subseteq (0, \tau(B^{(j+1)})] - \mathcal{T}(B^{(j+1)} - A^{(j+1)})$. Using Lemma 2.4.2, we have established that $B^{(j+1)}$ is a maximal clique set of $T^{(j+1)}$. Hence, $B^{(j+1)}$ must be a clique set of $T(V, E, r)$. Therefore, $(0, \tau(B^{(j+1)})] \subseteq (0, \tau(C^M))$.

Using Lemma 2.4.2, we have established that a link in $A^{(j+1)}$ only interferes with the links in $B^{(j+1)}$. As a result, the time that is unavailable due to interference for links in $A^{(j+1)}$ is

given by $\mathcal{T}(B^{(j+1)} - A^{(j+1)})$. As can be observed, the algorithm does not use the unavailable time: $\mathcal{T}(A^{(j+1)}) \subseteq (0, \tau(B^{(j+1)})] - \mathcal{T}(B^{(j+1)} - A^{(j+1)})$. Now we will show that algorithm allocates enough time by calculating length of the time allocation

$$\begin{aligned}|(0, \tau(B^{(j+1)})] - \mathcal{T}(B^{(j+1)} - A^{(j+1)})| \geq & \tau(B^{(j+1)}) - \tau(B^{(j+1)} - A^{(j+1)}) \\ = & \sum_{\ell \in B^{(j+1)}} \tau(\ell) - \sum_{\ell \in B^{(j+1)} - A^{(j+1)}} \tau(\ell) \\ = & \sum_{\ell \in A^{(j+1)}} \tau(\ell)\end{aligned}$$

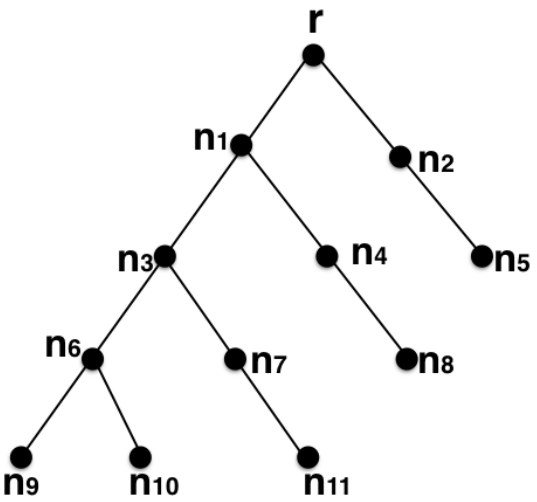
□

Let us look at an example: Consider the tree shown in Figure 2.4(a), under the 3-hop interference model. For this example, $T^{(0)}$ is shown in Figure 2.4(b). For $n = n_1$, $A^{(1)} = \mathcal{L}(\{n_6, n_7, n_8\})$ and $T^{(1)} = T^{(0)} \bigcup A^{(1)}$. The sets $A^{(1)}$, $B^{(1)}$ and $T^{(1)}$ are shown in Figure 2.4(c). Since n_2 does not have any 3rd generation descendants, for $n = n_2$, $A^{(1)} = V_2(n_2) = \phi$. Therefore, no time needs to be allocated to $A^{(2)}$, and $T^{(2)} = T^{(1)}$. For $n = n_3$, $A^{(3)} = \mathcal{L}(\{n_9, n_{10}, n_{11}\})$ and $T^{(3)} = T^{(2)} \bigcup A^{(3)}$. The sets $A^{(3)}$, $B^{(3)}$ and $T^{(3)}$ are shown in Figure 2.4(d). Since $T^{(3)} = T$, the algorithm terminates.

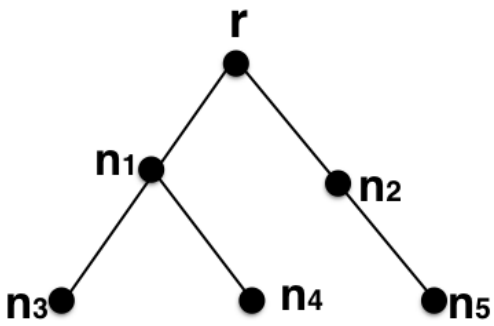
2.4.1.2 K is even

For this case, we use a slight modification of the rooted tree model. In this case, we have two roots r_1, r_2 that are joined by a link, say ℓ_0. It can be interpreted as the union of two rooted trees, joined by a link between their roots. e.g., see Figure 2.5. Let $\pi(n)$ denote the parent of a node n. Every node has an unique parent except for the roots r_1, r_2. The rest of the terminology and notation is carried over from the previous section. Now the algorithm is described in Algorithm 2 (see page 18).

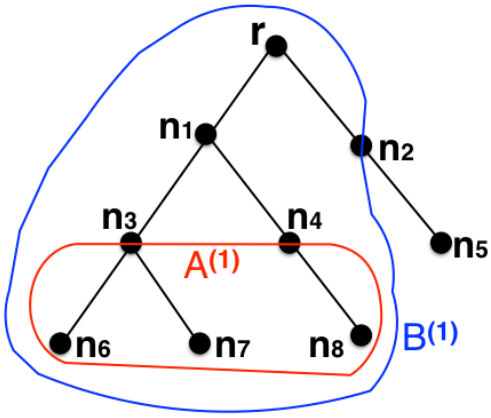
Similar arguments used in the previous case can establish the optimality of the algorithm. The Lemmas 2.4.1, 2.4.2, 2.4.3, and Theorem 2.4.4 can also be proved under Algorithm 2. Due to the strict page limit of this thesis, we will not repeat the arguments here.



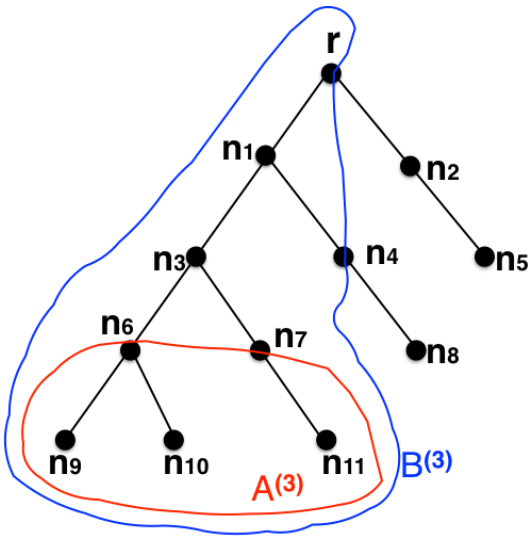
(a) T



(b) $T^{(0)}$



(c) $T^{(1)}$



(d) $T^{(3)}$

Figure 2.4: Algorithm example

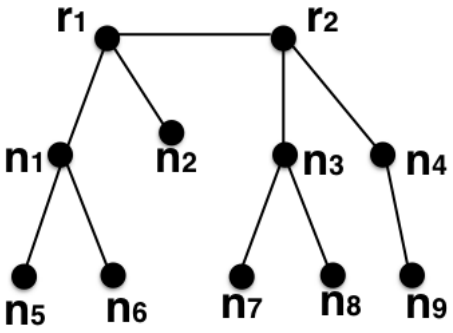


Figure 2.5: A tree with two roots r_1 and r_2

Algorithm 2 K is even

$i = 1, j = 1$;
$T^{(0)} = \bigcup_{m=1:K/2} \mathcal{L}(V_m(r_1)) \bigcup_{m=1:K/2} \mathcal{L}(V_m(r_2)) \bigcup \ell_0$;
$\mathcal{T}(T^{(0)}) = (0, \tau(T^{(0)})]$;
while $T^{(j)} \neq T$ **do**
    for $n \in V_i(r_1) \bigcup V_i(r_2)$ **do**
        $T^{(j+1)} \leftarrow T^{(j)} \bigcup A^{(j+1)}$;
where $A^{(j+1)} = \mathcal{L}(V_{K/2}(n))$
        Allocate time such that $\mathcal{T}(A^{(j+1)}) \subseteq (0, \tau(B^{(j+1)})] - \mathcal{T}(B^{(j+1)} - A^{(j+1)})$;
where $B^{(j+1)} = \{\ell \in T^{(j+1)} : |P(n, \ell)| \leq K/2\} \bigcup \{\ell \in T^{(j+1)} : |P(\pi(n), \ell)| \leq K/2\}$
        $j \leftarrow j + 1$;
    end for
    $i \leftarrow i + 1$;
end while

We conclude that the clique bound on the minimum clearing time is tight for tree networks under the K-hop interference model. We have also provided a greedy algorithm that solves the minimum clearing time problem in such graphs.

2.5 Applications

In this section, we apply the results on tree networks to a few particular, but interesting wireless networks, which are special cases of networks known as Heterogeneous Networks (HetNets).

2.5.1 Single Cell HetNet

Consider a setup shown in Figure 2.6 with a macro basestation M and a set of pico basestations $\mathcal{P} = \{P_1, P_2, ...P_N\}$. We consider the minimum clearing time problem in a downlink scenario. We assume that all the pico basestations in $\mathcal{P}$ are connected to the core network via the macro M using wireless backhaul. Let $\mathcal{U}_i = \{U_{i,1}, U_{i,2}, ...U_{i,|\mathcal{U}_i|}\}$ denote the set of users being served by the pico basestation P_i. Let $\mathcal{U}_0 = \{U_{0,1}, U_{0,2}, ...U_{0,|\mathcal{U}_0|}\}$ denote the set of users being served

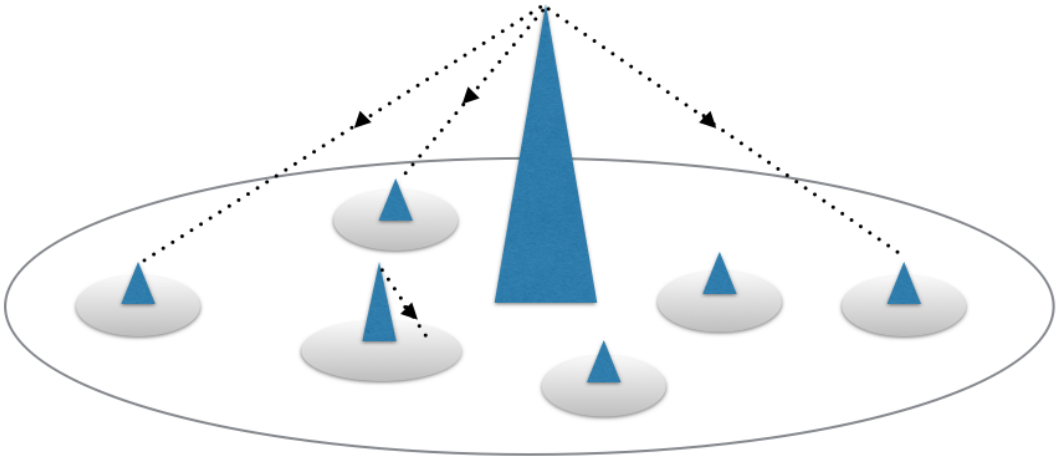


Figure 2.6: Single cell HetNet

by the macro basestation M. We assume that each user only associates to a single basestation. We assume that a pico cell does not experience any co-tier interference from the other pico cells. For this setup, the network graph is shown in Figure 2.7. Observe that the graph shown in Figure 2.7 is a tree. Also observe that the interference constraints for this setup can be characterized using a 2-hop interference model.

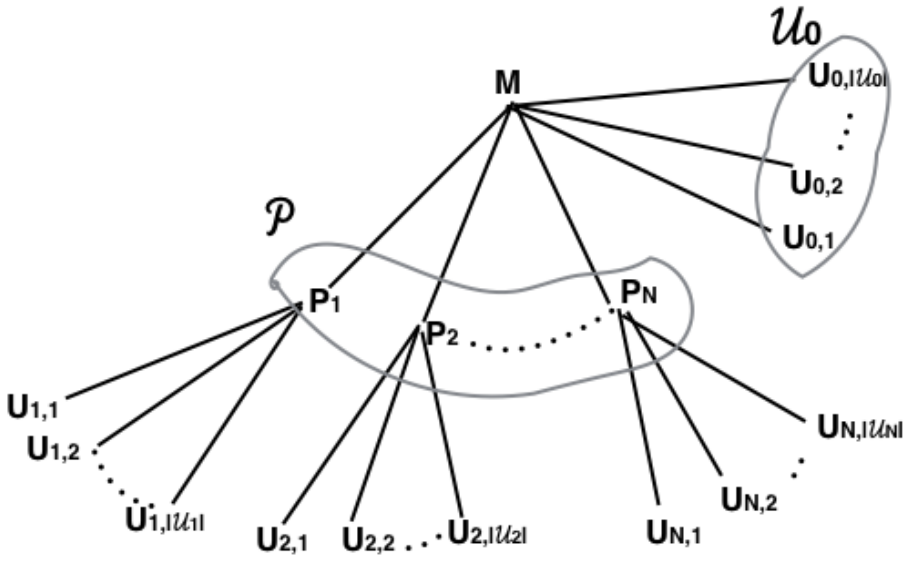


Figure 2.7: HetNet graph

For the graph shown in Figure 2.7, there are exactly N maximal clique sets, one involving (user links of) each pico cell P_i. The maximal clique set involving the pico cell P_i is shown

in Figure 2.8.

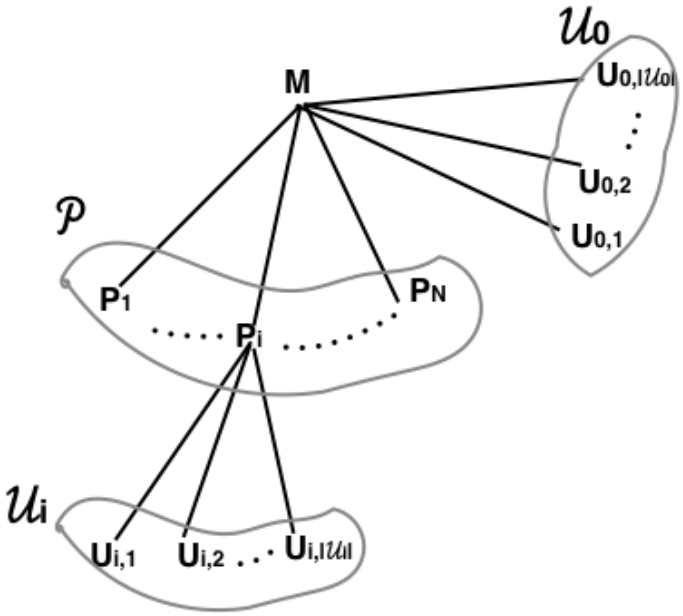


Figure 2.8: A Maximal clique set for the HetNet graph shown in Figure 2.7

Let $\tau(\ell)$ denote the time required to clear a link ℓ. The minimum clearing time for a single cell HetNet is given by the clique time = $\max_{i=1:N}\{\sum_{j=1:|\mathcal{U}_i|}\tau(P_iU_{i,j})\}+\sum_{i=1}^{N}\tau(M_iP_i)+\sum_{j=1}^{|\mathcal{U}_0|}\tau(MU_{0,j})$. The first term in the sum $\max_{i=1:N}\{\sum_{j=1:|\mathcal{U}_i|}\tau(P_iU_{i,j})\}$ is the time required to clear all the transmissions within the pico cells. Since pico cells do not face any co-tier interference, they can all transmit at the same time. So, the time required is determined by the bottleneck among the pico cells. The other two terms in the sum $\sum_{i=1}^{N}\tau(M_iP_i)+\sum_{j=1}^{|\mathcal{U}_0|}\tau(MU_{0,j})$ is the time required to clear the macro cell transmissions. Since a transmission from the macro cell interferes with a transmission from any pico cell, they cannot happen at the same time.

Notice that the considered model can be modified to include pico cells with wired backhaul by setting $\tau(M_1P_i)=0$. By choosing $\tau(M_1P_i)=0$, we are assuming that the backhaul has an infinite rate. Since the cross-tier interference constraint supplants the half-duplex constraint, the model is still valid for a setup where pico cells have wired backhaul.

The minimum clearing time scheme in a single cell HetNet can be realized using the Almost Blanking Subframes (ABS) scheme introduced as part of the interference mitigation eICIC in the 3GPP project. During the ABS subframes, the macro basestation does not

transmit any information, allowing the pico cells to transmit at the maximum rate. By setting the fraction of the ABS subframes based on the solution of the minimum clearing time problem, the minimum clearing time can be achieved. The minimum clearing time for the wired backhaul case was studied in [10, 11]. The authors have further optimized the clearing time with respect to user-cell association, and have derived the optimal user-cell association for this problem.

2.5.2 Two Cell HetNet

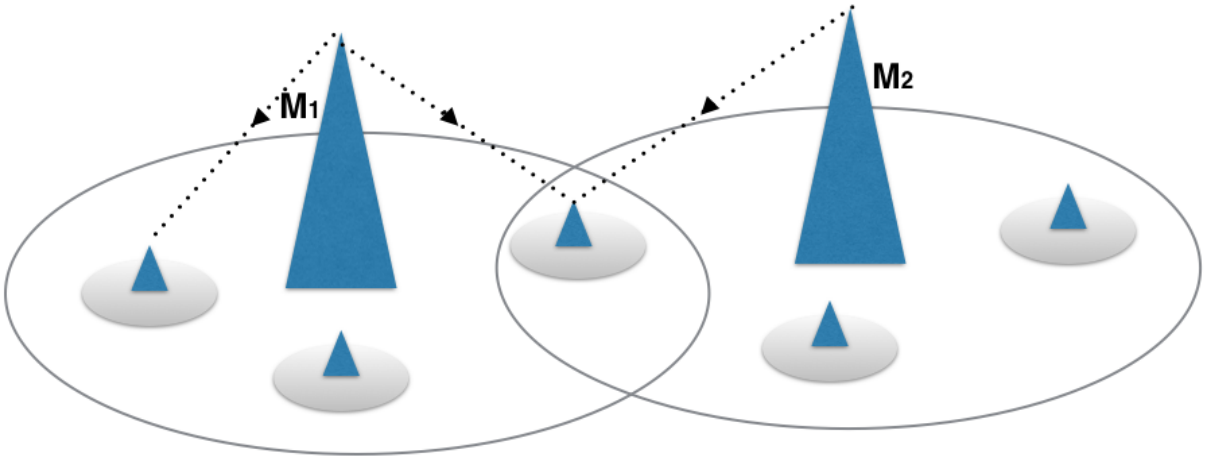


Figure 2.9: Two cell HetNet

Consider a model similar to earlier setup, but this time with two macro BSs M_1,M_2, shown in Figure 2.9. The pico cells are divided into three disjoint sets: $\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_0\}$. $\mathcal{P}_1 = \{P_{1,1}, P_{1,2}, .., P_{1,i}....\}$ represents the set of pico cells that are within the range of M_1, but not M_2. Similarly $\mathcal{P}_2 = \{P_{2,1}, P_{2,2}, .., P_{2,i}....\}$ represents the set of pico cells that are within the range of M_2, but not M_1. $\mathcal{P}_0 = \{P_{0,1}, P_{0,2}, .., P_{0,i}....\}$ represents the set of pico cells that are in the range of both M_1 and M_2. We assume that pico cells are connected to the core network via the macro basestations using wireless backhaul. The pico cells in $\mathcal{P}_0$ can be served by either M_1 or M_2. Similarly, pico cells in $\mathcal{P}_0$ face interference from both M_1 and M_2, and can only transmit data to users when both the macros are silent. $\mathcal{U}_{i,j}$ denotes the set of users that are being served by the pico cell $P_{i,j}$. We do not consider the macro exclusive users for the sake of simplicity, even though including them does not change any of the main

results. Observe that the interference constraints for this setup can still be characterized using a 2-hop interference model.

There are 4 types of maximal clique sets for this network, they are shown in Figure 2.10. The clique bound on the clearing time can be calculated as :

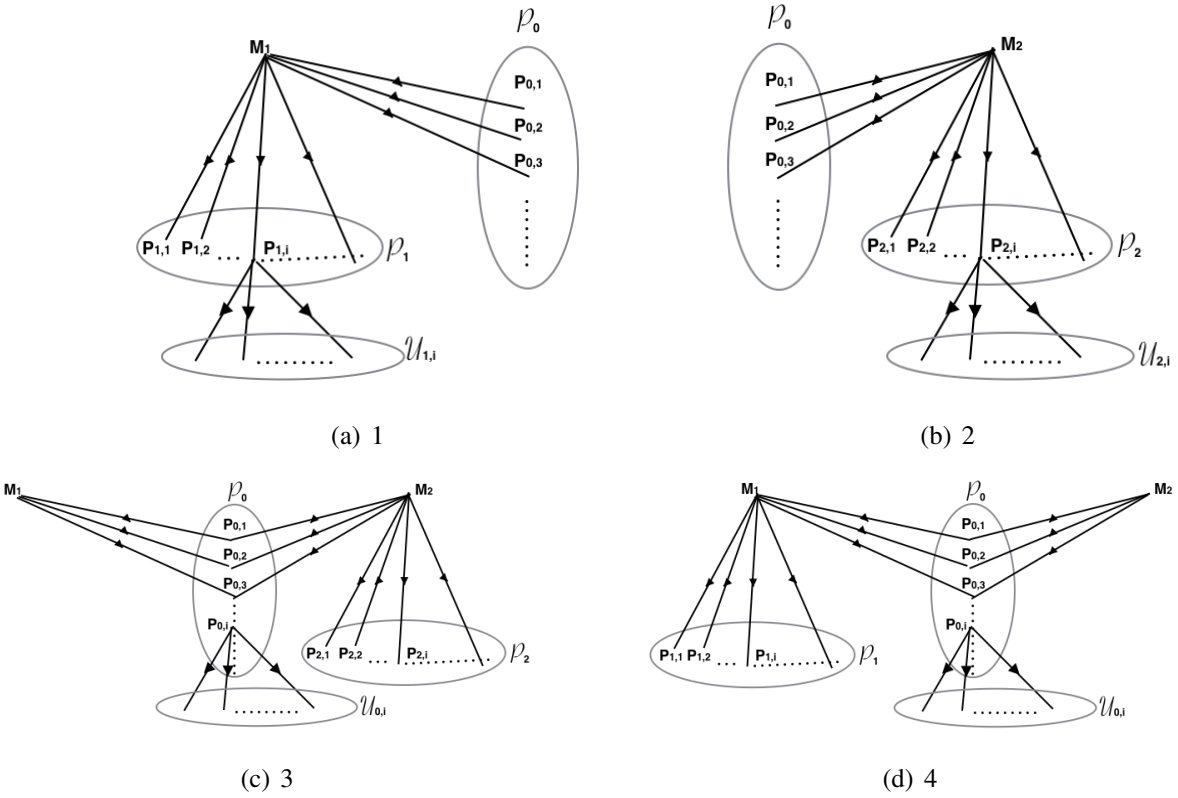
$$\begin{aligned}
\max\{ & \sum_{i:P_{1,i}\in\mathcal{P}_1} \tau(M_1P_{1,i}) + \sum_{i:P_{0,i}\in\mathcal{P}_0} \tau(M_1P_{0,i}) + \max_{i:P_{1,i}\in\mathcal{P}_1}\{\sum_{U\in\mathcal{U}_{1,i}} \tau(P_{1,i}U)\}, \\
& \sum_{i:P_{2,i}\in\mathcal{P}_2} \tau(M_2P_{2,i}) + \sum_{i:P_{0,i}\in\mathcal{P}_0} \tau(M_2P_{0,i}) + \max_{i:P_{2,i}\in\mathcal{P}_2}\{\sum_{U\in\mathcal{U}_{2,i}} \tau(P_{2,i}U)\}, \\
& \sum_{i:P_{1,i}\in\mathcal{P}_1} \tau(M_1P_{1,i}) + \sum_{i:P_{0,i}\in\mathcal{P}_0} \tau(M_1P_{0,i}) + \sum_{i:P_{0,i}\in\mathcal{P}_0} \tau(M_2P_{0,i}) + \max_{i:P_{0,i}\in\mathcal{P}_0}\{\sum_{U\in\mathcal{U}_{0,i}} \tau(P_{0,i}U)\}, \\
& \sum_{i:P_{2,i}\in\mathcal{P}_2} \tau(M_2P_{2,i}) + \sum_{i:P_{0,i}\in\mathcal{P}_0} \tau(M_1P_{0,i}) + \sum_{i:P_{0,i}\in\mathcal{P}_0} \tau(M_2P_{0,i}) + \max_{i:P_{0,i}\in\mathcal{P}_0}\{\sum_{U\in\mathcal{U}_{0,i}} \tau(P_{0,i}U)\}\}
\end{aligned}$$

For the sake of brevity, we re-write these expressions as follows:

$$\begin{aligned}
\max\{ & \tau(M_1\mathcal{P}_1) + \tau(M_1\mathcal{P}_0) + \tau(\mathcal{P}_1\mathcal{U}_1), \\
& \tau(M_2\mathcal{P}_2) + \tau(M_2\mathcal{P}_0) + \tau(\mathcal{P}_2\mathcal{U}_2), \\
& \tau(M_1\mathcal{P}_1) + \tau(M_1\mathcal{P}_0) + \tau(M_2\mathcal{P}_0) + \tau(\mathcal{P}_0\mathcal{U}_0), \\
& \tau(M_2\mathcal{P}_2) + \tau(M_1\mathcal{P}_0) + \tau(M_2\mathcal{P}_0) + \tau(\mathcal{P}_0\mathcal{U}_0)\}
\end{aligned}$$

In this new notation, each term represents the aggregate load of a particular set, e.g. $\tau(M_1\mathcal{P}_1) = \sum_{i:P_{1,i}\in\mathcal{P}_1} \tau(M_1P_{1,i})$, and $\tau(\mathcal{P}_1\mathcal{U}_1) = \max_{i:P_{1,i}\in\mathcal{P}_1}\{\sum_{U\in\mathcal{U}_{1,i}} \tau(P_{1,i}U)\}$. This motivates us to simplify the HetNet model by using a single pico cell in each set $\mathcal{P}_i$ and a single user in each set $\mathcal{U}_i$. The load on the links in the new model can be thought to represent the aggregate load of the whole set. This simplified model will also help better illustrate the next problem.

Using this model, we will illustrate the benefit of co-ordinating scheduling decisions between the two cells. The results may also provide insight into how to run the ABS scheme when there is interference from more than one macro basestation.



(a) 1 (b) 2

(c) 3 (d) 4

Figure 2.10: Maximal clique sets for the two cell HetNet

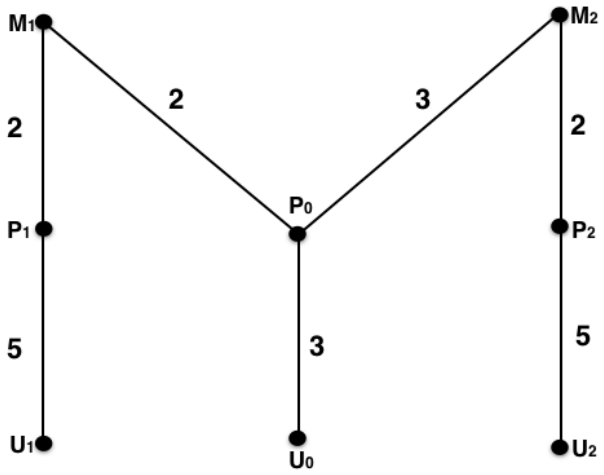
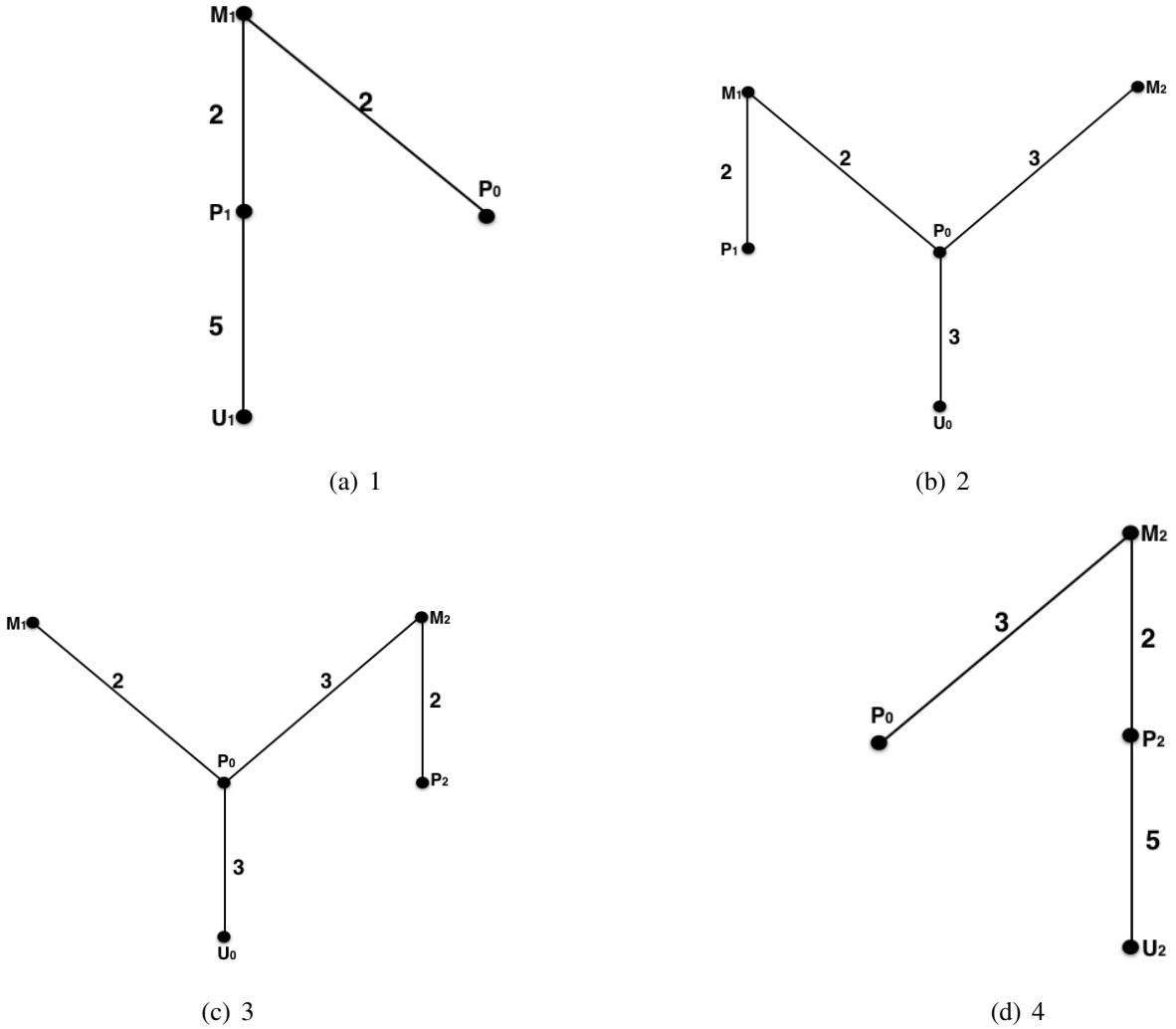


Figure 2.11: Two cell HetNet example

We will use a simple example with a fixed load on various links in the network. The considered model with loads is shown in Figure 2.11. The value of the loads represented in Figure 2.11 are in ms. As explained earlier, each load represents the aggregate value. For example, $\tau(M_1P_0)$ is the aggregate backlog of transmissions of the macro M_1 to the cell-edge

region of M_1 and M_2. $\tau(P_1U_1)$ is maximum transmission backlog among the pico cells only in range of M_1. Under this assumption, the maximal clique sets are shown in Figure 2.12. The clique time is given by $\max\{9, 10, 10, 10\} = 10$ms. Under the LTE standards, each frame has 10 sub-frames, each with a value of 1ms. Our results show that the network in Figure 2.11 can be cleared in 1 frame.



(a) 1 (b) 2

(c) 3 (d) 4

Figure 2.12: Maximal clique sets for the two cell HetNet in Figure 2.11

We use three different methods to clear the given load: 1) Joint Greedy ABS Scheme, 2) Joint Fixed ABS Scheme, and 3) Frequency Sharing on the Cell-Edge.

2.5.2.1 Joint Greedy ABS Scheme

In this scheme, we will try to emulate the greedy scheme discussed earlier in this chapter, using the ABS scheme with co-operation between macros M_1 and M_2. We assume macros M_1 and M_2 are synchronized and can exchange their ABS sub-frame values with each other. Under this assumption, the greedy scheme can be realized. For the example in Figure 2.11, Macro M_1 looks at the load with in its range, and sees a macro load of 4ms and a pico load of 5ms. Hence, M_1 can choose the ABS values for the frame as follows. N represents a normal sub-frame, and A represents an ABS sub-frame.

N	N	N	N	A	A	A	A	A	A

Similarly, M_2 sees a macro load of 5ms and a pico load of 5ms, and chooses 5 normal subframes and 5 ABS subframes. Since M_1P_1 and M_2P_2 can use the same time resource, M_2 chooses 2 normal subframes overlapping with the normal subframes of M_1. Since M_2P_0 cannot use same time as any macro transmission of M_1, 3 normal subframes of M_2 should overlap with ABS subframes of M_1. Therefore, relative to the ABS choice of M_1, M_2 can choose its frame as follows

N	N	A	A	N	N	N	A	A	A

Under this choice of ABS subframes, we will show that the network can be cleared in one frame.

	1	2	3	4	5	6	7	8	9	10
M_1:	N	N	N	N	A	A	A	A	A	A
M_2:	N	N	A	A	N	N	N	A	A	A

During subframes 1 and 2, the links M_1P_1 and M_2P_2 are cleared. During sub-frames 3 and 4, the links M_1P_0 is cleared and P_2U_2 clears a traffic of 2ms. During sub-frames $5-7$, the link M_2P_0 is cleared and a traffic of 3ms is cleared from P_1U_1. During sub-frames $8-10$, the links P_1U_1, P_0U_0, P_2U_2 are cleared. Therefore with some co-operation, the scheduling decisions can be co-ordinated between cells to clear the network in minimum time.

2.5.2.2 Joint Fixed ABS scheme

We consider a scheme similar to the one proposed in [13]. Although, the salient feature of [13] is the joint optimization of user-association and interference mitigation, we are only interested in the interference mitigation part achieved through ON-OFF scheduling. Our model does not have interference between pico-basestations as considered in [13]. Macro basestations are assumed to be synchronized in [13]. Our main focus is on the assumption that all the macro basestations transmit simultaneously. Due to this assumption, a macro basestation and pico basestation never transmit simultaneously, which could be inefficient.

Under the scheme proposed in [13], the macro cell transmissions and pico-cell transmissions are cleared at different times. The time to clear macro cell transmissions in Figure 2.11 is given by $\max\{\tau(M_1P_1), \tau(M_2P_2)\} + \tau(M_1P_0) + \tau(M_2P_0) = 7$ms. The time to clear pico cell transmissions is given by $\max\{\tau(P_1U_1), \tau(P_2U_2), \tau(P_0U_0)\} = 5$ms. The time required to clear the network under this scheme is given by 12ms. So, in this scheme, even with co-ordination and synchronization, the network could not be cleared in 1 frame. The scheme requires 2 extra sub-frames to clear all the traffic.

2.5.2.3 Frequency Sharing on the Cell Edge

In this scheme, the macro basestations use the whole frequency band for transmissions within the cell, where as on the cell-edge, the macro basestations M_1 and M_2 share the frequency band. The macro basestations M_1 and M_2 use orthogonal frequencies for transmissions on the cell-edge so as to eliminate the interference. This scheme is similar to the fractional frequency reuse (FFR) schemes proposed in the literature.

By sharing the frequency band on the cell-edge, the rate of the links on the cell-edge decreases. This means the clearing times of the traffic on the cell-edge increase inversely proportional to the fraction of the frequency band used. Assuming infinite divisibility of the frequency band, let M_1 use an β proportion of the available frequency band on the cell-edge, and let M_2 use the other $1 - \beta$ proportion of the band. The new values of the load under the frequency sharing scheme is shown in Figure 2.13.

Due to the frequency sharing, the links M_1P_0 and M_2P_0 no longer interfere. The maximal clique sets in this case are given by $\{M_1P_1, P_1U_1, M_1P_0\}$, $\{M_2P_2, M_2P_0, P_2U_2\}$,

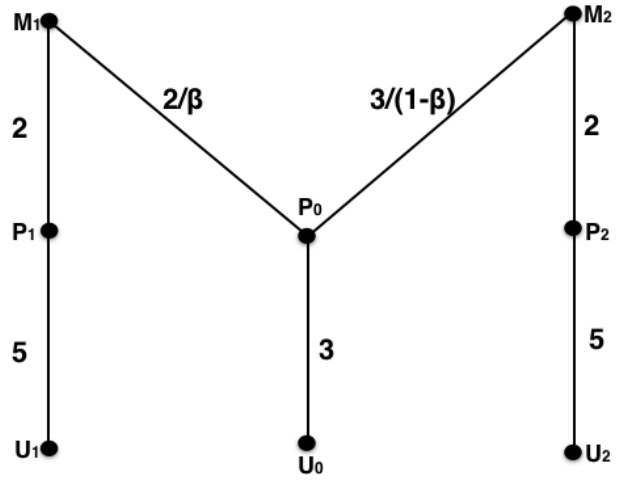


Figure 2.13: Value of the loads with frequency sharing on the cell-edge

$\{M_1P_1, M_1P_0, P_0U_0\}$ and $\{M_2P_2, M_2P_0, P_0U_0\}$. Therefore, the minimum clearing time is given by $\max\{7 + 2/\beta, 7 + 3/(1 - \beta)\}$ms.

The scheduling scheme which can achieve minimum time clearing is very straightforward. M_1P_1 and M_2P_2 are cleared together, which takes 2ms. Clearing link P_0U_0 takes 3ms, during this time links P_1U_1, P_2U_2 also clear a traffic of 3ms. Since M_1P_0 and M_2P_0 do not interfere, the rest of the transmissions in each cell can happen independently of the transmissions in the other cell. The time to complete the remaining transmissions can be calculated as $\max\{2 + 2/\beta, 2 + 3/(1 - \beta)\}$ms. Therefore, the total time is given by $\max\{7 + 2/\beta, 7 + 3/(1 - \beta)\}$ms.

Observe that the value of $\max\{7+2/\beta, 7+3/(1-\beta)\}$ is minimized at $\beta = 2/5$. Therefore, $\max\{7+2/\beta, 7+3/(1-\beta)\} \geq 12$, which shows that frequency sharing on the cell-edge (even when done in an optimal manner) is sub-optimal compared to the greedy scheme.

2.5.3 Chain of HetNets

Consider the network shown in Figure 2.14. At the centre of each cell i is a macro basestation M_i. A pico basestation P_i is exclusively present in cell i, and experiences interference only from the macro basestation M_i. Each pico cell P_i is serving a single user U_i. A pico basestation $P_{i,i+1}$ is present on the cell-edge of cell i and cell $i + 1$. The pico basestation $P_{i,i+1}$ receives data, and experiences interference from both the macros M_i and M_{i+1}. And

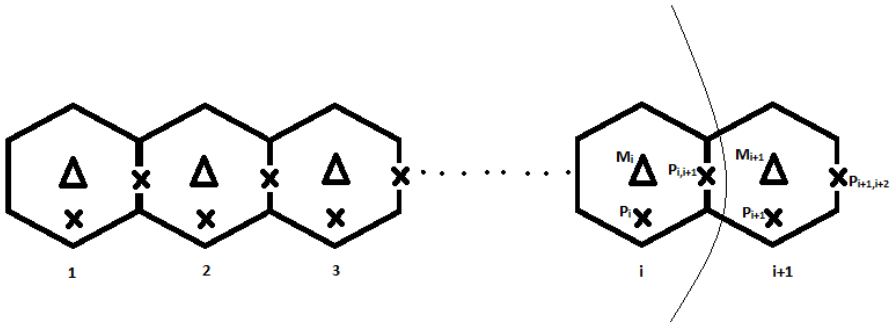


Figure 2.14: Chain of HetNets

each pico cell $P_{i,i+1}$ is serving a single user $U_{i,i+1}$. We assume that pico cells are connected to the core network via the macro basestations using wireless backhaul, and that pico cells do not experience any co-tier interference. Assume that there are a total of N cells in the chain. This model is the HetNet analogue of the Wyner Model [14] for cellular networks. It can be noted that even though we are solving the problem for a chain, the result can also be extended to tree like arrangements involving cells. The graph model for this network is shown in Figure 2.15. Observe that the interference constraints for this setup can still be characterized using a 2-hop interference model.

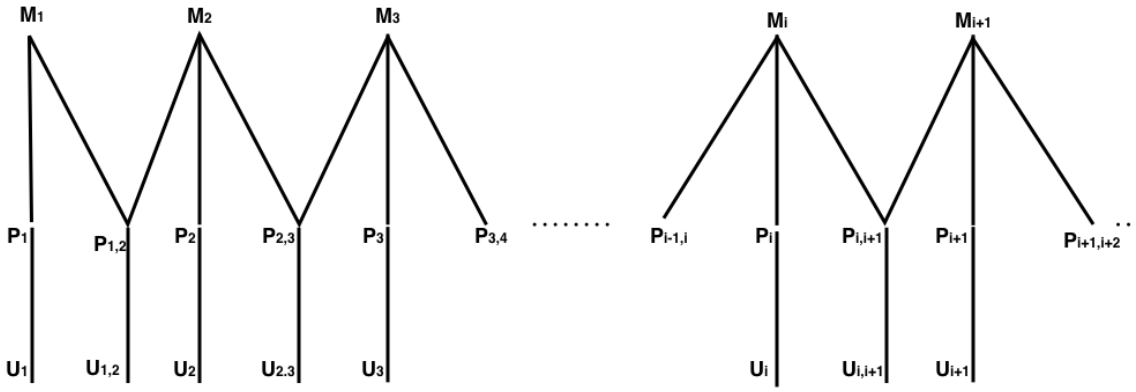
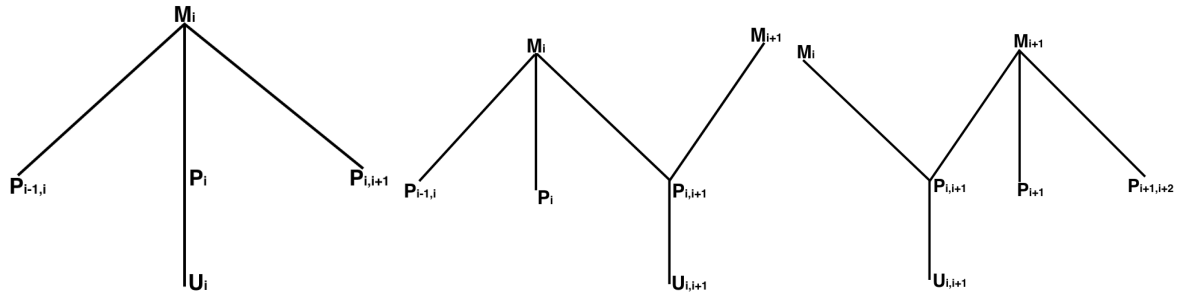


Figure 2.15: Graph for the chain of HetNets

There are three types of maximal clique sets for this network. They are shown in Figure 2.16. There are $3N-2$ maximal clique sets for the considered network. Let us denote them as $\{C_k\}_{k=1:3N-2}$ going from cell 1 to cell N. See Figure 2.16. Define $\tau(C^M) = \max_{k=1:3N-2}\{\tau(C_k)\}$. Let $\mathcal{T}(\ell)$ denote the union of time intervals allocated to a link ℓ,



(a) C_{3i-2} (b) C_{3i-1} (c) C_{3i}

Figure 2.16: Maximal clique sets for the HetNet chain

observe that $|\mathcal{T}(\ell)| = \int_{\mathcal{T}(\ell)} \mathrm{dt} = \tau(\ell)$. For a clique set C, define $\mathcal{T}(C) = \bigcup_{\ell \in C} \mathcal{T}(\ell)$. The greedy algorithm can be used to solve the minimum clearing time problem. Allocate time to links in C_1 from the interval $(0, \tau(C_1)]$, i.e., $\mathcal{T}(C_1) = (0, \tau(C_1)]$. Assume that time allocation for links in C_k is known. Using the greedy algorithm, we allocate time to links in $C_{k+1} - C_k$ as follows

if $\tau(C_{k+1}) \leq \tau(C_k)$

Allocate time from the set $\mathcal{T}(C_k - C_{k+1})$ to the links in $C_{k+1} - C_k$. Such an allocation is possible since $\tau(C_{k+1} - C_k) \leq \tau(C_k - C_{k+1})$.

if $\tau(C_{k+1}) > \tau(C_k)$

Assign time from the set $\mathcal{T}(C_k - C_{k+1})$ to the links in $C_{k+1} - C_k$. There is still a residual demand of $\tau(C_{k+1}) - \tau(C_k)$ in $C_{k+1} - C_k$. For this residual demand, time can be allocated from $(0, \tau(C^M)] - \mathcal{T}(C_k)$. Since $\tau(C^M) - \tau(C_k) \geq \tau(C_{k+1}) - \tau(C_k)$, such an allocation is possible.

As shown earlier with the two cell HetNet case, realizing minimum clearing time scheme is possible by co-ordinating the ABS scheme between neighbouring macro basestations. This still holds true for the chain of HetNets. e.g., Under the greedy scheme, the time resources used by $C_{3i-2} - C_{3i-1} = \{P_i U_i\}$ should be reused by $C_{3i-1} - C_{3i-2} = \{M_{i+1}P_{i,i+1}, P_{i,i+1}U_{i,i+1}\}$. This can be achieved using co-operation between the cell i and the cell $i + 1$.

Chapter 3

Ring Networks

3.1 Introduction

In the previous chapter, we have solved the minimum clearing time problem in tree networks. For tree networks, the minimum clearing time is equal to the clique time. We have also shown that a distributed greedy algorithm that allocates time to a link based on only local information is optimal. At this point, one would like to understand the consequences of introducing a loop into the graph. Is the clique bound still tight? Does some greedy algorithm still solve the minimum clearing problem? As it turns out, the clique bound is not necessarily tight in presence of loops. We demonstrate this with a simple counter example: Consider the pentagon shown in Figure 3.1, where each edge ℓ_i represents a link. We assume that each link interferes with its immediate neighbours i.e., 1-hop interference model. Each link has a load of 1 unit i.e., $\tau(\ell_i) = 1$.

Maximal feasible sets for this setup are given by $S_1 = \{\ell_1, \ell_3\}, S_2 = \{\ell_2, \ell_4\}, S_3 = \{\ell_3, \ell_5\}, S_4 = \{\ell_4, \ell_1\}, S_5 = \{\ell_5, \ell_2\}$. Let f_i denote the time allocated to a feasible set S_i. The minimum clearing time problem can be formulated as:

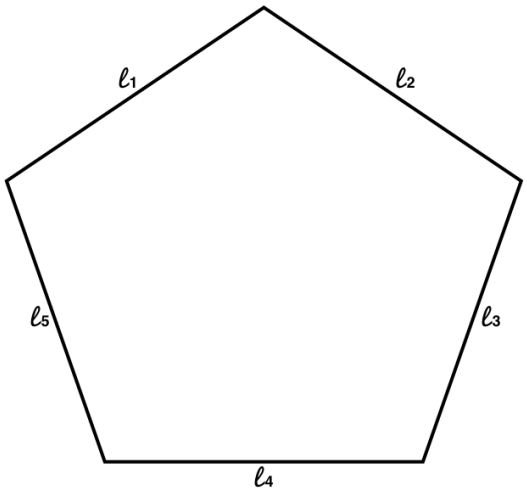


Figure 3.1: Ring network: pentagon

$$\min \sum_{i=1}^{5} f_i$$

s.t.

$$f_1 + f_3 \geq 1$$

$$f_2 + f_4 \geq 1$$

$$f_3 + f_5 \geq 1$$

$$f_4 + f_1 \geq 1$$

$$f_5 + f_2 \geq 1$$

Any two adjacent links in the network form a maximal clique set. Hence, clique bound on the network is given by 2 units. Observe that by adding the constraints of the LP, we obtain the following inequality.

$$2 \sum_{i=1}^{5} f_i \geq 5.$$

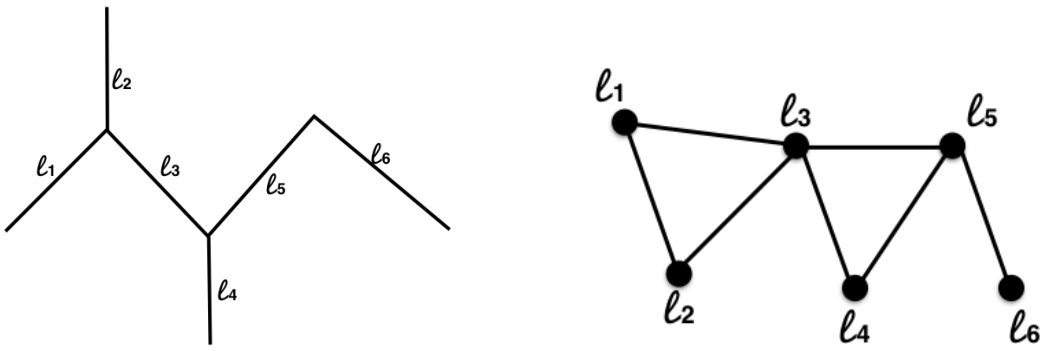
This shows that the minimum clearing time has to be at least 2.5 units, which means that the clique bound is not tight. This new bound is tight, and the solution is given by $f_i = 0.5, \forall i = 1 : 5$. Observe that constraint matrix is full-rank, and also all the constraints are binding.

Hence, the solution is unique. This hints at the fact that the minimum clearing algorithm may have to use global information.

Solving the minimum clearing time problem even in a simple ring like a pentagon has shown that the clique constraint is not necessarily tight. Therefore, before proceeding to solve the minimum clearing time problem in more complex graphs, it is important to study the problem in context of rings. Ring networks are also often used to demonstrate the sub-optimality of LQF or GMS algorithm [5, 7]. This further motivates us to understand the minimum clearing time problem in context of ring networks.

First thing to observe is that the number of maximal feasible sets still grows exponentially with the number of links. Therefore, we face the same problems discussed in the previous chapter to explicitly solve the linear program. So, our main aim here again is to find feasible algorithms that solve the minimum clearing time problem in rings, i.e., in polynomial time or in linear time if possible.

In this chapter, we will be using the *conflict graph* representation of the network. We will solve the minimum clearing time problem for the conflict graph.



(a) $G(V, E)$ (b) $G_c(V_c, E_c)$

Figure 3.2: Under 1-hop interference model, the graph on right is the conflict graph of the graph on the left.

Consider the conflict graph representation of the graph $G(V, E)$ denoted by $G_c(V_c, E_c)$. In the conflict graph representation $G_c(V_c, E_c)$, each node corresponds to an edge of the original graph $G(V, E)$. Hence, $V_c = E$. Two nodes in the conflict graph are joined by an edge, if the corresponding links interfere in the original graph $G(V, E)$. Notice that maximal feasible sets

of $G(V, E)$ corresponds to maximal independent sets of $G_c(V_c, E_c)$. An example is shown in Figure 3.2. Another example, the conflict graph of the pentagon network in Figure 3.1 is shown in Figure 3.3. Observe that when using the conflict graph representation, we no longer need a model for interference since it is implicit in the conflict graph: no two adjacent nodes can be cleared simultaneously.

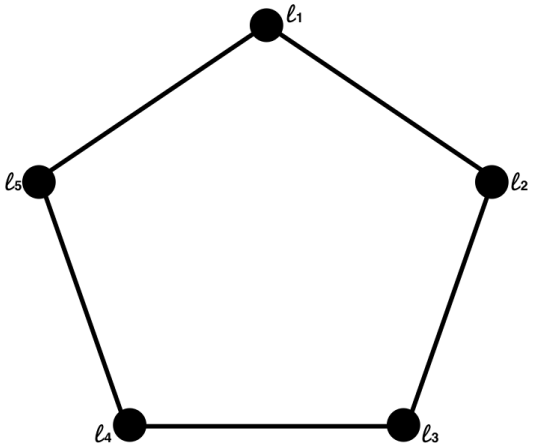


Figure 3.3: Conflict graph of the pentagon network

Definition 3.1.1. *An independent set is a set of nodes $I \subseteq V$, such that no two nodes in I are adjacent.*

Definition 3.1.2. *A maximal independent set is an independent set that is not a subset of any other independent set.*

Let $\tau(n)$ denote the time to clear a node $n \in V_c$. Let $\mathcal{S}$ denote the set of all the maximal independent sets S of the graph $G_c(V_c, E_c)$. For the conflict graph $G_c(V_c, E_c)$, we reformulate the minimum clearing time linear program as follows

$$\begin{aligned} &\min \sum_{S \in \mathcal{S}} f_S \\ &\text{s.t.} \\ &\sum_{S: n \in S} f_S \geq \tau(n), \forall n \in V_c \\ &f_S \geq 0, \forall S \in \mathcal{S} \end{aligned} \tag{3.1}$$

where f_S represents the time allocated to a maximal independent set S. And the dual program can be reformulated as

$$\begin{aligned} &\max \sum_{n \in V_c} \tau(n).x_n \\ &\text{s.t.} \\ &\sum_{n:n \in S} x_n \leq 1, \forall S \in \mathcal{S} \\ &x_n \geq 0, \forall n \in V_c \end{aligned} \tag{3.2}$$

Definition 3.1.3. *A clique C is either a singleton set $C \subseteq V_c$ or if $|C| > 1$, it is a subset of nodes $C \subseteq V_c$, such that $\forall n_1, n_2 \in C$, n_1 and n_2 are adjacent.*

Definition 3.1.4. *A maximal clique is a clique that is not a subset of any other clique.*

Note that clique of a conflict graph represents a set of mutually interfering links in the original graph. Observe that every clique set of $G(V, E)$ corresponds to a clique of the conflict graph $G_c(V_c, E_c)$.

3.2 Bipartite Graphs

Before proceeding to analyze ring networks, we study the special case of *bipartite graphs*, for which the clique bound can be shown to be tight. We later apply these results to ring networks.

Bipartite graphs were studied extensively in the context of graph colouring problems. Bipartite graphs have a chromatic number and a clique number of 2. Bipartite graphs do not contain an odd cycle. Conversely, any graph that does not contain an odd cycle is bipartite [15]. It so happens that the structure of bipartite graphs also provides an easy solution to minimum clearing time problem.

Definition 3.2.1. *Bipartite Graph*

A graph G(V,E) is bipartite iff $\exists V_1, V_2$ such that

1) $V = V_1 \bigcup V_2$, where $V_1 \bigcap V_2 = \phi$.

2) every edge $e \in E$ must join a node in V_1 to a node in V_2.

Consider a bipartite graph $G_c(V_c, E_c)$, where V_c is divided into two disjoint sets V_1 and V_2. Every edge $e \in E$ connects a node in V_1 to a node V_2. A node a is a neighbour of a node b iff there is an edge joining the nodes a and b. For every node $n \in V_c$, let $N(n)$ denote the set of neighbours of node n. Since the graph is bipartite, observe that if $n \in V_2$, then $N(n) \subseteq V_1$, and if $n \in V_1$, then $N(n) \subseteq V_2$. Let $\tau(n)$ denote the clearing time of a node n. Let $\mathcal{T}(n)$ denote the time interval allocated to a node n. We describe the greedy algorithm that uses only local information as follows:

$$\mathcal{T}(n) = (0, \tau(n)], \forall n \in V_1$$
$$\mathcal{T}(n) = (\max_{n_1 \in N(n)} \tau(n_1), \max_{n_1 \in N(n)} \tau(n_1) + \tau(n)], \forall n \in V_2$$

Observe that this allocation is feasible. Since $|\mathcal{T}(n)| = \tau(n), \forall n \in V_c$, each node gets enough time for clearing. Since $\mathcal{T}(n) \bigcap \mathcal{T}(n_1) = \phi, \forall n_1 \in N(n), \forall n \in V_c$, interference constraints are satisfied for all nodes $n \in N$.

Claim 4. *If the conflict graph $G_c(V_c, E_c)$ is bipartite, then the clique bound is tight.*

Proof. Firstly, observe that maximal cliques of the graph $G_c(V_c, E_c)$ are given by the node pairs $C = \{n_1, n_2\}$, where $n_1 \in V_c$ and $n_2 \in N(n_1)$. Let $\mathscr{C}$ denote the set of all the maximal cliques of $G_c(V_c, E_c)$. Define $C^M := \arg\max_{C \in \mathscr{C}} \tau(C)$, where $\tau(C) = \sum_{n \in C} \tau(n)$.

Under the proposed algorithm, for every node $n \in V_1$, $\mathcal{T}(n) = (0, \tau(n)]$. Since any single node $n \in V_1$ is a clique, $\mathcal{T}(n) \subseteq (0, \tau(C^M)]$.

For every node $n \in V_2$, $\mathcal{T}(n) = (\max_{n_1 \in N(n)} \tau(n_1), \max_{n_1 \in N(n)} \tau(n_1) + \tau(n)]$. As discussed earlier, the node pair $C = \{n_1, n_2\}, n_1 \in V_c$ and $n_2 \in N(n_1)$ is a maximal clique of $G_c(V_c, E_c)$, therefore $\mathcal{T}(n) \subseteq (0, \tau(C^M)]$. Hence, the clique bound is tight. □

3.3 Ring with $2N$ nodes

Note that a ring R_{2N} with $2N$ nodes is the conflict graph representation of the following wireless network: $2N$ links placed in a ring and under the 1-hop interference model. Choose a node in R_{2N}, and label the nodes as $n_1, n_2, \ldots n_{2N}$ going clockwise around the ring.

Claim 5. *R_{2N} is bipartite.*

Proof. Say, $V = \{n_1, n_2, ...n_{2N}\}$ represents the set of nodes. Choose $V_1 = \{n_1, n_3, ...n_{2N-1}\}$ and $V_2 = \{n_2, n_4, ...n_{2N}\}$. $V = V_1 \bigcup V_2$ and $V_1 \bigcap V_2 = \phi$. Observe that every edge of R_{2N} connects a node in V_1 to a node in V_2. Hence, R_{2N} is bipartite. □

So for R_{2N}, clique bound is tight and a greedy solution using only local information is optimal.

3.4 Ring with $2N + 1$ nodes

Note that a ring R_{2N+1} with $2N + 1$ nodes is the conflict graph representation of the following wireless network: $2N + 1$ links placed in a ring and under the 1-hop interference model. Choose a node in R_{2N+1}, and label the nodes as $n_1, n_2, ...n_{2N+1}$ going clockwise around the ring. Let the time to clear a node n_i be denoted by $\tau(n_i)$. Observe that any two adjacent nodes in R_{2N+1} forms a maximal clique. Let $C_i = \{n_i, n_{i+1}\}$ denote a maximal clique, $\forall i = 1 : 2N$, and $C_{2N+1} = \{n_{2N+1}, n_1\}$. Define $\tau(C_i) = \sum_{n \in C_i} \tau(n)$. Also define $\tau_A = \sum_{i=1}^{2N+1} \tau(C_i)/2N$.

Let $\mathcal{S}$ denote the set of all the maximal independent sets S of the graph R_{2N+1}. We can reformulate the minimum clearing time as follows :

$$\begin{aligned} &\min \sum_{S \in \mathcal{S}} f_S \\ &\text{s.t.} \\ &\sum_{S: n_i \in S} f_S \geq \tau(n_i), \forall i = 1 : 2N + 1 \\ &f_S \geq 0, \forall S \in \mathcal{S} \end{aligned} \tag{3.3}$$

where f_S represents the time allocated to a maximal independent set S.

Claim 6. *τ_A is a lower bound of the minimum clearing time.*

Proof. Consider the dual of the linear program (3.3).

$$\max \sum_{i=1}^{2N+1} \tau(n_i).x_i$$

s.t.

$$\sum_{i:n_i \in S} x_i \leq 1, \forall S \in \mathcal{S}$$

$$x_i \geq 0, \forall i = 1 : 2N + 1 \tag{3.4}$$

Consider the maximal independent set $\{n_1, n_3, n_5, ...n_{2N-1}\}$, constructed in this manner starting at n_1 and going around the ring including every other node. The node n_{2N+1} cannot be included because it is adjacent to the node n_1. Due to the circular symmetry of the ring, similar sets can be constructed starting at any node, by rotating the above choice so that the start node is at the desired node. Observe that the cardinality of any of the considered maximal independent sets is N. It can also be noted that N is the maximum possible cardinality for a maximal independent set $S \in \mathcal{S}$. We propose the following solution to the dual program (3.4)

$$x_i = 1/N, \forall i = 1 : 2N + 1 \tag{3.5}$$

Clearly, $x_i > 0, \forall i = 1 : 2N + 1$. Also observe that $\sum_{i:n_i \in S} x_i = |S|/N \leq 1$, where $|S|$ represents the cardinality of the set S. Hence, the proposed solution (3.5) is feasible. The value of the dual objective function under (3.5) is given by

$$\begin{aligned} &\sum_{i=1}^{2N+1} \tau(n_i)/N \\ =&\sum_{i=1}^{2N+1} \tau(C_i)/2N \\ =&\tau_A \end{aligned}$$

Using weak-duality theorem, τ_A is a lower bound to the primal problem (3.3). □

We will solve the minimum clearing time problem for the two cases: 1) $\tau_A \geq \tau(C^M)$ and 2) $\tau(C^M) \geq \tau_A$. We will show that the minimum clearing time equals $\max\{\tau_A, \tau(C^M)\}$. More

importantly, we will discuss the algorithms that solve the minimum clearing time problem in each case.

3.4.1 $\tau_A \geq \tau(C^M)$

Claim 7. *If $\tau_A \geq \tau(C^M)$, τ_A is the minimum clearing time.*

Proof. Define the function

$$g(j,k) = \begin{cases} j + 2k \pmod{2N+1} & \text{if } j + 2k \pmod{2N+1} > 0 \\ 2N+1 & \text{if } j + 2k \pmod{2N+1} = 0 \end{cases}$$

Consider the maximal independent sets $S_j = \{n_{g(j,k)}\}_{k=1:N}$ for $j = 1 : 2N + 1$. Observe that $|S_j| = N$. e.g. $S_1 = \{n_3, n_5, n_7, ...n_{2N-1}, n_{2N+1}\}$. Let $\mathcal{S}_N = \{S_j\}_{j=1:2N+1}$ denote the set of all such maximal independent sets. We only timeshare among the sets in $\mathcal{S}_N$. We present the following solution to the primal LP (3.3)

$$f_j = \tau_A - \tau(C_j), \forall j = 1 : 2N + 1.$$

where f_j is the time allocated to the maximal independent set S_j.

Since $\tau_A \geq \max_{i=1:2N+1} \tau(C_i)$, $f_j \geq 0, \forall j = 1 : 2N + 1$.

Observe that the node n_1 is only present in the sets $S_2, S_4, S_6 ... S_{2N}$. Due to the circular symmetry of the problem, the node n_i is only present in the sets $\{S_{g(i-1,k)}\}_{k=1:N}$. Therefore,

$$\begin{aligned} &\sum_{j:n_i \in S_j} f_j \\ =&\sum_{k=1}^{N} f_{g(i-1,k)} \\ =&N.\tau_A - \sum_{k=1}^{N} \tau(C_{g(i-1,k)}) \\ =&\sum_{j=1}^{2N+1} \tau(n_j) - \sum_{k=1}^{N} \tau(n_{g(i-1,k)}) + \tau(n_{g(i-1,k)+1}) \\ =&\tau(n_i) \end{aligned}$$

Hence the solution is feasible. Each node n_i has exactly enough time to clear its load. The total time required under the solution is given by

$$\begin{aligned}
&\sum_{j=1}^{2N+1} f_j \\
=&\sum_{j=1}^{2N+1} \tau_A - \tau(C_j) \\
=&(2N+1)\tau_A - \sum_{j=1}^{2N+1} \tau(C_j) \\
=&(2N+1)\tau_A - (2N)\tau_A \\
=&\tau_A
\end{aligned}$$

Hence, the network can be cleared in τ_A. □

So, it can be seen that if $\tau_A \geq \tau(C^M)$, the minimum clearing time problem can be solved by time sharing among $2N+1$ maximal independent sets. We have also provided the value of time that needs to be spent in each of these states. The solution in this case has a linear complexity, as it only involves calculating the values of $\tau_A - \tau(C_j)$'s. As can be seen from the solution, the value of the minimum clearing time depends on global information.

3.4.2 $\tau(C^M) \geq \tau_A$

In this case, we will show that a greedy algorithm can solve the minimum clearing time problem. The greedy algorithm can clear the ring network in clique time. Without loss of generality, we assume $\tau(C_1) = \tau(C^M)$. To initialize the algorithm, we allocate time to the nodes n_1, n_2 as $\mathcal{T}(n_1) = (0, \tau(n_1)]$ and $\mathcal{T}(n_2) = (\tau(n_1), \tau(n_1) + \tau(n_2)]$. Observe that the time to n_3 can be allocated as $\mathcal{T}(n_3) = (0, \tau(n_3)] \subseteq \mathcal{T}(n_1)$, because $\tau(C_1) \geq \tau(C_2)$. Before describing the algorithm, we introduce some necessary notation. For each node n_i, the allocated time $\mathcal{T}(n_i)$ can be written as $\mathcal{T}(n_i) = \mathcal{T}_1(n_i) \bigcup \mathcal{T}_2(n_i)$, such that $\mathcal{T}_1(n_i) \bigcap \mathcal{T}_2(n_i) = \phi$ and $\mathcal{T}_1(n_i) \subseteq \mathcal{T}(n_1)$, $\mathcal{T}_2(n_i) \subseteq \mathcal{T}(n_2)$. $T_1(n_i) = |\mathcal{T}_1(n_i)|$ is the length of $\mathcal{T}_1(n_i)$. Similarly, $T_2(n_i) = |\mathcal{T}_2(n_i)|$. Under this notation, observe $T_1(n_3) = \tau(n_3)$, and $T_2(n_3) = 0$. For nodes $\{n_i\}_{i=4:2N}$, we describe the greedy time allocation as follows

For $i = 4 : 2N$

If i is even :

$T_1(n_i) = \min\{\tau(n_1) - T_1(n_{i-1}), \tau(n_i)\}$

$T_2(n_i) = \tau(n_i) - T_1(n_i) = \max\{\tau(n_i) + T_1(n_{i-1}) - \tau(n_1), 0\}$

If i is odd :

$T_2(n_i) = \min\{\tau(n_2) - T_2(n_{i-1}), \tau(n_i)\}$

$T_1(n_i) = \tau(n_i) - T_2(n_i) = \max\{\tau(n_i) + T_2(n_{i-1}) - \tau(n_2), 0\}$

For every $\{n_i\}_{i=4:2N}$, greedily allocate such that $\mathcal{T}_1(n_i) \subseteq \mathcal{T}(n_1) - \mathcal{T}_1(n_{i-1})$, with a length of $T_1(n_i)$ described in the algorithm. Similarly, greedily allocate time such that $\mathcal{T}_2(n_i) \subseteq \mathcal{T}(n_2) - \mathcal{T}_2(n_{i-1})$ with a length of $T_2(n_i)$ described in the algorithm. And for $i = 2N + 1$, we allocate time such that $\mathcal{T}(n_{2N+1}) \subseteq \mathcal{T}(n_2) - \mathcal{T}_2(n_{2N})$. This is because the node n_{2N+1} is adjacent to n_1, and therefore cannot use any time from the interval $\mathcal{T}(n_1)$. Now we prove some results that will help in establishing the feasibility of the greedy algorithm.

Lemma 3.4.1. $0 \leq T_1(n_i) \leq \tau(n_1), 0 \leq T_2(n_i) \leq \tau(n_2)$, *for* $i = 3 : 2N$.

Proof. We use proof by induction. Since $T_1(n_3) = \tau(n_3)$ and $T_2(n_3) = 0$. Therefore, for $i = 3$, we have $0 \leq T_1(n_i) \leq \tau(n_1), 0 \leq T_2(n_i) \leq \tau(n_2)$.

Assume $0 \leq T_1(n_i) \leq \tau(n_1), 0 \leq T_2(n_i) \leq \tau(n_2)$ for $i = k$, where $4 \leq k \leq 2N - 1$. Now, we will show that $0 \leq T_1(n_i) \leq \tau(n_1), 0 \leq T_2(n_i) \leq \tau(n_2)$ for $i = k + 1$ in two cases: 1)k is odd and 2) k is even.

If k is odd:

$T_1(n_{k+1}) = \min\{\tau(n_1) - T_1(n_k), \tau(n_{k+1})\}$. Since both $\tau(n_1) - T_1(n_k), \tau(n_{k+1}) \geq 0$, we can write $T_1(n_{k+1}) \geq 0$.

$$\begin{aligned} T_1(n_{k+1}) &= \min\{\tau(n_1) - T_1(n_k), \tau(n_{k+1})\} \\ &\leq \tau(n_1) - T_1(n_k) \\ &\leq \tau(n_1). \end{aligned}$$

It follows from the definition that $T_2(n_{k+1}) = \max\{\tau(n_{k+1}) + T_1(n_k) - \tau(n_1), 0\}$, thus $T_2(n_{k+1}) \geq 0$.

$$\begin{aligned} T_2(n_{k+1}) &= \max\{\tau(n_{k+1}) + T_1(n_k) - \tau(n_1), 0\} \\ T_2(n_{k+1}) + T_2(n_k) &= \max\{\tau(n_{k+1}) + T_1(n_k) + T_2(n_k) - \tau(n_1), T_2(n_k)\} \\ T_2(n_{k+1}) + T_2(n_k) &= \max\{\tau(n_{k+1}) + \tau(n_k) - \tau(n_1), T_2(n_k)\} \\ T_2(n_{k+1}) + T_2(n_k) &= \max\{\tau(C_k) - \tau(C_1) + \tau(n_2), T_2(n_k)\} \end{aligned}$$

Since $\tau(C_1) = \tau(C^M) \geq \tau(C_k)$, $\tau(C_k) - \tau(C_1) \leq 0$. Therefore, $\max\{\tau(C_k) - \tau(C_1) + \tau(n_2), T_2(n_k)\} \leq \tau(n_2)$, which implies that $T_2(n_{k+1}) \leq \tau(n_2)$.

If k is even:

$T_2(n_{k+1}) = \min\{\tau(n_2) - T_2(n_k), \tau(n_{k+1})\}$. Since both $\tau(n_2) - T_2(n_k), \tau(n_{k+1}) \geq 0$, we can write $T_2(n_{k+1}) \geq 0$.

$$\begin{aligned} T_2(n_{k+1}) &= \min\{\tau(n_2) - T_2(n_k), \tau(n_{k+1})\} \\ &\leq \tau(n_2) - T_2(n_k) \\ &\leq \tau(n_2) \end{aligned}$$

It follows from the definition that $T_1(n_{k+1}) = \max\{\tau(n_{k+1}) + T_2(n_k) - \tau(n_2), 0\}$, thus

$T_1(n_{k+1}) \geq 0$.

$$
\begin{aligned}
T_1(n_{k+1}) &= \max\{\tau(n_{k+1}) + T_2(n_k) - \tau(n_2), 0\} \\
T_1(n_{k+1}) + T_1(n_k) &= \max\{\tau(n_{k+1}) + T_1(n_k) + T_2(n_k) - \tau(n_2), T_1(n_k)\} \\
T_1(n_{k+1}) + T_1(n_k) &= \max\{\tau(n_{k+1}) + \tau(n_k) - \tau(n_2), T_1(n_k)\} \\
T_1(n_{k+1}) + T_1(n_k) &= \max\{\tau(C_k) - \tau(C_1) + \tau(n_1), T_1(n_k)\}
\end{aligned}
$$

Since $\tau(C_1) = \tau(C^M) \geq \tau(C_k)$, $\tau(C_k) - \tau(C_1) \leq 0$. Therefore $\max\{\tau(C_k) - \tau(C_1) + \tau(n_1), T_1(n_k)\} \leq \tau(n_1)$, which implies that $T_1(n_{k+1}) \leq \tau(n_1)$. □

Lemma 3.4.2. *$T_1(n_i) + T_1(n_{i-1}) \leq \tau(n_1)$ and $T_2(n_i) + T_2(n_{i-1}) \leq \tau(n_2)$ for $i = 4 : 2N$*

Proof. **If *i* is odd**

$$
\begin{aligned}
T_1(n_i) + T_1(n_{i-1}) &= T_1(n_{i-1}) + \max\{\tau(n_i) + T_2(n_{i-1}) - \tau(n_2), 0\} \\
&= \max\{\tau(n_i) + \tau(n_{i-1}) - \tau(n_2), T_1(n_{i-1})\} \\
&= \max\{\tau(C_{i-1}) - \tau(C_1) + \tau(n_1), T_1(n_{i-1})\}
\end{aligned}
$$

From Lemma 3.4.1 $T_1(n_{i-1}) \leq \tau(n_1)$, and since $\tau(C_1) = \tau(C^M)$, $\tau(C_{i-1}) - \tau(C_1) \leq 0$. Therefore, $T_1(n_i) + T_1(n_{i-1}) \leq \tau(n_1)$.

$$
\begin{aligned}
T_2(n_i) + T_2(n_{i-1}) &= T_2(n_{i-1}) + \min\{\tau(n_2) - T_2(n_{i-1}), \tau(n_i)\} \\
&= \min\{\tau(n_2), \tau(n_i) + T_2(n_{i-1})\} \\
&\leq \tau(n_2)
\end{aligned}
$$

If i is even

$$\begin{aligned} T_1(n_i) + T_1(n_{i-1}) &= T_1(n_{i-1}) + \min\{\tau(n_1) - T_1(n_{i-1}), \tau(n_i)\} \\ &= \min\{\tau(n_1), \tau(n_i) + T_1(n_{i-1})\} \\ &\leq \tau(n_1) \end{aligned}$$

$$\begin{aligned} T_2(n_i) + T_2(n_{i-1}) &= T_2(n_{i-1}) + \max\{\tau(n_i) + T_1(n_{i-1}) - \tau(n_1), 0\} \\ &= \max\{\tau(n_i) + \tau(n_{i-1}) - \tau(n_1), T_2(n_{i-1})\} \\ &= \max\{\tau(C_{i-1}) - \tau(C_1) + \tau(n_2), T_2(n_{i-1})\} \end{aligned}$$

From Lemma 3.4.1 $T_2(n_{i-1}) \leq \tau(n_2)$, and since $\tau(C_1) = \tau(C^M)$, $\tau(C_{i-1}) - \tau(C_1) \leq 0$. Therefore, $T_2(n_i) + T_2(n_{i-1}) \leq \tau(n_2)$.

□

Claim 8. *If $\tau(C_1) = \tau(C^M) \geq \tau_A$, then $\tau(n_{2N+1}) \leq \tau(n_2) - T_2(n_{2N})$*

Proof. Consider the following sequence $\{T_2(n_{2N}), T_1(n_{2N-1}), T_2(n_{2N-2}), \ldots$
$..T_2(n_{2k}), T_1(n_{2k-1}), ...T_2(n_4), T_1(n_3)\}$. Due to the iterative definition of the greedy scheme, the value of $\tau(n_{2N+1}) + T_2(n_{2N}) - \tau(n_2)$ can be calculated from this sequence.

If every element in the sequence > 0:

$T_2(n_{2k}) = \max\{\tau(n_{2k}) + T_1(n_{2k-1}) - \tau(n_1), 0\} = \tau(n_{2k}) + T_1(n_{2k-1}) - \tau(n_1)$ and $T_1(n_{2k-1}) = \max\{\tau(n_{2k-1}) + T_2(n_{2k-2}) - \tau(n_2), 0\} = \tau(n_{2k-1}) + T_2(n_{2k-2}) - \tau(n_2)$, for $k = 2 : N$.

Summing the expressions $T_2(n_{2k}) = \tau(n_{2k}) + T_1(n_{2k-1}) - \tau(n_1)$ and $T_1(n_{2k-1}) = \tau(n_{2k-1}) +$

$T_2(n_{2k-2}) - \tau(n_2)$ from $k = 2 : N$, we have

$$\begin{aligned} T_2(n_{2N}) &= \left(\sum_{i=3}^{2N} \tau(n_i)\right) - (N-1)\tau(n_1) - (N-1)\tau(n_2) + T_2(n_2) \\ &= \left(\sum_{i=3}^{2N} \tau(n_i)\right) - (N-1)\tau(n_1) - (N-1)\tau(n_2) + \tau(n_2) \\ &= \left(\sum_{i=1}^{2N} \tau(n_i)\right) - N\tau(n_1) - (N-1)\tau(n_2) \end{aligned}$$

Now, the value of $\tau(n_{2N+1}) + T_2(n_{2N}) - \tau(n_2)$ can be calculated as

$$\begin{aligned} &\tau(n_{2N+1}) + T_2(n_{2N}) - \tau(n_2) \\ &= \left(\sum_{i=1}^{2N+1} \tau(n_i)\right) - N\tau(n_1) - N\tau(n_2) \\ &= N\tau_A - N\tau(C_1) \\ &= N\{\tau_A - \tau(C_1)\} \\ &\leq 0 \text{ (since } \tau(C^M) \geq \tau_A) \end{aligned}$$

Hence, $\tau(n_{2N+1}) \leq \tau(n_2) - T_2(n_{2N})$.

If one or more elements in the sequence are $= 0$**:**

Case 1:

In the sequence $\{T_2(n_{2N}), T_1(n_{2N-1}), T_2(n_{2N-2}), ...T_2(n_4), T_1(n_3)\}$, going from the left to the right, $T_2(n_{2i})$ is the first zero that we encounter. So $T_2(n_{2i}) = 0$.

We can write $T_1(n_{2k-1}) = \tau(n_{2k-1}) + T_2(n_{2k-2}) - \tau(n_2)$ and $T_2(n_{2k}) = \tau(n_{2k}) + T_1(n_{2k-1}) - \tau(n_1)$ for $k = i + 1 : N$. Summing these expressions from $k = i + 1 : N$, we can write $T_2(n_{2N}) = \left(\sum_{j=2i+1}^{2N} \tau(n_j)\right) - (N - i)\{\tau(n_1) + \tau(n_2)\}$.

$$
\begin{aligned}
&\tau(n_{2N+1}) + T_2(n_{2N}) - \tau(n_2) \\
&= \left(\sum_{j=2i+1}^{2N+1} \tau(n_j)\right) - (N-i)\tau(n_1) - (N-i+1)\tau(n_2) \\
&= \left(\sum_{j=i}^{N-1} \{\tau(n_{2j+1}) + \tau(n_{2j+2})\}\right) + \{\tau(n_{2N+1}) + \tau(n_1)\} - (N-i+1)\{\tau(n_1) + \tau(n_2)\} \\
&= \left(\sum_{j=i}^{N} \tau(C_{2j+1})\right) - (N-i+1)\tau(C_1) \\
&\leq 0 \text{ (since } \tau(C_1) = \tau(C^M))
\end{aligned}
$$

Hence, $\tau(n_{2N+1}) \leq \tau(n_2) - T_2(n_{2N})$.

Case 2:

In the sequence $\{T_2(n_{2N}), T_1(n_{2N-1}), T_2(n_{2N-2}), \ldots T_2(n_4), T_1(n_3)\}$, going from the left to the right, $T_2(n_{2i-1})$ is the first zero that we encounter. So $T_2(n_{2i-1}) = 0$.

We can write $T_2(n_{2k}) = \tau(n_{2k}) + T_1(n_{2k-1}) - \tau(n_1)$ and $T_1(n_{2k+1}) = \tau(n_{2k+1}) + T_2(n_{2k}) - \tau(n_2)$ for $k = i : N-1$. Summing these expressions from $k = i : N-1$ along with $T_2(n_{2N}) = \tau(n_{2N}) + T_1(n_{2N-1}) - \tau(n_1)$, we can write $T_2(n_{2N}) = \left(\sum_{j=2i}^{2N} \tau(n_j)\right) - (N-i+1)\tau(n_1) - (N-i)\tau(n_2)$.

$$
\begin{aligned}
&\tau(n_{2N+1}) + T_2(n_{2N}) - \tau(n_2) \\
&= \left(\sum_{j=2i}^{2N+1} \tau(n_j)\right) - (N-i+1)\{\tau(n_1) + \tau(n_2)\} \\
&= \left(\sum_{j=i}^{N} \tau(n_{2j}) + \tau(n_{2j+1})\right) - (N-i+1)\{\tau(n_1) + \tau(n_2)\} \\
&= \left(\sum_{j=i}^{N} \tau(C_{2j})\right) - (N-i+1)\tau(C_1) \\
&\leq 0 \text{ (since } \tau(C_1) = \tau(C^M))
\end{aligned}
$$

Hence, $\tau(n_{2N+1}) \leq \tau(n_2) - T_2(n_{2N})$. □

Using Lemma 3.4.1 and Lemma 3.4.2, for nodes $\{n_i\}_{i=3:2N}$, we can establish the feasibility of time allocation under the greedy algorithm. From Lemma 3.4.1, we know that since $T_1(n_i), T_2(n_i) \geq 0$, we have established that all the terms are non-negative. From Lemma 3.4.2, we know that $T_1(n_i) + T_1(n_{i-1}) \leq \tau(n_1)$ and $T_2(n_i) + T_2(n_{i-1}) \leq \tau(n_2)$, implying that it is possible to allocate time such that $\mathcal{T}_1(n_i) \subseteq \mathcal{T}(n_1) - \mathcal{T}_1(n_{i-1})$, and $\mathcal{T}_2(n_i) \subseteq \mathcal{T}(n_2) - \mathcal{T}_2(n_{i-1})$. Since, this also means $\mathcal{T}(n_i) \bigcap \mathcal{T}(n_i - 1) = \phi$, interference constraints are not violated. From the definition, $T_1(n_i) + T_2(n_i) = \tau(n_i)$ which implies that the demand of each node is satisfied under the allocation.

Now the only remaining step is to allocate time for the node n_{2N+1}. As discussed earlier, time has to be allocated such that $\mathcal{T}(n_{2N+1}) \subseteq \mathcal{T}(n_2) - \mathcal{T}_2(n_{2N})$. Therefore, for the greedy algorithm to work the following condition must be satisfied: $\tau(n_{2N+1}) \leq \tau(n_2) - T_2(n_{2N})$. And using Claim 8, we have established that the condition is satisfied whenever $\tau(C^M) \geq \tau(C_a)$.

Chapter 4

Future Work

4.1 More General Networks

In the future, one can attempt to solve the minimum clearing time problem in more general networks. With the help of additional structure, the minimum clearing time problem may be polynomial time solvable. We present one such example: Consider a setup where the plane is divided into square cells. e.g., see Figure 4.1. A transmission in any cell interferes with its immediate neighbours. e.g., In Figure 4.1, a transmission in cell C_{24} interferes with transmissions in cells $C_{14}, C_{23}, C_{34}, C_{25}$. Observe that there are 4 immediate neighbours for any cell.

C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
C_{21}	C_{22}	C_{23}	C_{24}	C_{25}
C_{31}	C_{32}	C_{33}	C_{34}	C_{35}
C_{41}	C_{42}	C_{43}	C_{44}	C_{45}
C_{51}	C_{52}	C_{53}	C_{54}	C_{55}

Figure 4.1: Square Cells

For this setup, the conflict graph is given by the square lattice, shown in Figure 4.2. As can be seen from Figure 4.2, the conflict graph is bipartite. Therefore for this example, using the results from Chapter 3, the clique bound is tight, and the minimum clearing time problem can be solved using a greedy algorithm based on local information.

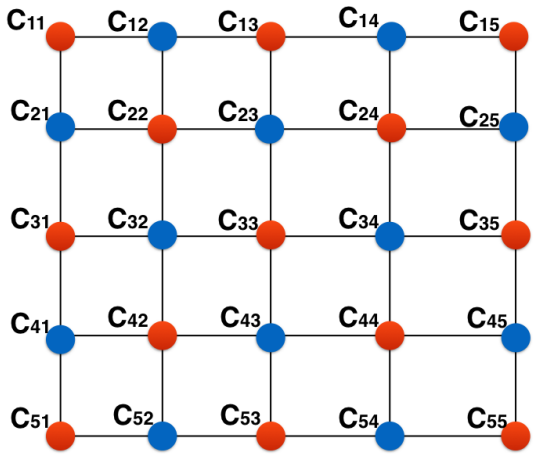


Figure 4.2: Square Lattice

This approach to solving the minimum clearing time problem in general networks is not scalable. We will eventually have to concede, due to the NP-hardness of the general minimum clearing time problem [9]. However, there might be a few important general networks where the problem is polynomial time solvable.

4.2 Scheduling Algorithm

The solution to the minimum clearing time problem helps us understand how the network is constrained under various load distributions. It also provides the quickest way to clear a given load. In the future, we want to use these insights to design a scheduling policy.

We expect such a policy to perform optimally in tree like network topologies, and perform close to optimal in simple ring topologies. We would also like to study its performance in a general network, and benchmark it against other similar policies such as LQF scheduling.

4.3 HetNet Applications

In Chapter 2, we have discussed a few applications of the minimum clearing time problem in a HetNet. We want to solidify these results by considering a setup with random traffic arrivals, instead of deterministic loads. This will also be a part of the scheduling policy design.

In Chapter 2, we have demonstrated the benefit of coordinating scheduling decision between neighbouring cells in a deterministic setup. Under random arrivals, we want to design a policy to coordinate the ABS scheme between neighbouring cells.

Using our current model, we want to derive results on how the wireless backhaul might be coordinated in an optimal manner. We are also interested in the capacity of a wireless backhaul system relative to its wired counterpart. Understanding this will have implications on the deployment of pico-cells, and HetNet design.

We are also interested in other parameters such as cell-association, and load balancing in the context of sharing cell-edge traffic.

References

[1] S. Simeons, P. Pellati, J. Gosteau, K. Gosse, and C. Ware, "The evolution of 5GHz WLAN toward higher throughputs," in *IEEE Wireless Communications*, vol. 10, no. 6, 2003, pp. 6–13.

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

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

[4] X. Wu, R. Srikant, and J. R. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 595–605, 2007.

[5] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, 2009.

[6] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Advances in Applied probability*, pp. 505–521, 2006.

[7] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on throughput efficiency of greedy maximal scheduling in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 709–720, 2011.

[8] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. ACM, 2006, pp. 227–238.

[9] E. Arikan, "Some complexity results about packet radio networks (corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 4, pp. 681–685, 1984.

[10] S. V. Hanly and P. A. Whiting, "On the capacity of HetNets," in *Information Theory and Applications Workshop (ITA)*, 2014, pp. 1–9.

[11] S. V. Hanly, C. Liu, and P. Whiting, "Capacity and stable scheduling in heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 6, pp. 1266–1279, 2015.

[12] Q. He, V. Angelakis, A. Ephremides, and D. Yuan, "Polynomial complexity minimum-time scheduling in a class of wireless networks," *arXiv preprint arXiv:1403.4144*, 2014.

[13] C. Liu, M. Li, S. Hanly, and P. Whiting, "Joint downlink user association and interference management in two-tier HetNets with dynamic resource partitioning," *IEEE Transactions on Vehicular Technology*, vol. PP, pp. 1–1, 2016.

[14] A. D. Wyner, "Shannon-theoretic approach to a gaussian cellular multiple-access channel," *IEEE Transactions on Information Theory*, vol. 40, no. 6, pp. 1713–1727, 1994.

[15] A. S. Asratian, T. M. Denley, and R. Häggkvist, *Bipartite graphs and their applications*. Cambridge University Press, 1998, vol. 131.