FRACTIONAL EDGE COLORING FOR WIRELESS LINK SCHEDULING IN THE PHYSICAL INTERFERENCE MODEL

Guilherme Iecker Ricardo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: José Ferreira de Rezende
Valmir Carneiro Barbosa

Rio de Janeiro
Março de 2018

FRACTIONAL EDGE COLORING FOR WIRELESS LINK SCHEDULING IN THE PHYSICAL INTERFERENCE MODEL

Guilherme Iecker Ricardo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. José Ferreira de Rezende, Ph.D.

Prof. Abílio Pereira de Lucena Filho, Ph.D.

Prof. Diego Gimenez Passos, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2018

Ricardo, Guilherme Iecker

Fractional Edge Coloring for Wireless Link Scheduling in the Physical Interference Model/Guilherme Iecker Ricardo. – Rio de Janeiro: UFRJ/COPPE, 2018.

IX, 52 p.: il.; 29, 7cm.

Orientadores: José Ferreira de Rezende

Valmir Carneiro Barbosa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 51 – 52.

1. Wireless Networks. 2. Link Scheduling. 3. Operational Research. I. Rezende, José Ferreira de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

To my parents,
Dulceléa and Hélio

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

COLORAÇÃO FRACIONÁRIA DE ARESTAS PARA O ESCALONAMENTO DE ENLACES SEM FIO NO MODELO FÍSICO DE INTERFERÊNCIA

Guilherme Iecker Ricardo

Março/2018

Orientadores: José Ferreira de Rezende
Valmir Carneiro Barbosa

Programa: Engenharia de Sistemas e Computação

A popularidade de aplicações para Redes de Sensores Sem Fio (WSN) e Internet of Things (IoT) tem aumentado bastante nos últimos e, com ela, a demanda por redes mesh sem fio (WMN) mais eficientes e econômicas em termos de utilização de recursos. O problema de Escalonamento de Enlaces tem por objetivo melhorar a capacidade das redes por meio da adoção de uma estratégia inteligente de ativação dos enlaces sem fio. Essa estratégia garante a correta comunicação entre dispositivos, respeitando as restrições do Modelo de Interferência Física adotado. O presente trabalho oferece uma abordagem para encontrar o escalonamento ótimo por meio da redução do problema de Escalonamento de Enlaces ao problema de Coloração Fracionária de Arestas. Uma formulação de Programação Linear com complexidade exponencial no tamanho do grafo e um algoritmo para auxiliar uma construção mais eficiente dos modelos são apresentados. Finalmente, uma grande quantidade de experimentos foram realizados objetivando verificar a aplicabilidade e o desempenho da técnica na prática.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

FRACTIONAL EDGE COLORING FOR WIRELESS LINK SCHEDULING IN THE PHYSICAL INTERFERENCE MODEL

Guilherme Iecker Ricardo

March/2018

Advisors: José Ferreira de Rezende
Valmir Carneiro Barbosa

Department: Systems Engineering and Computer Science

The popularity of Wireless Sensor Networks (WSN) and Internet of Things (IoT) applications is experiencing an unprecedent increase in the last few years. Along with it, the demand for more efficient and economic Wireless Mesh Networks (WMNs) in terms of resource management. The Link Scheduling problem aims to improve network capacity and the resource usage of WMNs through adopting a smart strategy for wireless links activation. This strategy guarantees the strict communication between network devices, satisfying the adopted Physical Interference Model (PIM) constraints. The current work offers an approach to find the optimal scheduling by modeling the Link Scheduling problem as a Fractional Edge-Coloring problem. A Linear Programming (LP) formulation with exponential complexity on the graph size and an algorithm to aid efficiently building such models are introduced. Finally, a considerable amount of experiments were run in order to assess the technique's practical applicability and performance.

Contents

List of Figures

Chapter 1

Introduction

The Link Scheduling is a classic problem in Computer Networks, studied since the early 1980s. Enforcing a good link scheduling is important to optimize the amount of information exchanged in a certain network. More efficient wireless communication capabilities can lead to significant resource savings, such as for power, bandwidth, and so on. This is especially interesting to resource limited network architectures and technologies, such as Wireless Sensor Networks (WSNs) and Internet of Things (IoT) applications. Recently, with the increasing popularity of WSNs and IoT paradigms, the problem of finding better schedulings has been revisited and brought back to discussion in both scientific and industrial community. Therefore, investigating this problem has great relevance to the Wireless and Mobile Network field's status quo.

In order to help understanding the Link Scheduling problem, we first introduce a wireless mesh network (WMN) infrastructure that often offers the basis for WSN, IoT, and many other ad-hoc communication technology systems. A WMN is a sort of Ad-Hoc wireless network so its architecture is decentralized and radio nodes are arranged in a mesh topology. This means that nodes are able to communicate directly to each other and have a certain autonomy to decide what is the proper way to forward data. Figure 1.1 shows an example where a WMN architecture defines the network's backbone.

Usually, in WMNs, all nodes share the same transmission medium and multiple transmissions are allowed simultaneously. So in order to better use network resources, transmissions need to be organized in an efficient way. In this work, we use the Space Time Division Multiple Access (STDMA) strategy so we divide the transmission period into timeslots and try to arrange as many non-interfering transmission links as possible in a same timeslot. Therefore, the Link Scheduling problem can be roughly defined as determining the least number of timeslots such that all links are scheduled once.

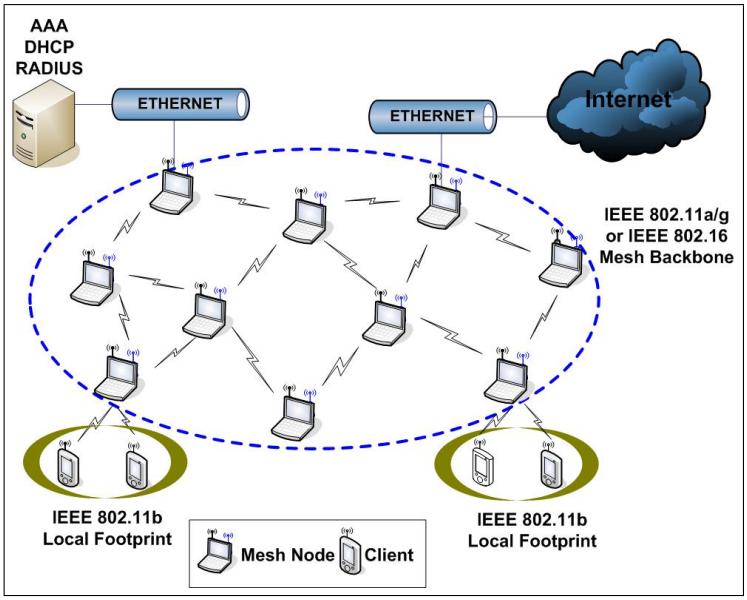


Figure 1.1: Example of WMN [1]

The scheduling rules are defined by the enforced Interference Model. Due to its simplicity, the Wireless Link Scheduling problem has been broadly approached using the Protocol Interference Model. Basically, under that protocol model, a successful transmission occurs when the intended receiving node falls within the transmission range of its sender node and falls outside the interference ranges of other non-intended senders [2]. Usually, a conflict graph is used to model such interference constraints and the problem can be reduced to the problem of vertex or edge-coloring the conflict graph. However, the Protocol Model is not realistic because it does not consider the cumulative interference of the many other possible simultaneous transmissions' contributions.

The Physical Interference Model (PIM), or the SINR Model, represents wireless communication considering different kinds of interference constraints, as we will discuss in details in Chapter 4. Although it makes a more realistic modeling possible, it also considerably increases the problem's complexity. There are many approaches to this problem in the literature and we present some of them in Chapter 2. An intuitive strategy would be representing the WMN as a graph and finding the optimal edge-coloring. The total number of colors used in the final coloring is the number of timeslots. This strategy is actually adopted by many other works. Here, we introduce in Chapter 3 the concept of edge-multicoloring and try to use this idea do model the Link Scheduling problem.

The Link Scheduling model based on edge-multicoloring can provide the optimal scheduling, promising even better results than edge-single-coloring schedulings. However, the model has an exponential size complexity due to combinatorial explosion. In theory, there is no way to escape from the exponential complexity. However, we use the

PIM's hereditary unfeasiblity property, also described in Chapter 4, to efficiently enumerate the feasible sets of links. Therefore, the model's applicability is directly connected to how well we can perform the enumeration in practice. So, the first objective is to understand the enumeration's performance variation for networks with different structures. Furthermore, we want to understand for what network structures the proposed technique is a good fit.

Regardless of its practical applicability, the second objective is observing how better are the schedulings produced by the model's solution. This objective can be reached by analyzing the overall performance from different perspectives. For example, we need to know how many networks actually admit a multicoloring-based scheduling. Also, how better is a multicoloring-based scheduling compared to the single-coloring based ones. Furthermore, how the different kinds of schedulings influence the networks' capacities. All these questions are addressed by an experiment consisting in applying the model to a huge variety of random networks and observing some key quantities such as running times and objective functions.

The remainder of this dissertation is organized as follows. Chapter 2 describes some other work on the Link Scheduling problem and its variants. Chapter 3 provides an overview to some useful theory and introduces the Fractional Edge-Coloring Problem. Chapter 4 defines the PIM adopted in this work, formalizes the Link Scheduling problem as an optimization problem and shows how to build the model enumerating the feasible sets of links. Chapter 5 describes the experiments and the results are discussed. Finally, Chapter 6 summarizes the findings and makes the last comments on this work.

Chapter 2

Literature Review

The Link Scheduling problem has many different flavors and solution approaches. This chapter is dedicated to listing some interesting and relevant work in this area. In time, we indicate the differences and the similarities between them and the present work.

BJÖRKLUND *et al.* [3] proposes a STDMA method very similar to ours. They tackle the Link Scheduling problem under the Physical Interference Model using a column generation method on the linear relaxation of the IP model for the respective Set Covering problem. They guarantee a very tight bound to the optimal scheduling. However, the biggest difference to the present work is that fractional solutions are considered unfeasible schedulings in this work. For us, fractional solutions are desirable because they lead to potentially better schedulings, as we will demonstrate throughout this text.

The second work has a similar problem set up. GANDHAM *et al.* [4] tackles the Link Scheduling problem in its TDMA form so basically links need to be allocated to time slots. As a first difference, this work does not consider physical interference and limited decoding capabilities. The only constraint is the exclusive dedication of a node to a specific transmission within a time slot, so the communication constraints are purely defined by the graph structure. The proposed technique is based on edge-coloring the graph representing the transmissions for a period of time. Authors propose a distributed algorithm to find a scheduling using at most $\Delta + 1$ colors in polynomial time. As we will discuss in further chapters, this target number is actually the theoretical upper bound for the chromatic index so it is indeed an impressive result.

Another relevant work, in spite of using the same WMN environment to define the problem constraints, is an application-driven approach. That choice is good on the one hand because the application characteristics helped DEZFOULI *et al.* [5] to develop two classes of algorithms. Both algorithms are distributed but one class allows nodes to set their own channels, while the second class queries the neighborhood. On the other hand, this is a technique with limited application scope, basically for data gathering in WSNs.

However, they could significantly increase the network throughput.

Centralized polynomial scheduling algorithms are proposed by HAJEK e SASAKI [6]. Besides the centralized approach, this work also deals with how data is forwarded in a WMN. Again, two versions of algorithms are presented. In the first one, nodes are concerned with their own interests and find the best way to allocate their transmission pools to the available timeslots. The forwarding issue is addressed by the second class of algorithms that determines the minimum scheduling for situations where communication spans multiple hops.

One of the first works to use the initials MANET for Mobile Ad-Hoc Networks was that of FANG e BENSAOU [7]. It uses the idea of conflict graphs to represent the harmful direct interference between communication links. Maximal cliques in the conflict graph determines what they called contention context and are used as constraints in an optimization formulation. This formulation is based on non-cooperative and cooperative games frameworks and the resulting models are solved using Lagrangian relaxation. In the end, a distributed medium access control algorithm that uses this technique is described providing a fair scheduling. Finally, the second algorithm is based on cooperative game and, although the fairness concept is no longer valid, nodes face an on-demand competition for the medium and this considerably increase the network throughput.

Considering this modest selection of related works, WAN *et al.* [8] is the first one that explicitly uses the Physical Interference Model (PIM) in the problem formalization. As the authors emphasize in the paper, the inclusion of the PIM makes the problem even harder to address. They propose an approximative algorithmic approach to the Link Scheduling problem for different kinds of WMNs and multihop-based architectures. However, the authors claim that the physical interference and its strong combinatorial nature invalidates all traditional graph coloring techniques. As we will show in the present work, they are only partially right given that several traditional coloring concepts and tools helped us devising our novel link scheduling technique.

Chapter 3

Fractional Edge-Coloring Problem

This chapter presents an overview of some theoretical concepts useful to understand the model proposed in Chapter 4. First, some concepts and techniques of Linear Optimization and Traditional Graph Coloring are reviewed. The last and most important topic is the Fractional Coloring that works as a framework for the proposed model.

3.1 Optimization Preliminaries

Models for various problems are frequently described through accompanying mathematical formulation for them. The formulation specifies the optimization direction (minimization or maximization), the objective function to be optimized, and a set of constraints that capture the essence of the problem. For simplicity, we will call these problems/formulations as "optimization problems".

3.1.1 Linear Programming

Linear Programming (LP) is a set of techniques used to solve formulations that are restricted to linear objective functions and constraints. A linear program in standard form is described as

$$\begin{aligned} (P) \quad & \min z = \mathbf{c}^T\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3.1}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$.

For program P, $\mathbf{c}$ is the objective function's coefficients array, $\mathbf{b}$ is the constraint's boundaries array, and $\mathbf{A}$ is the constraint's coefficients matrix. The set of variables are represented by the $\mathbf{x}$ array. Let $\mathbb{X} = \{\mathbf{x} : x \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ be the Feasible Region, i.e., the set of points in $\mathbb{R}^n$ that satisfy all constraints of the program. Accordingly, an optimal solution for it must necessarily belong to $\mathbb{X}$. If $\mathbf{x} \in \mathbb{X}$, then $\mathbf{x}$ is a feasible solution for P. If $\mathbf{x}^* \in \mathbb{X}$ and $\mathbf{c}^T\mathbf{x}^* \leq \mathbf{c}^T\mathbf{x}, \forall \mathbf{x} \in \mathbb{X}$, then $\mathbf{x}^*$ is an optimal solution.

After applying a solution method, e.g. the Simplex method, three outcomes are possible: (i) the model has one or more optimal solutions; (ii) the model is infeasible and therefore has no feasible solutions ($\mathbb{X} = \emptyset$); or (iii) the model is unbounded and no constraint is able to stop the objective function from being improved – if it is a minimization problem, then $\mathbf{c}^T x \to -\infty$, else $\mathbf{c}^T\mathbf{x} \to +\infty$. Figure 3.1 shows an example for each of the possibilities above.

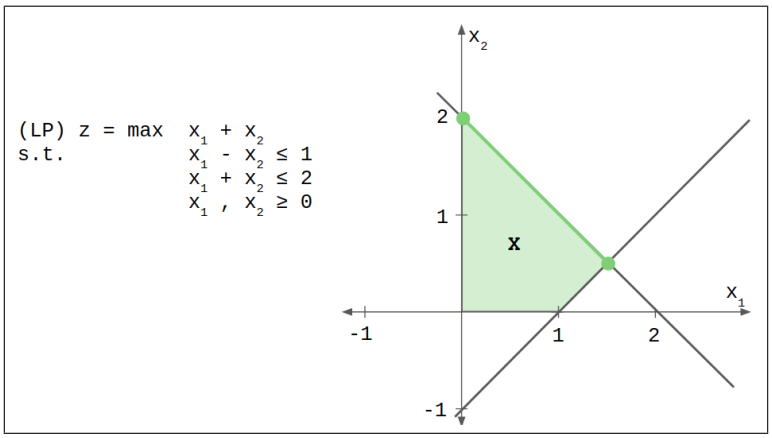
Figure 3.1(a) has a well defined feasible region X bounded by its constraints. In this case, any values for $\mathbf{x}$ on the green line is an optimal solution with $z = 2$. Notice that the feasible region is convex so the Simplex method will return only one of its vertices. Figure 3.1(b) has two separated feasible regions X_1 and X_2. Each region is determined by the respective constraint. However, because there is no intersection between these regions, the problem's feasible region is empty ($X = X_1 \cap X_2 = \emptyset$). Lastly, Figure 3.1(c) shows an example of an unbounded model. In this case, because the feasible region is not properly bounded by its constraints, the objective function can always increase its value. It is desirable that formulations never lead to unfeasible or unbounded models.

3.1.2 Duality

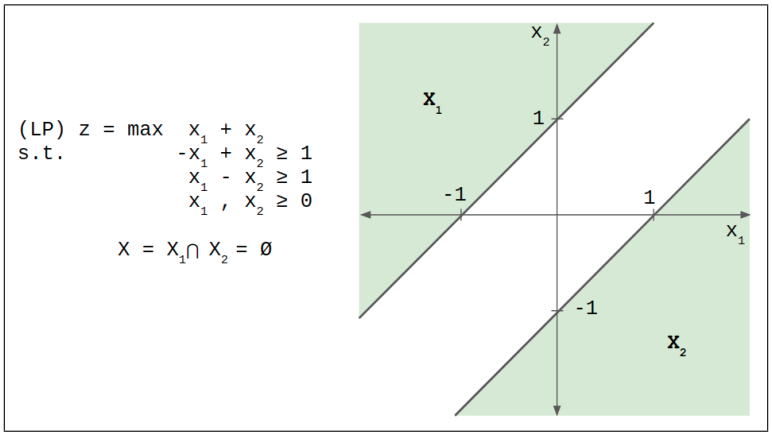
It is possible to explore the opposite optimization direction writing a formulation in its dual form. The dual of problem P in (3.1) is formulation D in (3.2). Note that D has no non-negativity constraint such as $\mathbf{x} \geq \mathbf{0}$ which means that variables are free to assume any values. See LEE [9] for instances for a systematic procedure to perform primal-dual conversion. The conversion procedure may be repeatedly applied so that one returns to P after applying it to D. Thus, it is reasonable to affirm that the dual's dual is the primal.

$$\begin{aligned} (D) \quad & \max z = \mathbf{y}^T\mathbf{b} \\ s.t. \quad & \mathbf{y}^T\mathbf{A} \leq \mathbf{c}^T \end{aligned} \tag{3.2}$$

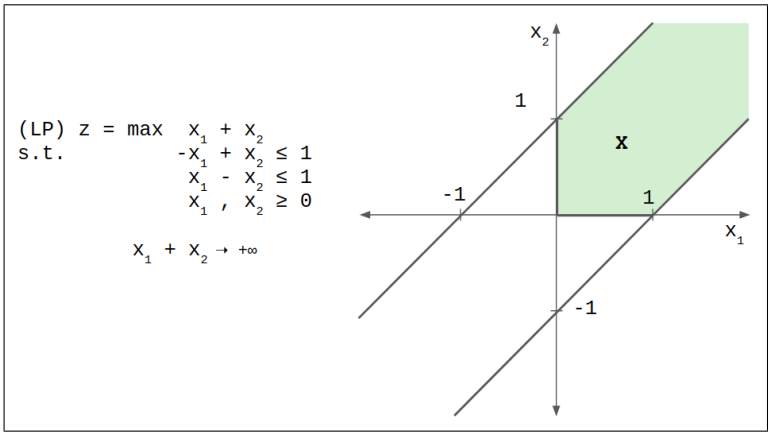
In Linear Programming Theory, there are some important theorems and corollaries regarding Duality. See LEE [9] once again for more interesting and useful ones. The



(a)



(b)



(c)

Figure 3.1: Examples of Solution Possibilities

Existence Theorem for optimization problems with convex feasible solution regions summarizes the most relevant ones for this work.

The Existence Theorem: Let P and D be a primal-dual pair. Only one of the following sentences holds:

- P (D) is unlimited $\Rightarrow$ D (P) is unfeasible
- P and D are unfeasible
- P and D have optimal solutions and, in this case, $z_P^* = z_D^*$

3.1.3 Integer Programming

Important implications occur when the variables' domain is changed. When the variables assume integral values, i.e. $\mathbf{x} \in \mathbb{Z}^n$, the problem is called Integer Linear Programming problem, or simply Integer Programming (IP). Equation 3.3 shows a general formulation for IP problems.

$$\begin{aligned} (P) \quad & \min z = \mathbf{c}^T\mathbf{x} \\ s.t. \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}^n, \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3.3}$$

The discretized IP domain brings combinatorial characteristics to the problem, which makes its solution way more difficult to find than for most pure LP problems. Because they are similar in form, one can think that a rounded IP solution could be inferred from its LP version. It is true for some cases but it usually leads to very bad approximations. A linear relaxation for an IP problem can be roughly defined as its correspondent LP version. Figure 3.2 shows how the variables' domain can change the model structure and solution.

This is a classic example from WOLSEY [10] where both real and integer domains are represented in the same chart. Specifically, the integer domain is represented by the small circles. The green region is the LP feasible region while the green circles compose the IP feasible region. The IP solution is the dark green point on the bottom $(5, 0)$ and its linear relaxation solution is the point $(1.95, 4.92)$ on the top. The closest integer point to the linear relaxation solution is the point $(4, 2)$. Although they result in similar objective functions, both the LP and the rounded solutions are quite different from the IP solution.

Due to its inherent difficulty to find optimal IP solutions, sometimes it is enough to determine bounds on their value, which can be much easier to obtain than solving the problem itself. For example, for a minimization IP problem, any feasible solution for it

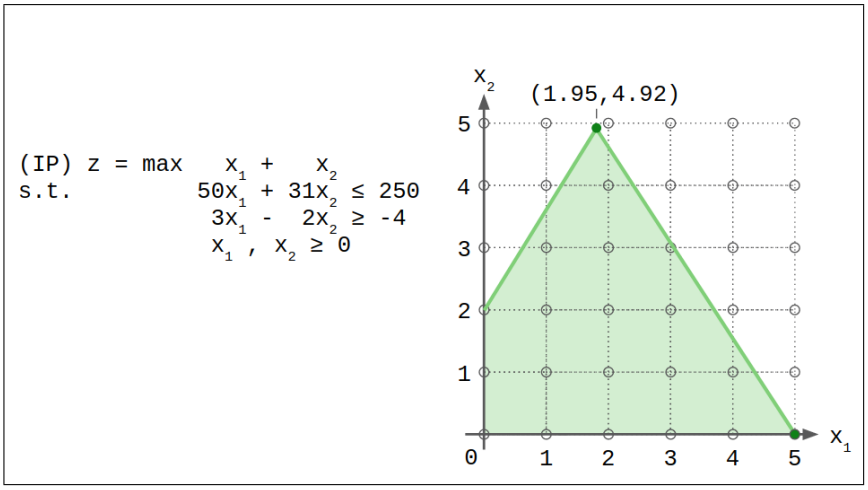


Figure 3.2: Example of a model with real and integer domains

already provides an upper bound while its LP relaxation dual solution provides a lower bound. Another well-known strategy to solve IP problems is to adopt Branch and Bound algorithms.

3.1.4 Combinatorial Optimization

Consider a finite set $N = \{1, 2, 3, ..., n\}$ and a weight function $c(i)$, defined for each element $i \in N$. Let $\mathcal{C}$ be the set of all combinations of elements in N. Thus, $|\mathcal{C}| = 2^n$. Some elements of $\mathcal{C}$ satisfy some constraints established by the problem definition and are called feasible. The set of all feasible elements is the feasible set, denoted $\mathcal{F}$, such that $\mathcal{F} \subseteq \mathcal{C}$. The problem of finding an $F \in \mathcal{F}$ with the minimum total weight is called a Combinatorial Optimization Problem (COP) and can be formalized as

$$(COP) \quad \min_{S \in \mathcal{C}} \{ \sum_{i \in \mathcal{S}} c(i) : S \in \mathcal{F} \}. \tag{3.4}$$

This problem could be solved by individually verifying if each combination $S \in \mathcal{C}$ is feasible and choosing the one with the lowest total weight. This brute-force approach is not a good idea because all $|\mathcal{C}| = 2^n$ combinations must be tested, making this search not suitable for most applications. This phenomenon is called Combinatorial Explosion.

It is common to rewrite a COP as a Binary IP (BIP) problem. In this case, an incidence vector $\mathbf{x(C)} \in \{0, 1\}^n$ represents a combination of elements C, such that $x(C)_i = 1$ if element $i \in N$ is in C, and $x(C)_i = 0$, otherwise. Equation 3.5 shows a BIP formulation for a general COP.

$$
\begin{aligned}
(BIP) \quad & \min z = \mathbf{c}^T\mathbf{x} \\
s.t. \quad & \mathbf{Ax} = \mathbf{b} \\
& \mathbf{x} \in \{0,1\}^n
\end{aligned}
\tag{3.5}
$$

The same solution strategies used in IP problems can be used to solve COP/BIP problems. There are some cases where Greedy and Dynamic Programming algorithms can be used to find good approximations or, sometimes, effectively solve such problems as well.

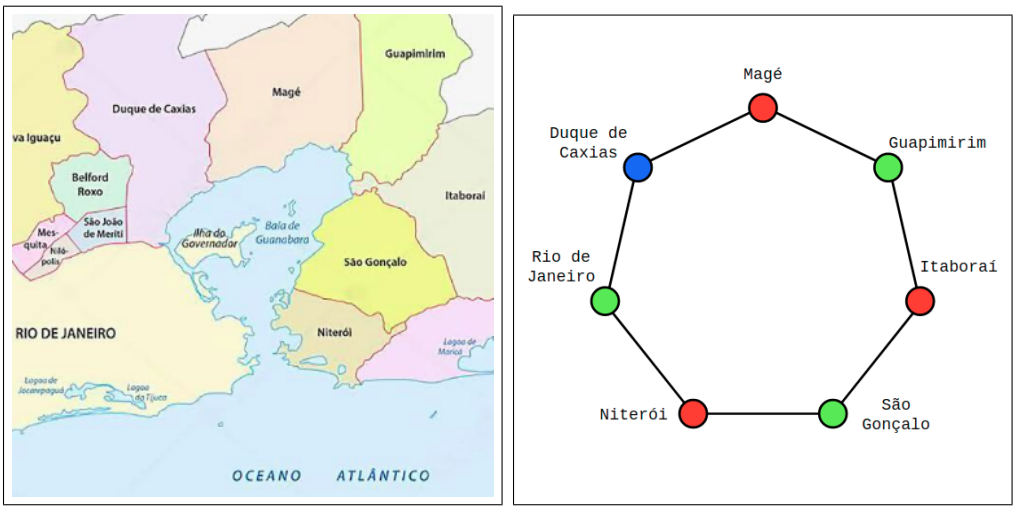
3.2 Traditional Coloring

Graph Theory provides the mathematical abstraction used to model relationships between objects, e.g. networks in general. Therefore, models for many different problems and their solution techniques and algorithms were conceived in this area. In order to understand the Fractional Edge-Coloring Problem introduced in the next section, a few notations and definitions must be presented first.

Consider a graph $G = (V, E)$ with the set of vertices V and the set of edges E. Let $|V| = n$ and $|E| = m$. The degree d_v of a vertex v is the number of incident edges or the number of its neighboring vertices. The maximum degree $\Delta(G)$ of a graph G is defined as $\max\{d_v\}, \forall v \in V$. By definition, a matching $M \subseteq E$ of G is a set of disjoint edges (no two edges are incident to a same vertex).

Graph coloring is usually presented using the canonical Cartography's example, in which coloring a map is essentially assigning colors to geographical units (e.g. countries, cities, etc.), in a way that neighboring units get different colors. Let the map be represented as a graph $G = (V, E)$ where vertices are countries and an edge is placed connecting two vertices if their respective countries are neighbors. Equivalently, colors are assigned to vertices such that adjacent vertices get different colors. In this case, the actual color assignment is named vertex-coloring.

Regarding the vertex-coloring task, there exists a decision and an optimization problem. The decision problem consists in determining if G can be colored using k colors. If so, G is called k-colorable. The optimization problem consists in finding the chromatic number $\chi(G)$, or simply χ when the context is clear, that is the least number of colors needed to color all vertices of G once. The optimization problem is called the Vertex-Coloring Problem.



(a) (b)

Figure 3.3: Map coloring of all cities by the Baia de Guanabara that share borders

Figure 3.3 shows how coloring a map and a graph's vertices are equivalent tasks. The map coloring is only one very simple example of how real problems can be modeled as a graph coloring instance. Figure 3.3(a) shows a colored map and 3.3(b) shows its respective vertex-coloring version.

On the other hand, the edge-coloring is concerned with assigning colors to the edges instead of vertices. In this case, the edge-coloring constraint is defined such that no two edges incident to a same vertex can get the same color. From another perspective, all edges with the same color are necessarily disjoint. Both decision and optimization problems are also valid for this task. The optimization version is named Edge-Coloring Problem and its solution provides the chromatic index χ'. From now on, all discussions will be focused on edge-coloring due to its extreme relevance to this work.

The Edge-Coloring problem can be analyzed from the perspective of matchings. Notice that the edge-coloring constraint is equivalent to the matching definition so, basically, a color is a matching of G. Therefore, this problem can be reinterpreted as the problem of finding a partition of the edges of G into the smallest possible number of matchings. Given that $\mathcal{M}$ is the set of all possible matchings for G, the Edge-Coloring problem can be formalized using the optimization formulation presented in Equation 3.6.

$$\begin{aligned} (IP) \quad & z = \min \sum_{M \in \mathcal{M}} x_M \\ s.t. \quad & c(e) = \sum_{M \ni e} x_M = 1, \forall e \in E \\ & x_M \in \mathbb{Z}, x_M \geq 0, \forall M \in \mathcal{M} \end{aligned} \tag{3.6}$$

This formulation has $|\mathcal{M}|$ variables and m constraints. Variable x_M is related to matching $M \in \mathcal{M}$ and indicates whether this matching is in the resulting partitioning ($x_M = 1$) or not ($x_M = 0$). Each constraint $c(e)$ corresponds to an edge $e \in E$ and is the sum of all variables that corresponds to matchings that contain e.

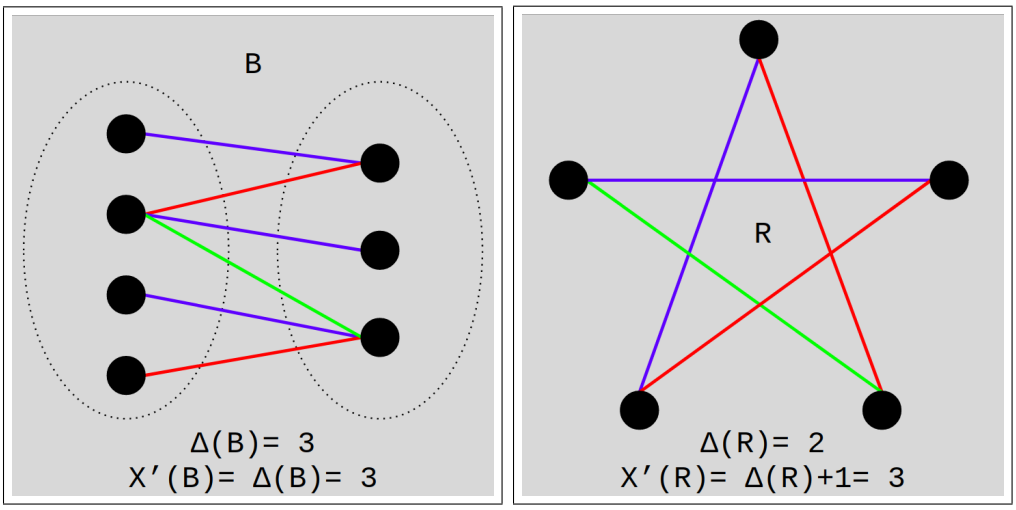
Because $c(e) = 1, \forall e \in E$ and all variables are non-negative integers, it is actually a BIP problem so indeed $x_M \in \{0, 1\}$. As a consequence, exactly one variable equals 1 in each $c(e)$ and this means that we are guaranteeing that e will be considered only once in the final partition.

The number of variables equal to 1 is the number of matchings in the final partition. Since the objective function is the sum of all variables, it represents the partition size. Moreover, as we are minimizing this function, we are actually finding a partition of the edges of G into the smallest possible number of matchings. Finally, using the matching-color equivalence, we know that each matching used in the optimal partition represents one color in the optimal coloring, thus $z^* = \chi'$.

The optimization problem is easy to solve analytically for simple graphs. However, this is an NP-hard problem so it is unknown if there exists a polynomial algorithm to solve it for arbitrarily large graphs. So, it would be interesting to have efficient ways to estimate the chromatic index using only the graph's structure and its properties.

The graph structure can provide information on lower and upper numerical bounds for χ'. Following the definition, all edges incident to a specific vertex must get different colors. Then, at least $\Delta(G)$ colors are needed. In worst cases, one different color is assigned to each edge, so m colors would be used. Hence, a first and naive boundary interval for χ' is $\Delta(G) \leq \chi' \leq m$.

VIZING [11] proved that all graphs could be categorized in two different classes. All class one graphs have $\chi' = \Delta$ and all class two graphs have $\chi' = \Delta + 1$. This allowed the boundary interval to be narrowed to $\Delta \leq \chi' \leq \Delta + 1$. He demonstrated that all bipartite graphs are class one while all regular graphs with an odd number of vertices are class two. However, deciding if a general graph is class one or two is also NP-hard. Figure 3.4 shows an example of each graph class.



(a) (b)

Figure 3.4: Vizing's two class of graphs and its chromatic indices

Because the problem is NP-hard, solving it directly through an optimization method such as Branch and Bound may not be encouraged for some practical applications. For not so large instances of this problem, even building the optimization model would demand an extremely large number of variables ($O(2^m)$) and an impractical amount of computational resources. If the solution must be obtained in a relatively short amount of time, the optimal coloring may not be a requirement; sometimes a good coloring is enough. For these cases, approximative and stochastic approaches can be good fits.

3.3 Fractional Coloring

Going deeper into edge-coloring, consider now that each edge of G is allowed to get multiple colors. This is a case of edge-multicoloring and, by solving its optimization problem, the fractional chromatic index χ'_f can be obtained; thus it is named Fractional Edge-Coloring or Edge-Multicoloring Problem. According to SCHRIJVER [12], the Fractional Edge-Coloring problem is defined as the LP formulation in Equation 3.7.

$$\begin{aligned} (LP) \quad & \min z = \sum_{M \in \mathcal{M}} x_M \\ s.t. \quad & c(e) = \sum_{M \ni e} x_M = 1, \forall e \in E \\ & x_M \in \mathbb{R}, x_M \geq 0, \forall M \in \mathcal{M} \end{aligned} \tag{3.7}$$

In fact, Equation 3.7 is the linear relaxation of the traditional Edge-Coloring problem formulation given in Equation 3.3. To prove that the proposed formulation indeed finds the optimal coloring with χ'_f, consider the following algebraic procedure.

First, if $x_M \in \mathbb{R}^+$ and $c(e) = \sum_{M \ni e} x_M = 1, \forall e \in E$, then $x_M \in \mathbb{Q}^+ : x_M \in [0, 1]$. Indeed, if x_M admits rational non-negative values, then it can be written in terms of a fraction of integers, such as:

$$x_M = \frac{p_M}{q_M} \tag{3.8}$$

where $p_M, q_M \in \mathbb{Z}^+,\ q_M \geq 1,\ p_M \leq q_M$.

Now, consider a generic solution of this problem. If we analyze the optimal objective function only in terms of variables in their fractional form, we get:

$$z^* = \sum_{M \in \mathcal{M}} \frac{p_M}{q_M} = \frac{p_1}{q_1} + \frac{p_2}{q_2} + ... + \frac{p_{|\mathcal{M}|}}{q_{|\mathcal{M}|}} \tag{3.9}$$

This is a sum of $|\mathcal{M}|$ rational terms so there is an integer q that is the least common multiple of all q_M. In other words, q can be divided by each $q_M, \forall M \in \mathcal{M}$ such that $\frac{q}{q_M} \in \mathbb{Z}^+$. Thus,

$$z^* = \frac{1}{q} \sum_{M \in \mathcal{M}} \frac{p_M}{q_M} q = \frac{\frac{p_1}{q_1} q + \frac{p_2}{q_2} q + ... + \frac{p_{|\mathcal{M}|}}{q_{|\mathcal{M}|}} q}{q} \tag{3.10}$$

It is important to notice that $\frac{p_M}{q_M} q$ is also an integer. Now, consider the sum of integral terms in the numerator of Equation 3.10 and focus on those terms belonging to an arbitrary constraint $c(e)$ related to edge e.

$$\sum_{M \ni e} \frac{p_M}{q_M} q = q \sum_{M \ni e} \frac{p_M}{q_M} = qc(e) = q \tag{3.11}$$

Equation 3.11 says that each constraint contributes with q colors to its respective edge in the final solution. However, because some constraints share variables, part of their individual contribution overlap the others'. Therefore, the numerator of Equation 3.10 represents the total number of colors used and is denoted by L. The optimal objective function can be rewritten as

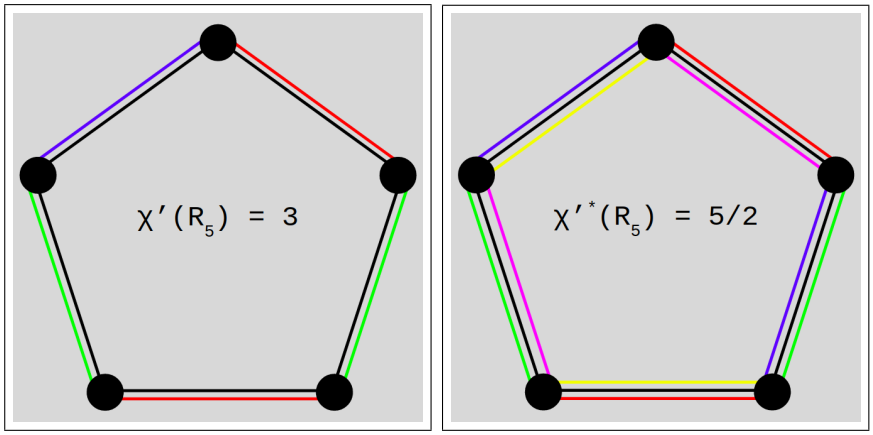
$$z^* = \frac{L}{q} \Rightarrow \chi'_f = \frac{L}{q} \tag{3.12}$$

where $L = \sum_{M \in \mathcal{M}} \frac{p_M}{q_M} q$.

If q is the number of colors assigned to each edge in the optimal solution, the variable $x_M = \frac{p_M}{q_M}$ represents the proportion of q provided by matching M. For example, if $x_M = \frac{1}{3}$, then matching M provides to its edges $\frac{1}{3}$ of the q colors they need. So, the other $\frac{2}{3}$ of q are covered by other matchings. The number $\frac{p_M}{q_M} q$ is the multiplicity of matching

M in the final solution. In other words, matching M is repeated $\frac{p_M}{q_M}q$ times and each of its instances represents a different color.

Figure 3.5 shows the difference between single-coloring and multicoloring an R_5. Both colorings are optimal and their chromatic indices are also indicated in the respective figure. In this example, the multicoloring in Figure 3.5(b) results in $\chi'^* = 2.5 = \frac{5}{2}$, so $L = 5$ colors are used in a way that each edge gets $q = 2$ different colors.



(a) (b)

Figure 3.5: How the edges of an R_5 can be colored and multicolored

It is important to prove that this formulation will not lead to unfeasible or unbounded models. Let the matchings consisting of a unique edge be called basic sets; so, there exists m basic sets. Now, split the columns of matrix A into B and N for the basic and non-basic sets, respectively. Also, split the columns of x such that $x = [x_B|x_N]$. Notice that the columns of B can be arranged as an identity matrix. That being said, B is actually a base for the model and, consequently, $x_B = \mathbf{1}$ and $x_N = \mathbf{0}$ is a feasible solution. In fact, such solution represents the worst coloring possible, in which every edge gets a different color. So, because there is always at least one feasible solution, the model can never be unfeasible. Moreover, it was shown that $0 \leq x \leq 1$, so the actual solution is contained within a feasible region that can be represented as an m-dimensional polyhedron of side 1; more specifically, in one of its vertices. Therefore, models produced by this formulation can not be unbounded as well.

Now that we presented the optimization formulation and proved it is conceptually valid, it is important to discuss its solution process. Finding χ'_f can lead to three different color assignments. First, if $q = 1$, edges get only one color each so the final color assignment is a traditional edge-coloring (TC). If $q > 1$, there are two different edge-multicoloring cases. In the weak multicoloring (WM) case, $\chi'_f \in \mathbb{Z} : \chi'^* = \chi'$ so, although

edges get multiple colors, the same χ'_f could be achieved through q repetitions of a TC. Finally, the strong multicoloring (SM) is characterized by $\chi'_f \in \mathbb{Q} : \chi'_f < \chi'$.

The Fractional Edge Coloring Problem can also be modeled using the Equation 3.7's dual form as presented in Equation 3.13. Duality theory guarantees that both formulations have the same solution and, consequently, result in the same coloring.

$$\begin{aligned} (LP) \quad & \max z = \sum_{e \in E} x_e \\ s.t. \quad & \sum_{e \in M} x_e \leq 1, \forall M \in \mathcal{M} \\ & x_e \in \mathbb{R}, \forall e \in E \end{aligned} \tag{3.13}$$

Unlike the traditional Edge-Coloring Problem, χ'_f can be found in polynomial time [13]. PADBERG e RAO [14] use the Edmonds' Fractional Edge-Coloring Theorem [15] to propose a polynomial algorithm to find χ'_f. Then, the dual formulation is polynomially solved applying the Ellipsoid method [16] using EDMONDS [15]'s max-weighted matching algorithm.

Chapter 4

The Link Scheduling Problem

This chapter provides a detailed description of the Link Scheduling problem and introduces a method for solving it using a model based on edge-multicoloring. In the first two sections, we define the transmission properties and the PIM used in this work. Then, we prove that the LP formulation presented in Chapter 3 can also be used to model the Link Scheduling problem. Afterwards, we describe a PIM-based enumeration algorithm capable of finding all feasible sets of links in a faster way and, consequently, building the model more efficiently. Finally, we present a simple example of how the optimal scheduling can be found using this technique.

4.1 Transmission Properties

In a WMN with a set of nodes $\mathcal{N}$ and a set wireless links $\mathcal{L}$, a link $i \in \mathcal{L}$ is established from a sender node $s_i \in \mathcal{N}$ to a receiver node $r_i \in \mathcal{N}$. A transmission consists of a node emitting an electromagnetic signal carrying coded information through a shared physical medium, e.g., through the air or underwater. A wireless link is said to be active when a transmission is actually taking place. The signal is sent to multiple directions (most cases use omni-directional antennas) and all nodes sharing the medium are able to detect and possibly decode it. The generated signal gradually loses its power according to the distance traveled. This work uses the Log-Distance Path Loss Propagation Model as described by RAPPAPORT [17] and a simpler version is presented in Equation 4.1.

$$RP(s, r) = \frac{TP}{d_{s,r}^{\alpha}} \tag{4.1}$$

This model represents how much of a signal transmitted by the sender s at a transmission power TP is attenuated on the way to reach the receiver node r, both separated by a distance of $d_{s,r}$. The signal reception power RP on r is a result of a power decay

in α, the propagation exponent. The value for α varies according to the environment in which the transmission is occurring. RAPPAPORT [17] also list some examples of typical values for α in various radio environments shown in Table 4.1.

Environments	Path Loss Exponent (α)
Free space	2
Urban area cellular radio	2.7 - 3.5
Shadowed urban cellular radio	3 - 5
In building line-of-sight	1.6 - 1.8
Obstructed in building	4 - 6
Obstructed in factories	2 - 3

Table 4.1: The values of α for specific environments

Let β denotes the node's sensibility and represents the minimum reception power required for the receiver to be able to decode the signal's information. If the receiver node is too far from the sender, the incoming signal's power may have suffered severe reduction due to attenuation. So, it is reasonable to think that a sender node has a transmission range such that all other nodes within this area can safely decode its signal. Henceforth, for simplicity, all nodes have the same TP such that their transmission ranges are all the same. Therefore, a node's neighbors are those nodes within its transmission range in an interference-free environment. The transmission range, denoted by ρ, can be calculated in meters using Equation 4.2.

$$\rho = 10^{\frac{TP-\gamma-\beta}{10\alpha}} \tag{4.2}$$

The noise floor, γ, is the amount of attenuation caused by other electromagnetic sources present in the environment, e.g. light and radiation. Besides γ, because receivers have limited decoding capability bounded by β, transmissions can also interfere to each other. The next section describes the interference model used in this work.

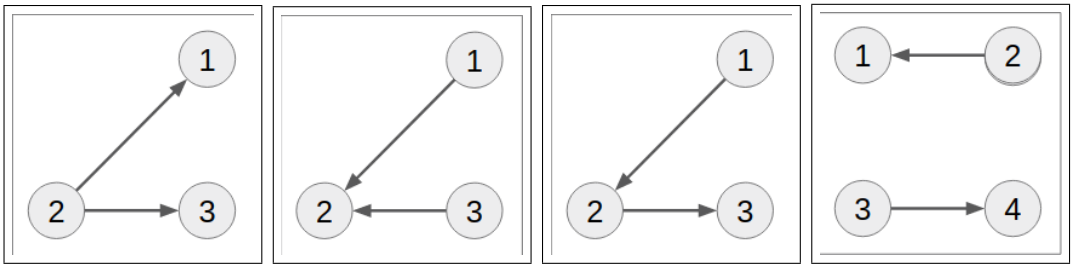
4.2 The Physical Interference Model

The Physical Interference Model (PIM) establishes interference constraints for a wireless communication environment. The primary and secondary constraints presented here are the same as those defined by GUPTA e KUMAR [18].

4.2.1 Primary Constraint

The Primary Constraint assumes that nodes are half-duplex. This means that nodes can not transmit and receive signals at the same time; a node is either a sender or a receiver

at any given moment. Moreover, in a given transmission period, nodes are dedicated to a single communication link. Figure 4.1 depicts four different transmission scenarios.



(a) (b) (c) (d)

Figure 4.1: Different transmission scenarios

In scenario 4.1(a) and 4.1(b), although nodes have unique roles as either sender or receiver, a same node is making part of two different transmissions simultaneously, so they violate the dedicated communication rule. Similarly, scenario 4.1(c) does not satisfy the primary constraint because node 2 is both sender and receiver, violating the half-duplex rule. The only allowed scenarios are those following the same structure as in 4.1(d), which has disjoint links.

4.2.2 Secondary Constraint

The Signal-to-Interference-plus-Noise Ratio (SINR) is a quantity that determines how like is a receiver r_i to decode the signal transmitted by a sender s_i. Let $C \subseteq \mathcal{L}$ be a set of links representing all transmissions that must be done within the same transmission period. The SINR of a link $i \in C$ is given by Equation 4.3. The SINR relates the power of the signal transmitted in i to all interference sources. As was mentioned before, γ is the noise portion provided by the environment. Furthermore, the $\sum_{j \in C, j \neq i} RP(s_j, r_i)$ part is the total interference caused by the senders of all other links active along with i.

$$SINR(i, C) = \frac{RP(s_i, r_i)}{\gamma + \sum_{j \in C, j \neq i} RP(s_j, r_i)} \tag{4.3}$$

The antenna sensitivity β can also be used as a numerical boundary for tolerated interference. Therefore, the Secondary constraint demands that, for a given $C \subseteq \mathcal{L}$, $\forall i \in C, SINR(i, C) \geq \beta$. An $SINR(i, C) < \beta$ means that receiver r_i is being exposed to too much interference and the communication is compromised.

4.2.3 Feasible Sets of Links

The PIM determines that a set of links $C \subseteq L$ is feasible if both primary and secondary constraints are satisfied. In summary, $C \subseteq L$ is feasible if, and only if,

1. $\forall i, j \in C$, i and j are disjoint; and
2. $\forall i \in C, SINR(i, C) \geq \beta$

4.3 The Link Scheduling Problem

Consider an application that has a pool of transmissions to be made within a specific period of time T. The transmission period can be split into L equally long timeslots, also known as scheduling unit. Now, links can be assigned to these timeslots in a way that a group of active links in a same timeslot is isolated from links of different timeslots. This reduces the interference between links and allows them being active in a same transmission period, although during shorter individual transmission times. There are two extreme situations worth being analyzed.

First, assume $L = 1$. This means that the scheduling unit has only one timeslot and all links are active at the same time. For usual transmission conditions, there would be so much interference that almost no receiver would be able to decode incoming data. Obviously, this network has a poor performance, presenting a very low network capacity.

Conversely, assume $L = m$. This means that the scheduling unit has m timeslots, one for each link. In this case, timeslots are too short, so is the data transmission duration in a given communication link. Besides, senders would have to wait too long to have a chance to transmit again. This makes a considerable part of the network idle for most of the transmission period, resulting in waste of network resources. Consequently, such network's capacity is also very low.

Therefore, there is a trade-off between the number of timeslots and the overall network capacity. In summary, if L is too large, timeslots are shorter and the retransmission waiting time is longer. Else, if L is too small, the cumulative interference is stronger in the whole network, compromising the general communication between nodes. Essentially, the Link Scheduling problem aims to find the optimal point in this trade-off. In other words, it tries to find the minimum value for $L : 1 \leq L \leq m$ such that, although there may exist interference inside each timeslot, communication happens harmlessly, resulting in the maximum network capacity. Better schedulings result in higher network capacity because transmissions perform with better success rates and, consequently more data is exchanged in the network. Additionally, the problem is defined such that all links must be scheduled the same number of times during the scheduling unit.

4.4 Fractional Edge-Coloring Model

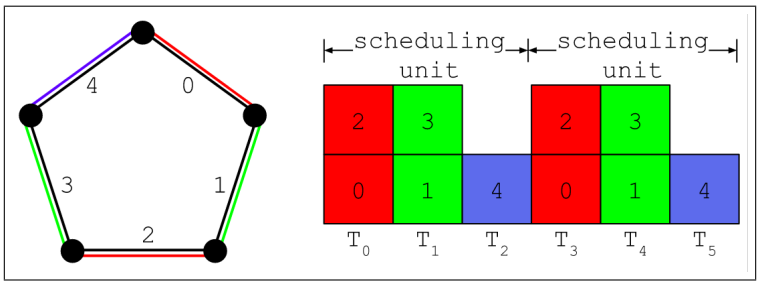
Consider a wireless network as defined in the first section. Such network can be modeled as a graph $G = (V, E)$ where $V \equiv \mathcal{N}$ and $E \equiv \mathcal{L}$. The communication directions are only useful to determine the interference components and, because it is not relevant to this part of the text, links will be safely represented as undirected edges.

Initially, let's consider a version of the Link Scheduling Problem where the PIM has only the primary interference constraint. In this oversimplified scenario, a node can only be part of a single communication link in a same timeslot. In other words, only a matching-based network structure is allowed in a same timeslot. This means that the primary interference constraint on the network induces a matching constraint over G. As we stated in Chapter 3, the edges in a same matching get the same color. Bringing it all together, a color may be interpreted as a timeslot. As long as these ideas are all equivalent, coloring the edges of G is actually determining which links are active in each time slot. Therefore, solving either the Edge-Coloring or the Edge-Multicoloring problem is equivalent to solving this loose version of the Link Scheduling problem.

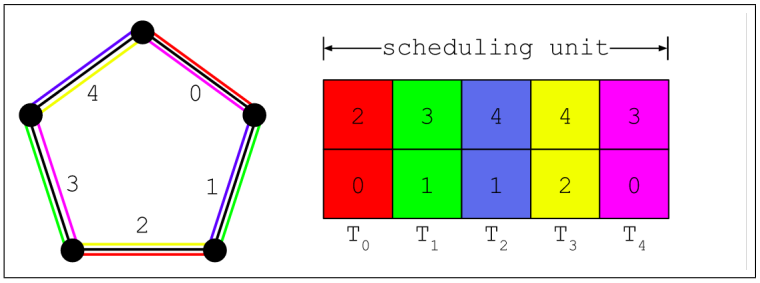
In order to help measuring a scheduling's efficiency, it is possible to use an alternative definition of network capacity. The idea is that the intensity of the data traffic in the network is related to how many times links are active during the transmission period T. Given the previously suggested equivalence between coloring and scheduling, the network capacity could be redefined as the inverse of the fractional chromatic index, i.e., $C(G) = \frac{1}{\chi'_f(G)} = \frac{q}{L}$. In other words, the capacity is the proportion of all active links in a scheduling unit. In fact, as demonstrated in Chapter 3, $\chi'_f \leq \chi'$ so the Fractional Edge-Coloring Problem solution process has better chances of producing schedulings with higher capacities.

Now, consider a network built on a ring topology represented by an R_5 graph, i.e., 5 nodes and 5 edges. For this graph, a TC and an SM examples are both given by Figure 4.2. This figure demonstrates how these colorings can be interpreted as schedulings.

In the TC example of Figure 4.2(a), three time slots $\{T_1, T_2, T_3\}$ were needed to allocate all links. Notice that T_1 and T_2 have two links while T_3 has only one and an empty link position; which could additionally host either edge 1 or 2 along with edge 4. Figure 4.2(b) shows an example of an SM that actually used all available link positions avoiding them being wasted. Moreover, in order to all links being active twice, the SM needed a scheduling unit of 5 timeslots while the TC needed two scheduling units with a total of 6 timeslots. This improvement is also reflected in the network capacity where $C_{TC} = \frac{1}{3}$ and $C_{SM} = \frac{2}{5}$ so, indeed, $C_{SM} > C_{TC}$. Accordingly, SM offers shorter schedulings than TC.



(a)



(b)

Figure 4.2: Examples of different types of scheduling on a R_5 network

So far, the version of the Link Scheduling Problem only with the primary constraint and the Fractional Edge-Coloring Problem are equivalents. Then, the LP formulation presented in Chapter 3 could be used to model both problems. However, from now on, the full PIM, with both primary and secondary interference constraints, is considered. Now, even though links are disjoint in a same timeslot, scheduling them together might not be an option due to excessive interference. To be able to keep the equivalence between coloring and scheduling problems, an adjustment is needed.

Basically, the equivalence between matchings and colors is no longer valid, so the matchings set $\mathcal{M}$ will be replaced by feasible sets of links $\mathcal{F}$. Notice that $\mathcal{F}$ is a result of applying a new interference constraint to the links in $\mathcal{M}$, so $\mathcal{F} \subseteq \mathcal{M}$. The $\mathcal{M}$ set can be safely replaced by the $\mathcal{F}$ set because $\mathcal{F}$ is still a matching, with a more limited set of links though. Therefore, the adapted LP formulation in Equation 4.4 can be used to model and solve the full Link Scheduling Problem as well.

$$
\begin{aligned}
(LP) \quad & z = \min \sum_{F \in \mathcal{F}} x_F \\
s.t. \quad & \sum_{F \ni e} x_F = 1, \forall e \in E \\
& x_F \in \mathbb{R}, x_F \geq 0, \forall F \in \mathcal{F}
\end{aligned}
\tag{4.4}
$$

This adaptation also works for IP models which is reflected in Equation 4.5.

$$
\begin{aligned}
(IP) \quad & z = \min \sum_{F \in \mathcal{F}} x_F \\
s.t. \quad & \sum_{F \ni e} x_F = 1, \forall e \in E \\
& x_F \in \mathbb{Z}, x_F \geq 0, \forall F \in \mathcal{F}
\end{aligned}
\tag{4.5}
$$

In the PIM only with the primary constraint considered initially, the coloring combinatorial nature was essentially defined by the graph's structure so the techniques presented in Chapter 3 were all valid for the Link Scheduling Problem. Now, the full PIM adds the SINR constraint, which depends on several random elements, such as nodes positions, for example. Unfortunately, it is not guaranteed that all those techniques remain valid. Moreover, it also not guaranteed that there exists a polynomial algorithmic approach to solve this problem. The optimization formulation is all we proved that is still valid. In order to find the LP models' variables and to solve the Link Scheduling problem, we propose an algorithm to enumerate all feasible sets of links for a given network. This algorithm uses the PIM characteristics to enumerate faster than pure brute force enumeration.

4.5 PIM-based Enumeration Algorithm

Firstly, to be able to reference the abstractions used to describe the algorithm, some notations are presented. Let $G = (V, E)$ be a graph representing a network such that $|V| = n$ and $|E| = m$. Moreover, let $\mathcal{C}$ be the set of all combinations of links and $\mathcal{F}$ be the set of all feasible sets of links, such that $\mathcal{F} \subseteq \mathcal{C}$.

Assume there is an $O(n)$ algorithm, named $IsFeasible(C)$, to check whether a certain combination $C \in \mathcal{C}$ is feasible or not. Theoretically, given that $|\mathcal{C}| = 2^m$, a full enumeration would take $O(n2^m)$ to accomplish the task. In practice, the PIM constraints can be used to avoid checking every single combination and, consequently, improve the enumeration performance by reducing its execution time.

Now, consider that all combinations in $\mathcal{C}$ are uniquely arranged in a depth-first structured tree. The root vertex is the empty combination. The other vertices are recursively added to the tree in a way that the vertex's children are the current combination plus another link such that no combination is repeated. Figure 4.3 shows an example for a G with $E = \{0, 1, 2, 3\}$.

Because C's children have always one more link, the interference environment is stronger than in C. The new link will only add more interference to the whole system. Therefore, if C is not feasible, its descendants are not feasible as well and this

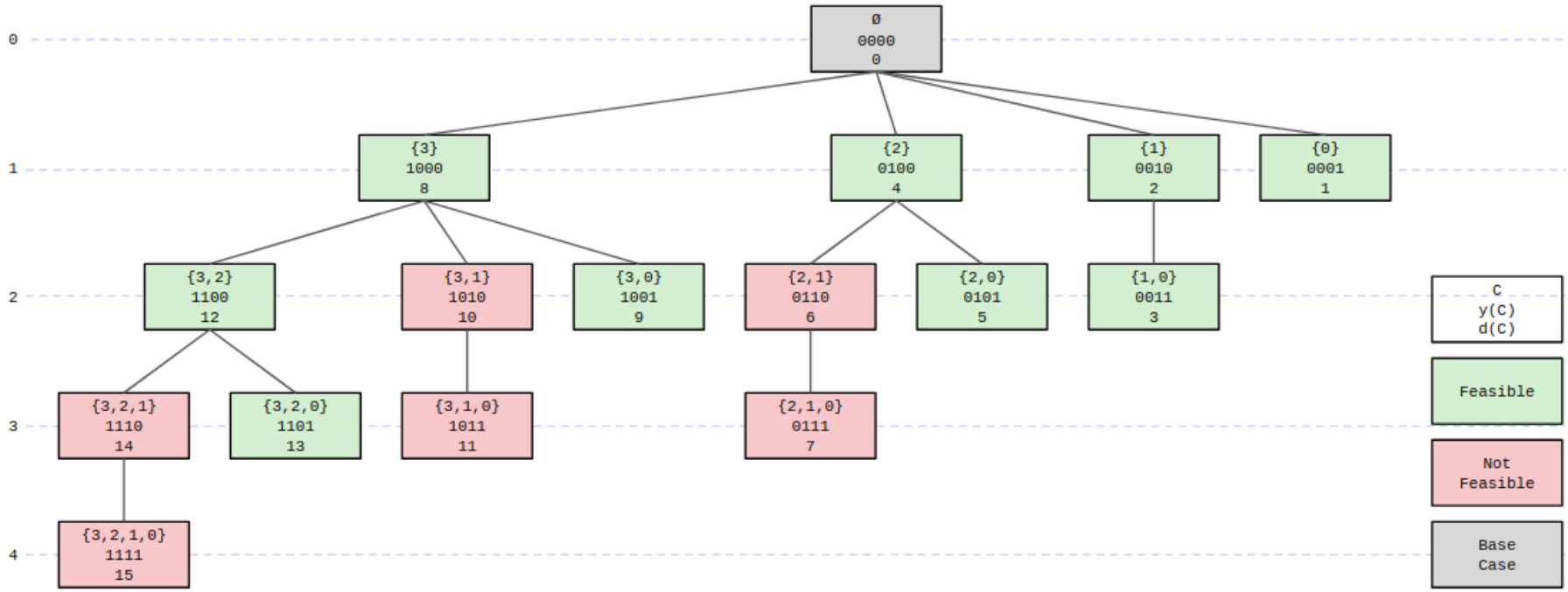


Figure 4.3: Tree of combination of links for a graph $G = (E, V)$ with $E = \{0, 1, 2, 3\}$

property is called hereditary infeasibility. The tree traversal can be performed faster once several combinations are pruned due to hereditary infeasibility.

Instead of storing the full tree in memory, which would spend too many computational resources, an incidence vector for a combination C, $y(C) \in \{0,1\}^m$, is used, such that, $y_e(C) = 1$ if link $l_e \in E$ is in C and $y_e(C) = 0$, otherwise. Henceforth, traversing the tree means systematically changing the value of $y(C)$. Figure 4.3 also shows the combinations' binary codes on all vertices.

The number of child nodes of a given combination C is the index of the rightmost active bit of its binary representation, denoted by $L(C)$. So, if $y_e^*(C)$ is the rightmost active bit, then $L(C) = e$ and there are e 0s to the right of $y_e^*(C)$ and C has e children. Conventionally, the indexation starts in 0 which is the rightmost bit. The only exception is the empty combination that, although there is no active bits, it has actually m children.

As a final abstraction, consider the decimal representation of $y(C)$, denoted by $d(C)$. If $y_e^*(C)$ is C's rightmost active bit, $L(C) = e$ can be calculated in $O(m)$ doing $L(C) = log_2(d(C) \wedge \neg(d(C) - 1))$. Moreover, it is possible to move from C to one of its child combination C_j assigning $y_j^*(C) = 1$. All C's children are reached doing $d(C_j) = d(C) + 2^j, 0 \leq j \leq e$. Again, if $C = \emptyset$, then it is a special case in which $0 \leq j \leq m - 1$.

Algorithm 1 covers the base case which is when $C = \emptyset$. Notice that the data structures used in the algorithm receive the same name as their respective theoretical abstractions. The only exception is $\mathcal{F}$ that is now a set of integers representing combinations of links. This is a simple entry point algorithm that basically assigns both the current combination C and the set of feasible sets $\mathcal{F}$ to $\emptyset$. It also calls the real enumeration algorithm for C's children and, when it is done, returns $\mathcal{F}$.

Algorithm 1 Enumeration Entry Point Algorithm

```
1: procedure ENTRY(E)
2:     d(C) ← 0
3:     C ← ∅
4:     F ← ∅
5:     for i = 0 : m − 1 do
6:         Enum(E, C, d(C) + 2^i, F)
7:     return F
```

Algorithm 2 performs the actual enumeration. It makes recursive calls until it reaches a leaf then returns to a parent call. First, in line 2, it calculates $L(C)$ for the current combination C. Then, adds link $l_{L(C)}$ to C and checks, in line 4, if the new C is feasible. If so, $d(C)$ is added to $\mathcal{F}$ and the enumeration algorithm is applied to C's children. Else, the whole branch of descendants of C is pruned by just not applying the algorithm to its children. Finally, the link added in the current recursive loop is removed from C and the

algorithm returns to the outer loop until it comes back to the entry point. This recursive design allows an implementation in which the SINR calculations for a certain C can be reused by its children, reducing the time to verifying whether C's descendants are feasible or not. Besides, notice that the running time is related to the number of visited vertices in the tree of combinations. If G has m links, then, in the worst case, 2^m recursive calls and feasibility verifications are performed.

Algorithm 2 Enumeration Algorithm

```
1: procedure ENUM(E, C, d(C), F)
2:     L(C) ← log_2(d(C) ∧ ¬(d(C) − 1)
3:     C ← C ∪ {l_{L(C)}}, s.t. l_{L(C)} ∈ E
4:     if IsFeasible(C) then
5:         F ← F ∪ {d}
6:         for i = 0 to L(C) − 1 do
7:             Enum(E, C, d(C) + 2^i, F)
8:     C ← C\{l_{L(C)}}, s.t. l_{L(C)} ∈ E
```

Figure 4.4 shows a flowchart for the execution of the enumeration algorithm applied to a network with $\mathcal{F} = \{1, 2, 3, 4, 5, 8, 9, 12, 13\}$. The tree root is the entry point algorithm and a vertex's height indicates its level in the algorithm's recursive stack. In other words, moving to a child vertex means starting an inner loop by adding a new instance of the algorithm to the top of the recursive stack. Conversely, returning to a parent vertex means resuming the outer loop by removing the last algorithm's instance from the top of the recursive stack. For a network with m links, the number of nested recursions is, at most, m. That being said, the downward arrows represent the algorithm calls leading to inner loops. The upward arrows indicates that all children were already visited (or pruned) so the algorithm will return to the outer loop. The numbers in the arrows represent the order in which the calling and returning events take place. All vertices with solid border lines are visited whereas the dashed ones are pruned. For this specific network, a pure brute force enumeration would take 15 recursive calls while this algorithm takes only 12. This difference tends to increase with the network complexity.

In summary, although its theoretical time complexity is still $O(n2^m)$, this algorithm performs better in practice for two reasons. First, because it uses the hereditary infeasibility to reduce the number of verifications and, second, because the SINR calculations are reused in inner recursive loops. Therefore, it can be used to enumerate all feasible sets faster than its theoretical time complexity in practice, allowing the mathematical formulation being applicable to some specific real scenarios, as discussed in Chapter 5.

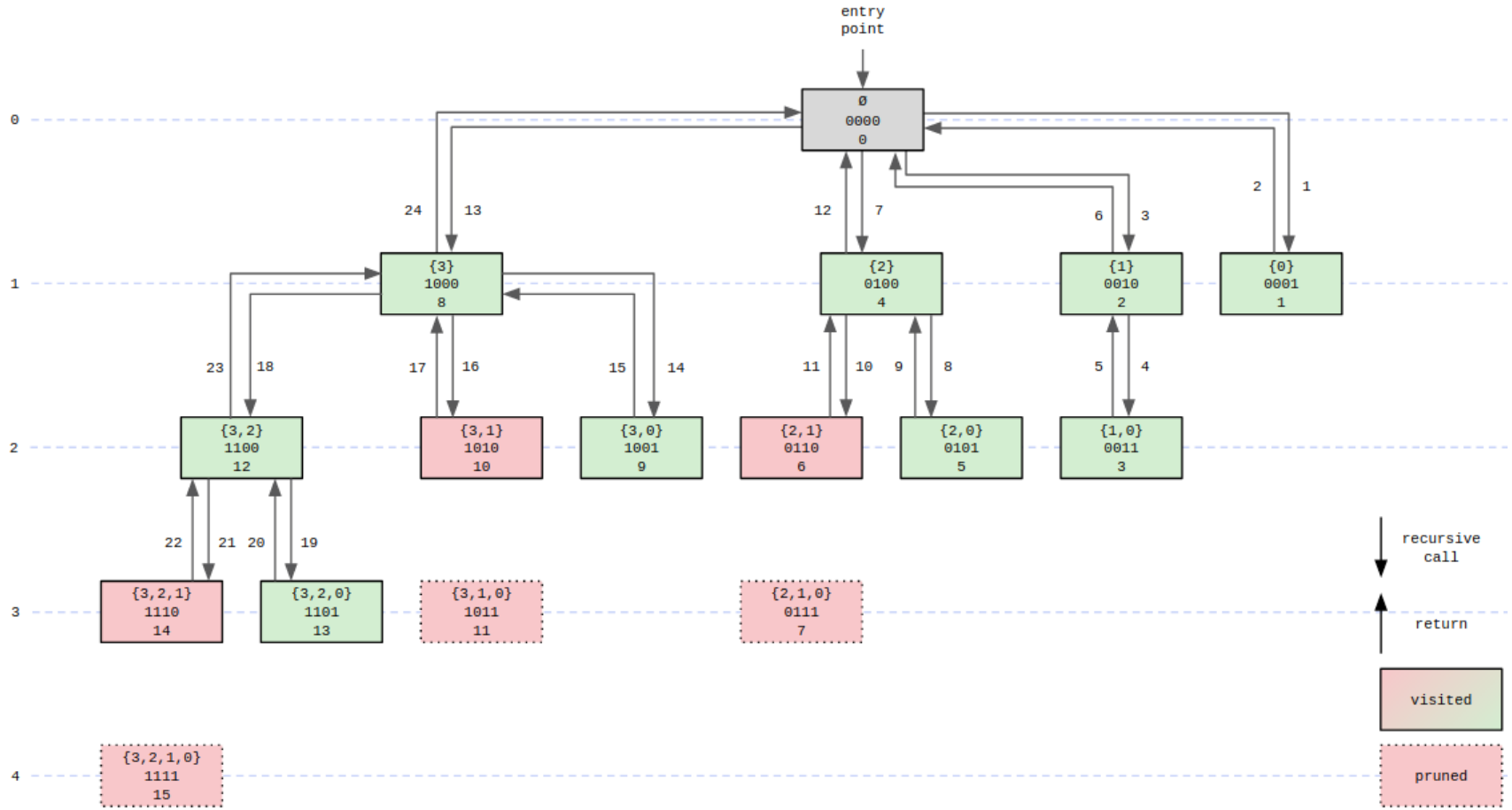


Figure 4.4: Enumeration algorithm execution flowchart.

4.6 Model Application Example

Figure 4.5 presents an arbitrary WMN with $\mathcal{N} = \{A, B, C, D, S_0, S_1\}$ and $\mathcal{L} = \{0, 1, 2, 3\}$. Regular sensor nodes are the blue circles, sink nodes are the blue hexagons, and links connecting two neighboring nodes are the solid green arrows. The idea is that sensor nodes are monitoring the environment and somehow want to constantly transmit collected data to either sink nodes. In this example, node B works as a bridge and helps node D to reach S_1 through a multi-hop transmission. A red arrow represents an interference signal generated from a sender to a non-neighboring receiver. The network in the figure can be thought of as a graphical representation of all transmissions demanded by a specific application for a fixed period of time. For example, node A wants to transmit data to S_0 while B wants to transmit to S_1. Because they are too close, B's residual signal reaches S_0, compromising its decoding ability.

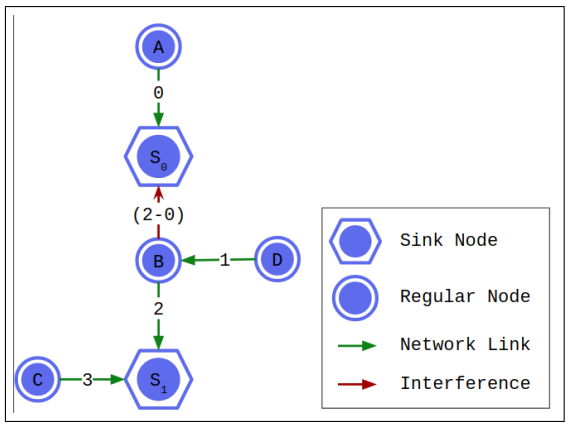


Figure 4.5: Example Network

In terms of the PIM constraints, links $\{1, 2\}$ can not be active at the same time because this pair violates the primary constraint as well as links $\{2, 3\}$. The secondary constraint is violated by links $\{0, 2\}$ because B is the sender of 2 and its residual signal interferes with S_0, the receiver of 0. That being said, we can run the enumeration algorithm following the flow chart on Figure 4.6 and the resulting set of feasible sets of links is $\mathcal{F} = \{\{0\}, \{1\}, \{2\}, \{3\}, \{0, 1\}, \{0, 3\}, \{1, 3\}\}$.

Now, all feasible sets of links are known and we can build the optimization model for this example. There are $|\mathcal{F}| = 7$ variables and $m = 4$ constraints. Equation 4.6 presents the resulting model. We intentionally omitted the variables domain because we want to

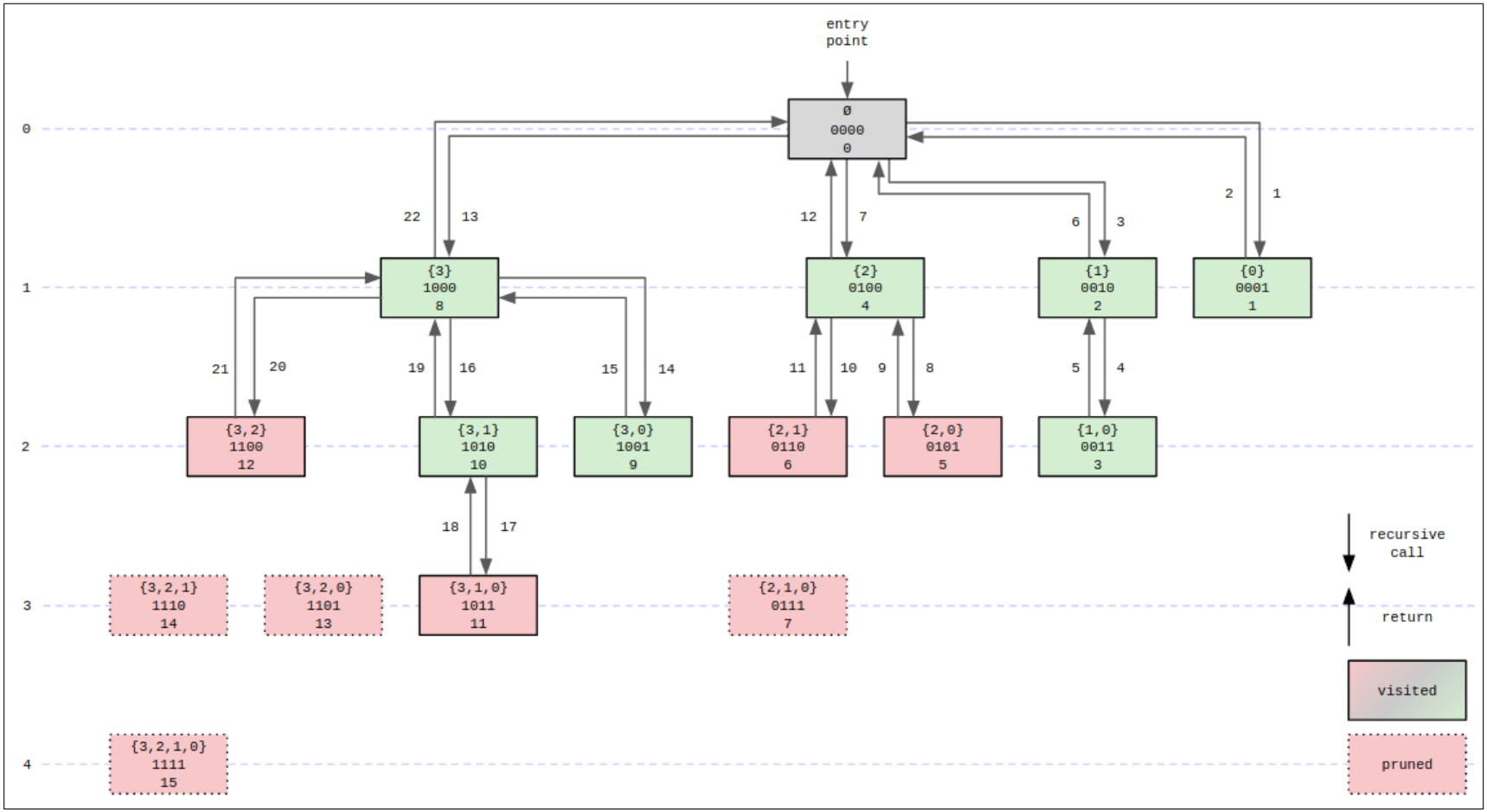
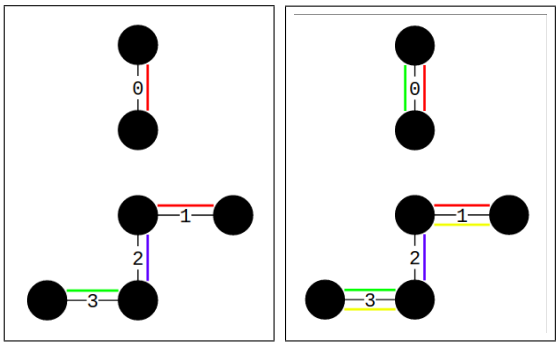


Figure 4.6: Enumeration algorithm execution flowchart for the example network

explore the LP and IP versions and compare both optimal schedulings.

$$
\begin{array}{rrrrrrrrr}
\min x_0 & +x_1 & +x_2 & +x_3 & +x_{0,1} & +x_{0,3} & +x_{1,3} & & (4.6) \\
s.t.\ x_0 & & & & +x_{0,1} & +x_{0,3} & & = 1 & \\
& x_1 & & & +x_{0,1} & & +x_{1,3} & = 1 & \\
& & x_2 & & & & & = 1 & \\
& & & x_3 & & +x_{0,3} & +x_{1,3} & = 1 & \\
x_0, & x_1, & x_2, & x_3, & x_{0,1}, & x_{0,3}, & x_{1,3} & \geq 0 &
\end{array}
$$

If $x \in \mathbb{R}^7$, then it is an LP model and the Simplex method can be applied. The optimal solution is $x_0 = x_1 = x_3 = 0$, $x_{0,1} = x_{0,3} = x_{1,3} = \frac{1}{2}$, and $x_2 = \frac{2}{2} = 1$, such that $\chi'_f = z^*_{LP} = \frac{5}{2}$. In this case, we need $L = 5$ colors to color each edge twice ($q = 2$). Notice that matchings $\{0, 1\}$, $\{0, 3\}$, and $\{1, 3\}$ meet half of their edges' coloring needs and each one represents one color. However, the link in matching $\{2\}$ needs to be colored twice and this demand is fully met by 2 repetitions of this very matching. So, each repetition will be a different color.

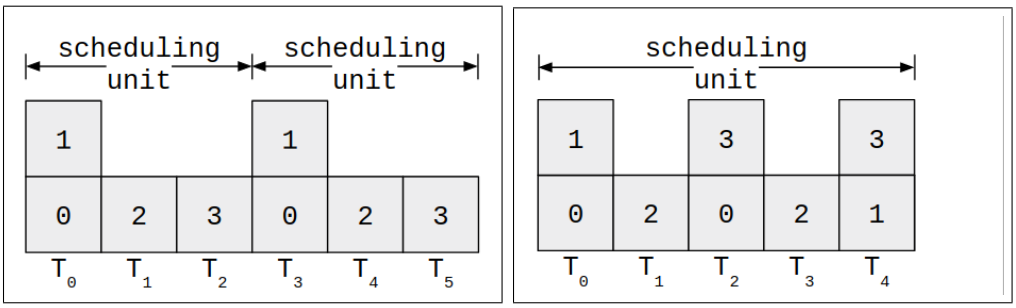


(a) (b)

Figure 4.7: The graph G edge coloring resulting from (a) IP and (b) LP models

On the other hand, if $x \in \mathbb{Z}^7$, this is an IP model and any Branch and Bound based technique can be applied. The optimal solution is $x_{0,1} = x_2 = x_3 = 1$ and all other variables are 0, such that $\chi' = z^*_{IP} = 3$; In this case, we need $L = 3$ colors and each edge is colored only once ($q = 1$). Matchings $\{0, 1\}$, $\{2\}$, and $\{3\}$ meet the full coloring demands of their edges so each one represents exactly one color. Using the network's graph representation, Figure 4.7 shows the colorings resulting from the models' solutions.

The IP model results in a TC while the LP model results in an SM because $z^*_{LP} < z^*_{IP}$. This also means that multicoloring the edges will produce a better scheduling than simply coloring them. In theory, we can prove it using the definition of capacity, which says that



(a) (b)

Figure 4.8: The resulting optimal link schedulings of (a) IP and (b) LP models

$C = \frac{q}{L}$. So, the capacity of the scheduling produced by the LP and the IP models are $C_{LP} = \frac{2}{5} = 0.4$ and $C_{IP} = \frac{1}{3} = 0.3$, respectively. Therefore, $C_{LP} > C_{IP}$ so the SM produced by the LP model is indeed better than the IP model's TC. This fact is reflected in Figure 4.8 which shows examples of different schedulings. Interestingly, in order to activate all links twice, the TC-based scheduling had to use two repetitions of its scheduling unit, which lead to a larger number of time slots.

Chapter 5

Experiments and Results

In Chapter 4, an LP formulation was presented as an exact approach to model and solve the Link Scheduling problem. To evaluate how this technique performs in practice, we simulate a proper experimental environment generating families of random networks. In this chapter, the process of generating random networks is described. Then, we provide some details on how the technique was computationally implemented. Finally, the experiments and their results are discussed.

5.1 Random Network Generation

A virtual random network is essentially a graph representation for a real wireless network. The architectural characteristics are controlled adjusting some parameters. Depending on which combination of values is used, different families of networks are produced. In a same family, networks may differ from each other by random factors. Once the parameters are defined, a deployment method must be chosen to effectively implement the network. The method proposed by VIEIRA *et al.* [19] produces more realistic networks and, for this reason, was used in this work.

The deployment method consists of two simple steps: choosing the nodes' location and defining the set of links. First, n nodes are randomly placed within a square area of side A. Then, if one node is located within the transmission range of another node, a wireless link is established with a random direction. This direction is important to assign nodes the roles of sender, receiver, or both. A random seed is used to better control the experiments allowing them being replicable. It works as a network id (or *netid* for short) that uniquely identifies a network within its family. According to the propagation model presented on Chapter 4, the transmission range is a function of the parameters α, β, γ, and TP. These parameters also determine how strong is the physical interference model for the generated network and consequently the number of feasible sets. Table 5.1

lists and gives a brief description of all parameters that somehow influence the network generation process.

Name	Value	Description
n	$\{10, 20, ..., 100\}$	Number of nodes
A	$\{1, 2, ..., 10\}$	Side of the square area (km)
$netid$	$\{1, 2, ..., 1000\}$	Random seed or network identifier
α	4.0	Log-distance propagation exponent
β	25.0	Antenna sensitivity (dB)
γ	-100.967	Noise floor (dBm)
TP	24.7712	Transmission Power (dBm)

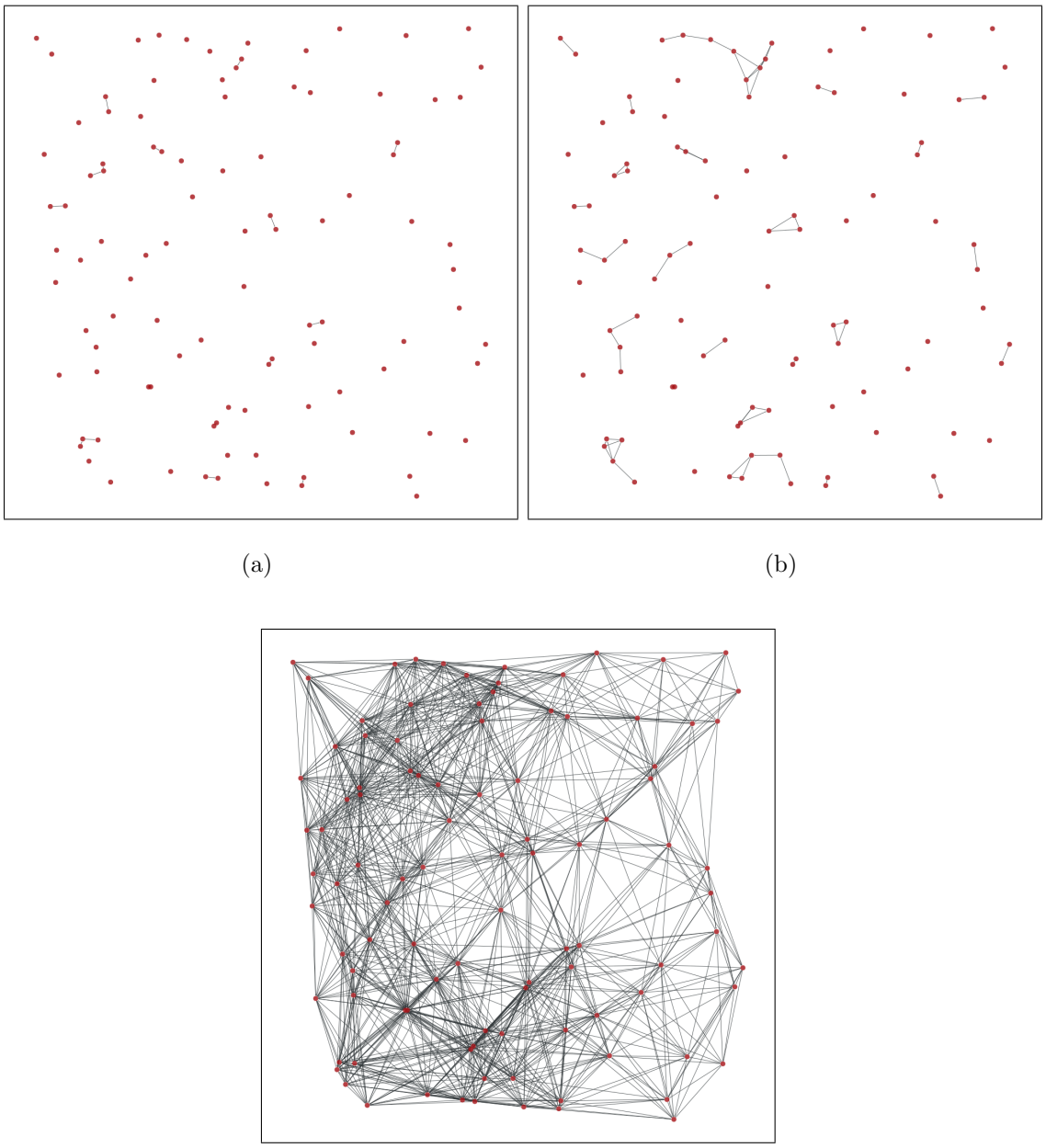
Table 5.1: Network Generation Default Values

In order to make the experiment simpler and more objective, this study only considered n, A, and $netid$ as varying parameters. The first two control the network general structure and the last one alters the random factors. Hence, in this work, the families of networks are characterized by the pair (A,n). The remaining parameters were kept fixed so all nodes share the same transmission range ($\rho = 329.954$m). This work's experimental setup followed the same procedure described by VIEIRA *et al.* [19] so it kept the same values for the fixed parameters.

Based on the fixed parameters and the resulting ρ, the variation intervals for the varying parameters were chosen such that more representative changes could be observed. For example, for this configuration and an arbitrary number of nodes, area sides larger than 10km result in networks with practically no links due to extremely long distances between nodes. On the other hand, area sides smaller than 1km would produce excessively dense networks where interference would be so strong that only single links could be considered as feasible sets. Therefore, exploring outside the interval $[1, 10]$ would offer no relevant experimental information. The variation interval for the varying parameters and the specific values for the fixed ones are also summarized on Table 5.1.

The resulting networks have structures following the examples on Figure 5.1. This figure shows three different examples of networks formed with the same number of nodes and random seed ($n = 100$, $netid = 1$) but with different area sides (Figure 5.1(a) $A = 10km$, Figure 5.1(b) $A = 5km$, and Figure 5.1(c) $A = 1km$). In order to improve visualization, edge directions were omitted as well as sender/receiver labels for the nodes.

Just confirming what was previously commented, it is important to observe the effect that different area sides have on the network structure. Basically, the smaller the area, the higher the links' density. From another perspective, although it is not possible to see directly on these pictures, a smaller area also means stronger interferences due to nodes' proximity. These two aspects concerning the area side have opposing influence on the number of feasible sets.



(a) (b)

(c)

Figure 5.1: Randomly generated networks examples

Given a random network generated by the previously described procedure, a series of experiments is performed on it. First, the LP and IP models are solved, providing the fractional and traditional chromatic indices, respectively. The time needed to enumerate the feasible sets and to find each solution is monitored. Once the solutions are obtained, it is possible to verify whether the network reaches the optimal scheduling through TC, WM, or SM. Finally, the chromatic index ratio and the network capacity are calculated.

5.2 Implementation Details

The random network was implemented as a class using the C++ programming language that can be included as a library. The class was developed using a graph data structure such that all interference information are kept and easily handled as a sort of meta data.

The enumeration algorithm was also implemented as a C++ class and receives a network object and an optimization model object in its constructor. As described in Chapter 4, it has two basic methods. In a few words, there is an entry point method that recursively calls the actual enumeration method. Every time a new feasible set is found, its respective variable is added to the objective function and to related constraints. Both objective function and constraints are members of the optimization model object.

Because the enumeration algorithm uses basic arithmetic manipulations to traverse the combination tree, we need unsigned integer variables to represent the combinations of links. In this abstraction, the variables must have m-bits, one for each link in the network. Most computer architectures use a 64-bit word so variables can not trespass this size. Although libraries with longer variables are found for most programming languages, they are implemented in higher levels. Therefore, they have a more complex structure and the operations are not performed on a CPU instruction level anymore, making the computing time considerably higher. However, the C language family has an extension that allows a 128-bit unsigned integer variable and provides some elementary operations on it. This extension performs slower than a 64-bit arithmetic system but faster than arbitrary-precision libraries, such as GMP. In order to consider the widest variety of networks as possible and not to take too long to perform the experiments, the C extension was chosen to implement the enumeration algorithm. Hence, the experiments were limited to networks with at most 128 links.

In this work, we first used GLPK library to implement the optimization model. It seemed to be a good option because it is open source and is widely used in numerous other research projects. However, some models demanded huge data structures and the GLPK's implementation is deficient providing them. This happens because GLPK's data structures are indexed using 2-complement 32-bits integers. As an alternative, we used Gurobi [20] with an academic trail license. It performed better than GLPK and dealt perfectly with larger problem instances. After evaluating the running time of all available solving methods, we forced Gurobi to use the Simplex method to solve both the LP model and the IP nodes' relaxations due to its overall better performance. Another project decision was disabling the presolve because, although it could solve the model faster for many simpler cases, it has an extremely high RAM usage peak that compromises its usage in more complex instances.

Although Gurobi is based on a fairly flexible and dynamic structure allowing the implementation of instances with practically any size, we still have to deal with limited hardware resources. All experiments run in a computer with 128GB of RAM so, with this amount of available memory, we could solve the model for instances with no more than 50 million of feasible sets.

In summary, during the network generation process, if a network has over 128 links, then it is dropped because it is not possible to use the enumeration algorithm. In the same way, during the enumeration process, if a network has over 50 million of feasible sets, it is dropped because it is not possible to solve the LP with the available computing resources. In addition, the combination of parameters can produce networks with no links at all, and this is considered a type of trivial limitation. Hence, if a network has zero links, it is also dropped.

5.3 Model Applicability

Although the formulation introduced in Chapter 4 can provide the optimal scheduling, it has two significant technical limitations. They arise from the necessity of enumerating all the feasible sets of links in order to find the formulation constraints and also from storing and processing a large amount of data to solve the model.

Considering the three situations summarized in the end of the previous section, it is important to determine for which kind of networks the scheduling model can be applied. Figures 5.2, 5.3, and 5.4 show how many networks were dropped due to absence of links, excessive number of links, and excessive number of feasible sets, respectively.

On Figure 5.2, it is possible to notice that the larger the area, the more networks are dropped due to the absence of links, for relatively small number of nodes. This happens because, for larger areas, generally the distance between nodes is longer than their transmission range, making links less likely to form.

The opposite happens on Figure 5.3, where networks with smaller areas have more links because nodes are closer to each other. When a valid number of links is produced inside an intermediate area side, the third situation emerges. In other words, when a considerable number of links does not produce enough interference to invalidate most of the feasible sets, then a large number of feasible sets are found and the related network must be dropped as well. This is depicted in Figure 5.3.

It is important to mention that, for each family of networks (A,n), a best effort attempt to find 1000 different networks was made. Because of those three dropping situations, some families have different number of samples. It was a project decision not forcing to find all 1000 networks for two reasons.

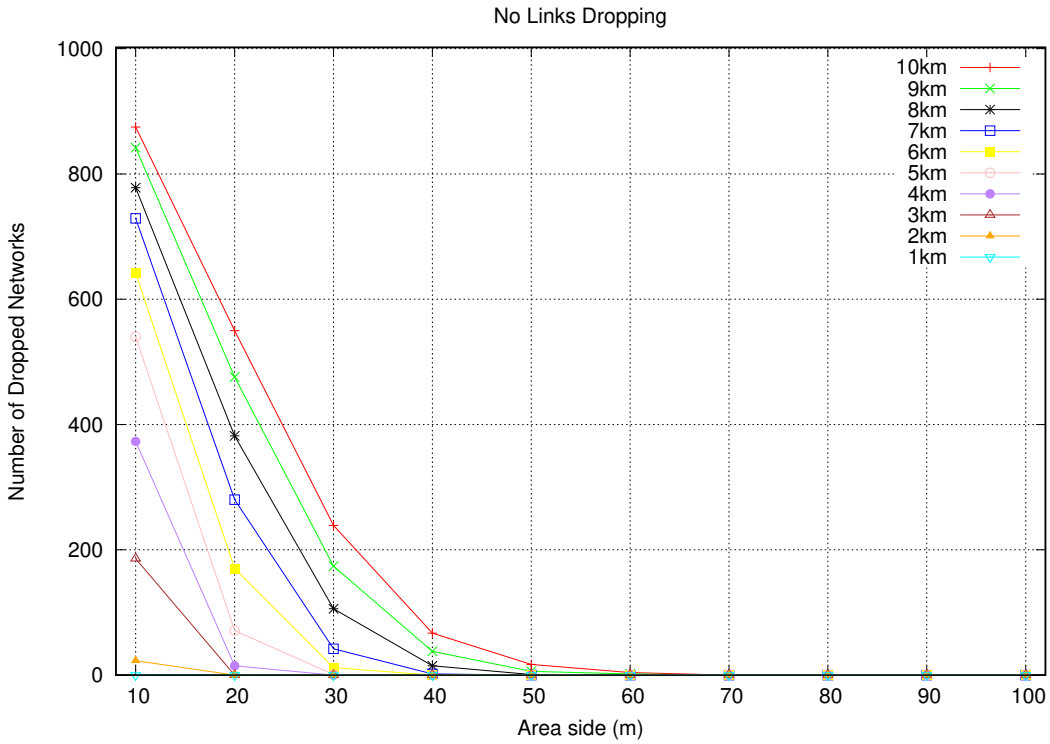


Figure 5.2: Number of dropped networks due to $m = 0$

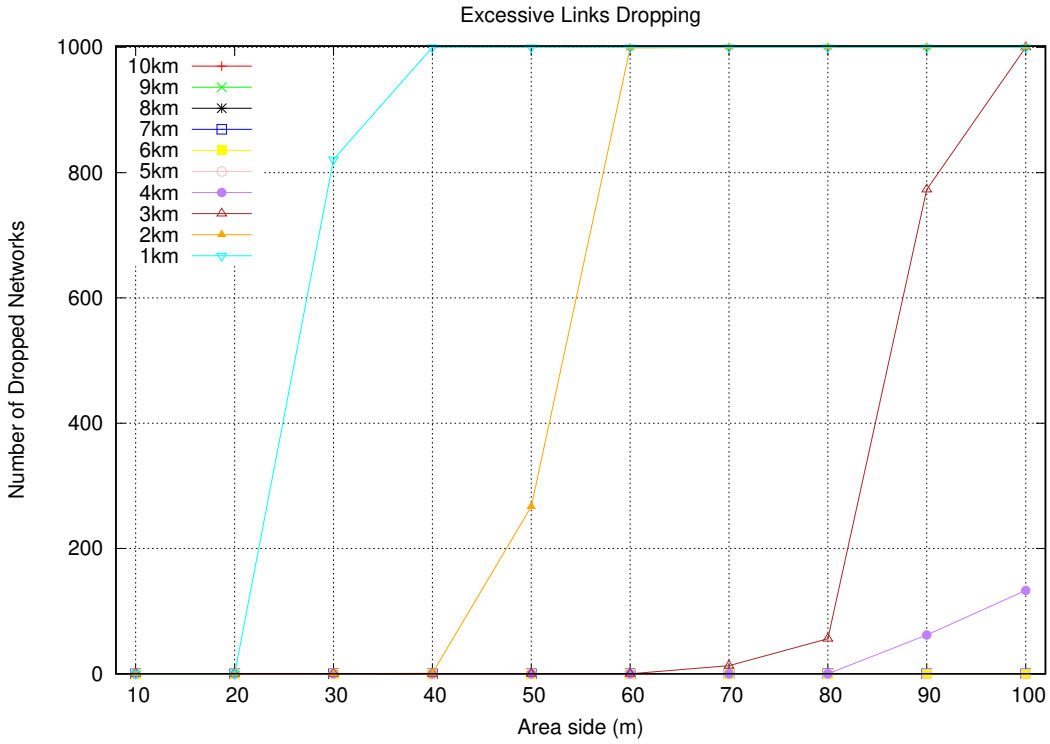


Figure 5.3: Number of dropped networks due to $m > 128$

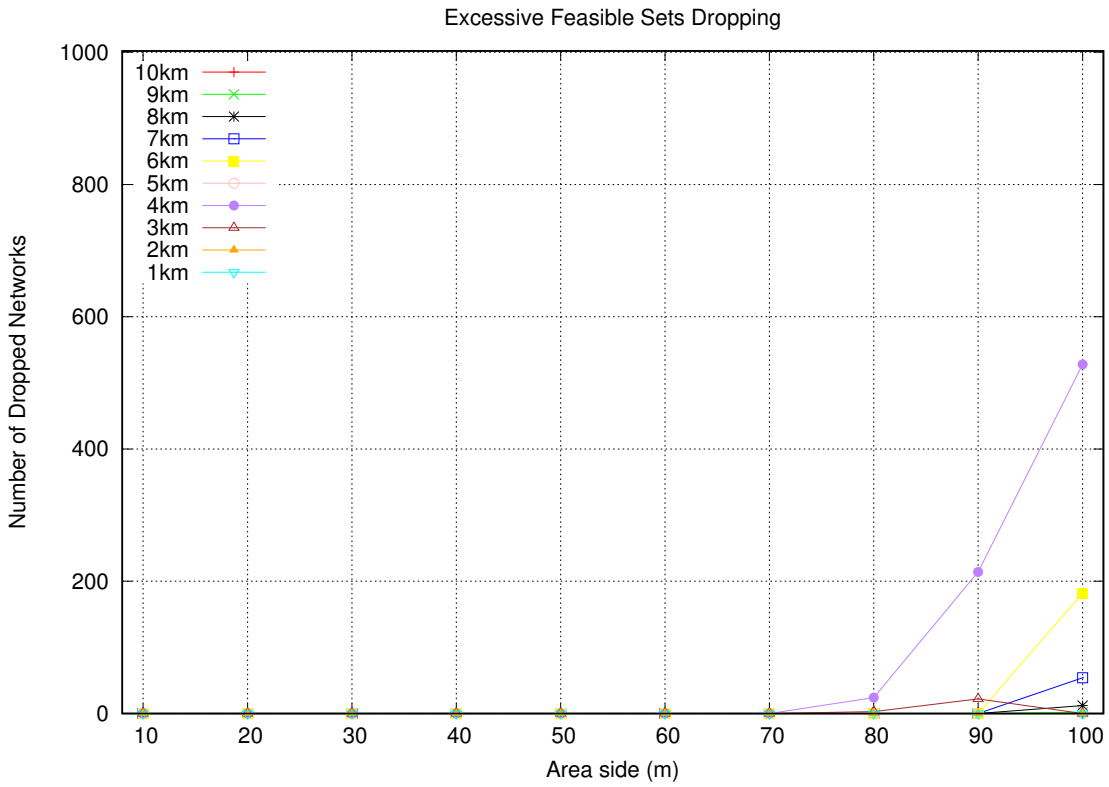


Figure 5.4: Number of dropped networks due to $|\mathcal{F}| > 50M$

First, for families with no link formation, if a parameter configuration demands too much dropping, the related family might not be a representative experimental data source and, thus, it would be a waste of time to insist on it. Second, the number of droppings is reflected on the confidence interval. If a certain configuration has only a few drops, this will result in a richer population and a smaller confidence interval, while families with smaller samples tend to have larger intervals and therefore not so reliable results. Table 5.2 shows exactly how many networks were considered for each parameter configuration.

nodes \ side	1km	2km	3km	4km	6km	7km	8km	9km	10km
10	1000	977	814	627	358	271	222	158	125
20	1000	1000	1000	985	830	720	618	524	450
30	178	1000	1000	1000	988	958	894	826	761
40	0	999	1000	1000	1000	998	985	962	933
50	0	733	1000	1000	1000	1000	999	994	983
60	0	1	1000	1000	1000	1000	1000	998	996
70	0	0	982	1000	1000	1000	1000	1000	1000
80	0	0	941	976	1000	1000	1000	1000	1000
90	0	0	205	724	1000	1000	1000	1000	1000
100	0	0	0	339	818	946	988	998	1000

Table 5.2: Number of networks for each parameter configuration

By analyzing the average of the feasible random networks for each family, it is possible to look to this issue from another perspective. As a first analysis, the enumeration feasibility can be assessed by observing the average number of links. Figure 5.5 shows this quantity as a function of the number of nodes for different area sides (represented by curves with different colors).

Again, each point on the chart represents the average number of links for a family of at most 1000 random networks. In this case, the confidence interval is really small for each point which means that the calculated average is actually a good representative of the whole population. Notice that there is an imaginary line on the mark of 128 links which means that only networks below it are allowed to be enumerated. Depending on the area side A, this line is reached sooner, for smaller areas, or later, for larger ones. The absence of links is a phenomenon that indeed only happens for larger areas with few nodes.

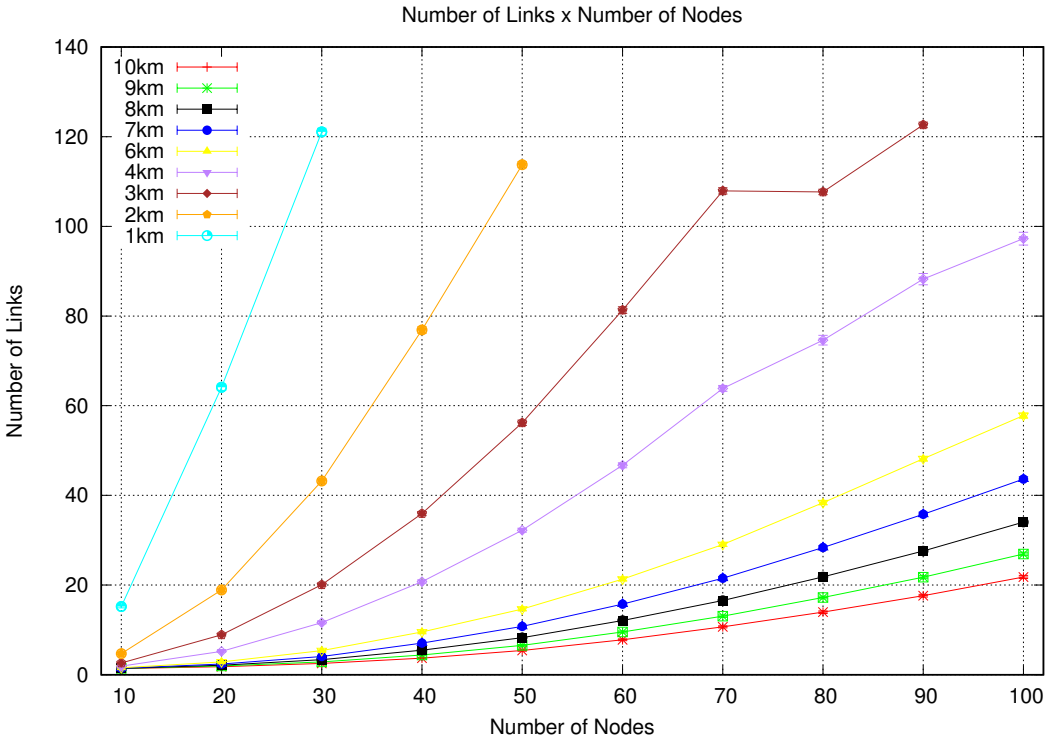


Figure 5.5: Average number of links

Similarly, the average number of feasible sets can help to determine for what kind of network the scheduling is actually doable. Figure 5.6 shows the log of the average number of feasible sets as a function of the number of nodes for different area sides. The curve for each area side is approximately a straight line parallel to each other which confirms the exponential growth. The threshold of 50 million feasible sets was used so those lines that approach the mark of 7 tend to have a higher dropping rate.

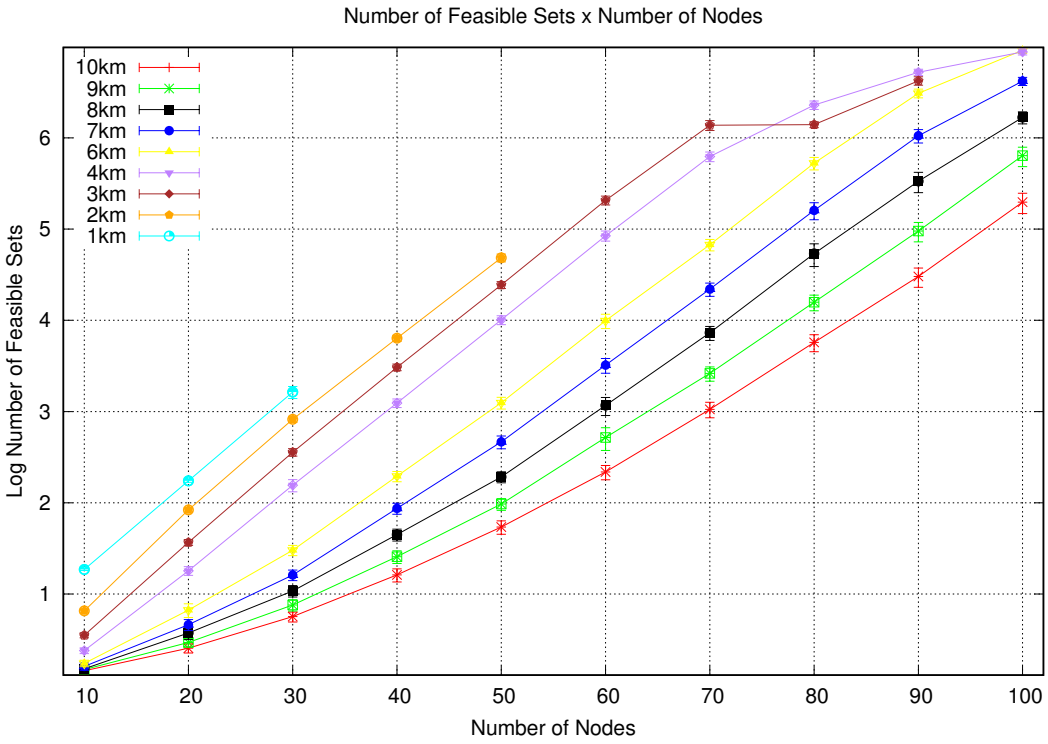


Figure 5.6: Average number of feasible sets

The same pattern is observed as before, the smallest areas trespass this mark sooner than the largest ones. For example, for $A < 6$km, this point is reached when networks have around 90 nodes and, for $A = 10$km, the same point is reached when networks have much more than 100 nodes. Thus Figures 5.5 and 5.6 explain the dropping rates and justify the successful use of the families of networks in the experiments.

The computational resource usage can be measured by the running time for the main system's activities. Figures 5.7, 5.8, and 5.9 show the average running time of the enumeration, LP solution, and IP solution activities, respectively. The average time is a function of the number of nodes for different values of area side (represented by the colored curves). For more complex networks with greater number of feasible sets, e.g. random networks $(3, 70)$, the average enumeration time to find around 10^6 feasible sets was less than 1 minute. Notice that the almost exponential growth of the average enumeration time follows the same behavior of the curve representing the number of feasible sets. This is coherent with what was exposed in theory when the enumeration algorithm was described. Essentially, the enumeration time complexity is exactly the number of visited nodes of the combination tree which can be fairly approximated by the number of feasible sets. Regardless of its potential to get exponentially high running times, the enumeration process is not a threat to the system because it performs relatively fast for the selected families of networks.

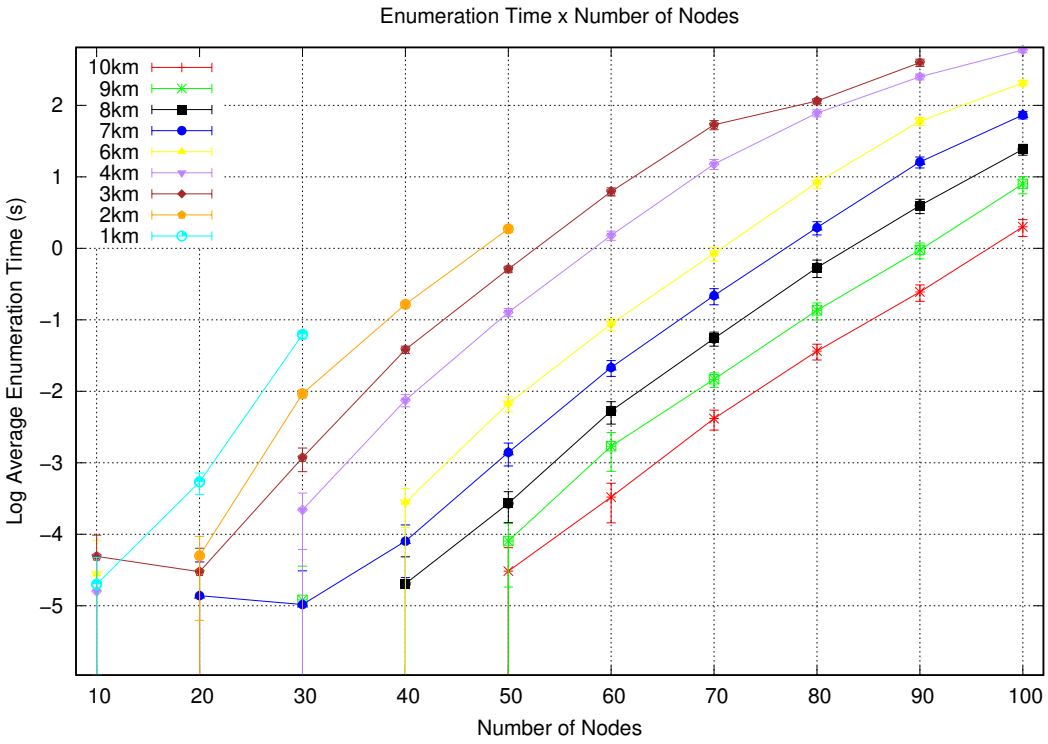


Figure 5.7: Average enumeration time

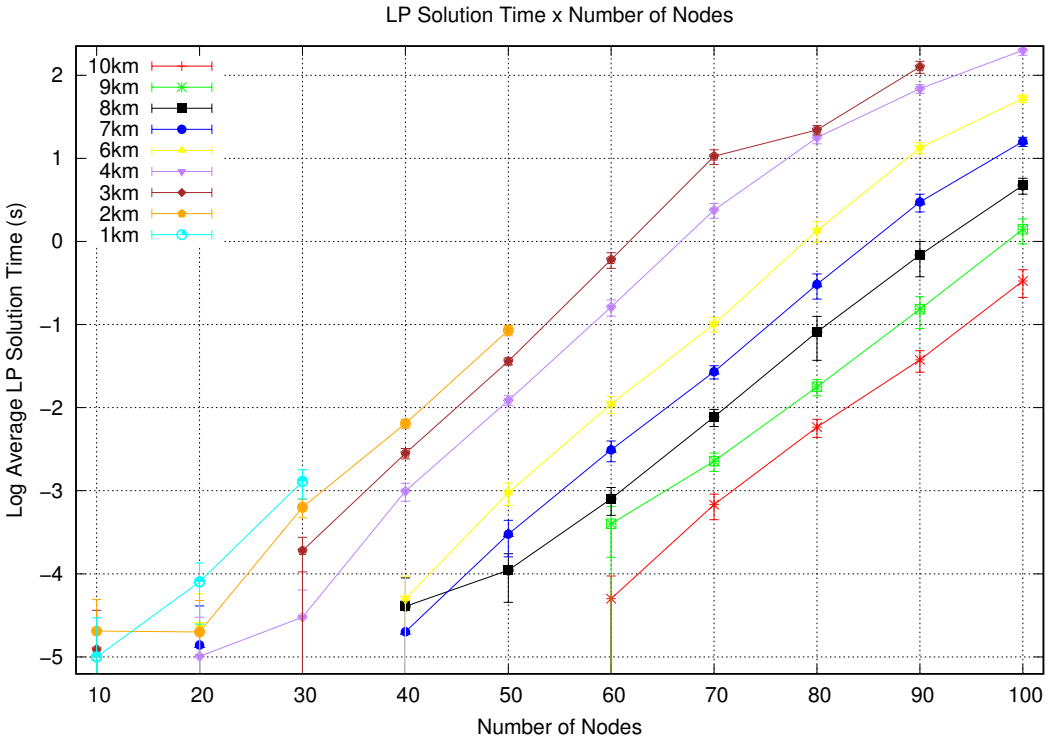


Figure 5.8: Average LP solution time

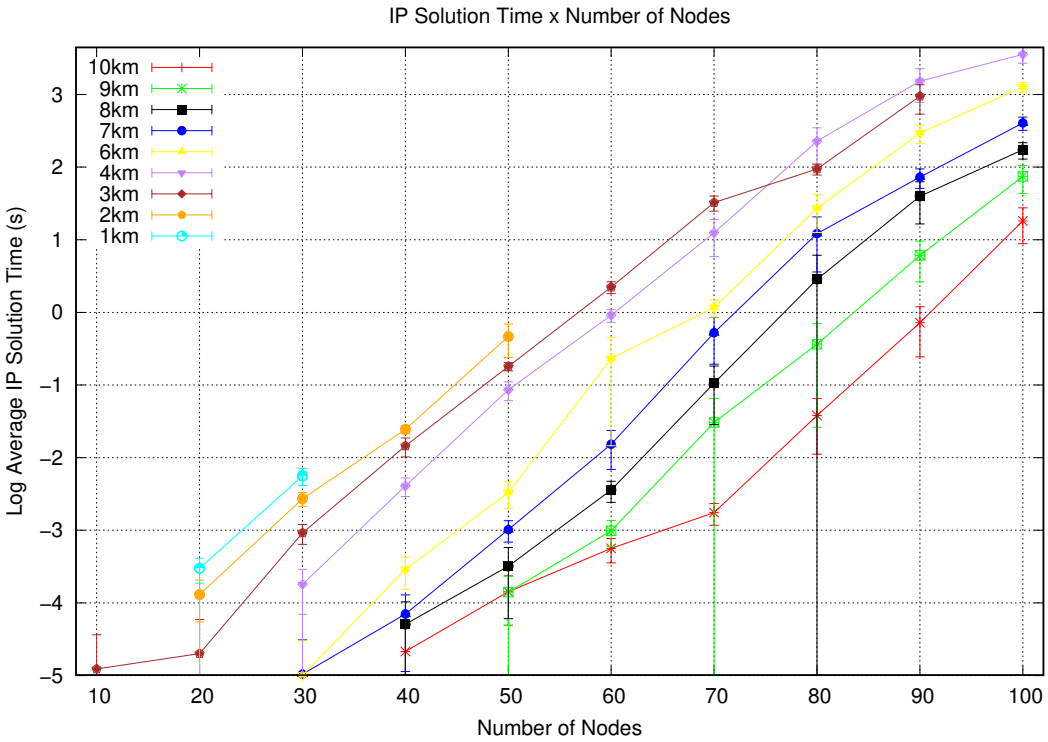


Figure 5.9: Average IP solution time

In the same fashion, Figures 5.8 and 5.9 show the computational demand in terms of running time to solve the LP and IP models, respectively. A general observation is that the time to solve an IP model is at least ten times greater than the time needed to solve an LP model, in average. There exist extreme cases in which the IP model takes hours to find a solution and this can invalidate its utilization in applications with real demands. Now that we know about its excessive computational expense, we are only going to use the IP model results to support our observations. Specifically for this work, solving the IP model helps to identify what case of optimal scheduling was assigned to a certain network, making clear what is the difference of performance between multicoloring and single-coloring based schedulings.

It is also interesting to highlight that solving the LP for the densest networks did not take much more than 10 minutes in average. In the end, the total time used to enumerate and solve the model was about 20 minutes long. Therefore, if an application demands the optimal scheduling of a fairly amount of wireless links under the physical interference model and is time tolerable, this technique is a good option.

5.4 Performance Analysis

From the theory, we know that IP models will result in TC-based schedulings while LP models can result in TC, WM, or SM-based schedulings. We also know that these models can have exponential size. Trying to work around this exponential complexity issue, most works in the literature propose approximative yet polynomial approaches that eventually also provide an optimal scheduling and usually are based on traditional vertex or edge-coloring.

As we observed in the previous section, our multicoloring-based technique can deal with this complexity issue, in practice, for the variety of tested networks. It takes advantage of the PIM characteristics to perform a faster enumeration and provide always the optimal scheduling. Therefore, it is pointless to evaluate the difference of performance of our proposed technique and other well-known heuristics. This comparison could be used to evaluate how far the heuristics' results are from the optimal. However, it is interesting to analyze how better are the schedulings provided by multicolorings and traditional colorings.

In our first analysis, we want to see how many cases of TC, WM, and SM were produced in our experiments. Figures 5.10 and 5.11 present two different charts with the percentage of networks scheduled with WM and SM, respectively. This percentage is related to the number of samples for each family of networks.

The denser are the networks, the more instances of WM and SM are detected. Practically all samples of smaller areas in Figure 5.10 use multiple colors on their edges although their performances are as good as a TC. More importantly, Figure 5.11 suggests that the number of SMs tends to increase with the graph density and the current results show a maximum of 39% for the area of 3km. Therefore a significant amount of networks are scheduled using multicoloring and a smaller yet representative portion of these networks have better schedulings than any TC based approach.

Now, let's take a closer look to all networks scheduled with SM. We want to demonstrate the SM-based scheduling superiority making a comparison between its resulting network capacity and the capacity of a scheduling based on TC. In order to find a TC for a network, we simply solved the IP model. Figures 5.12 and 5.13 plot the average of the inverse of the optimal objective function versus the number of nodes for each different area side. Both charts have similar behavior which in a way confirms the subtle theoretical difference between fractional and traditional chromatic indices. This result also matches GUPTA e KUMAR [18]'s argument that the network capacity decreases with the increasing number of nodes.

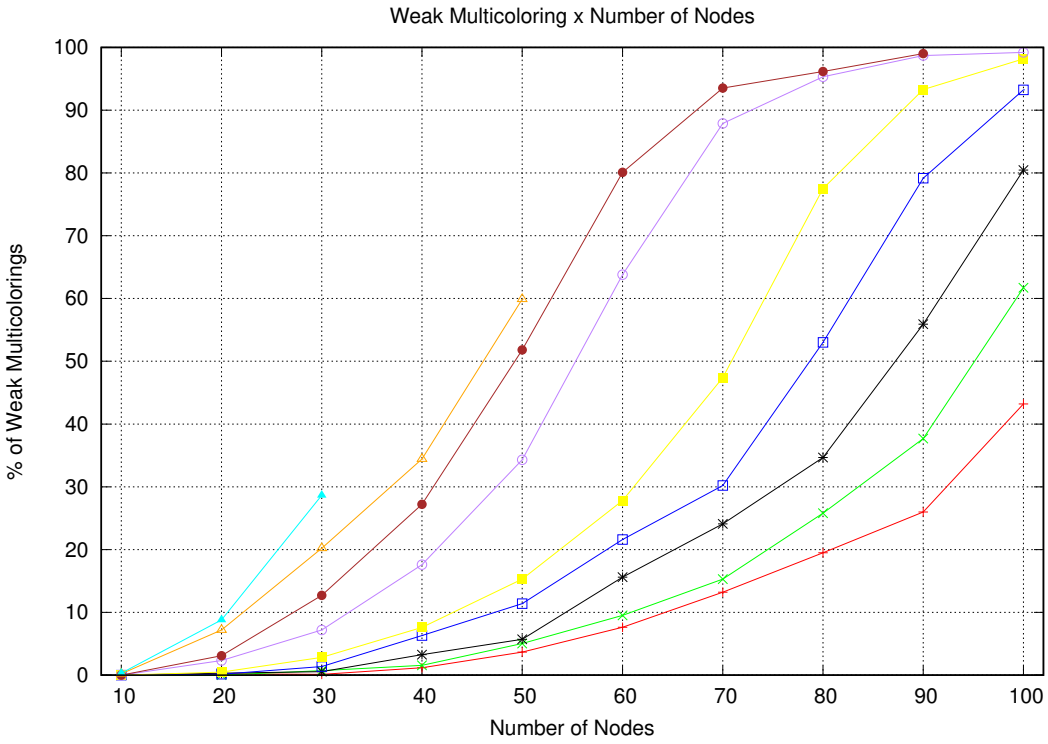


Figure 5.10: Number of WM schedulings

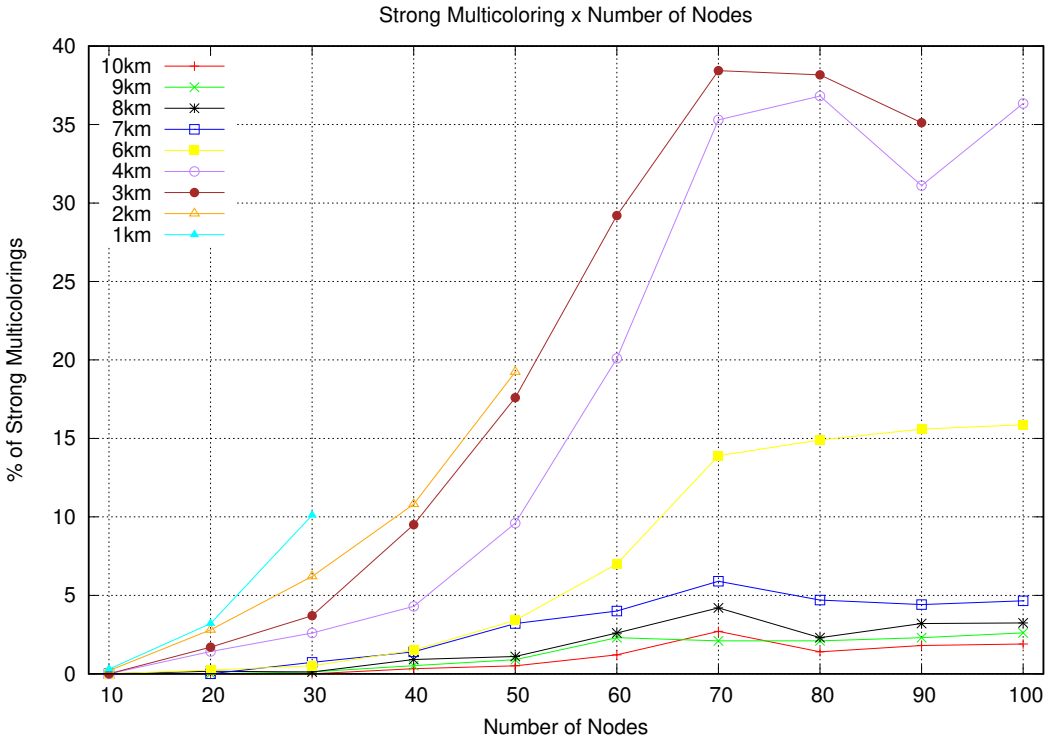


Figure 5.11: Number of SM schedulings

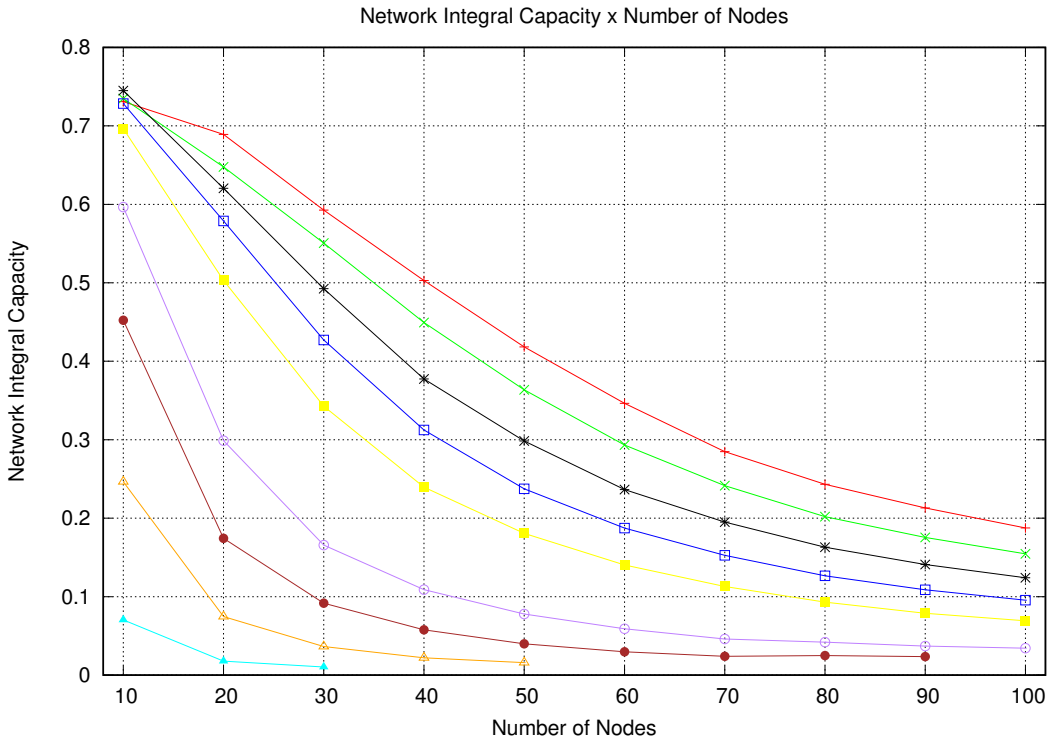


Figure 5.12: Network capacity for TC-based schedulings

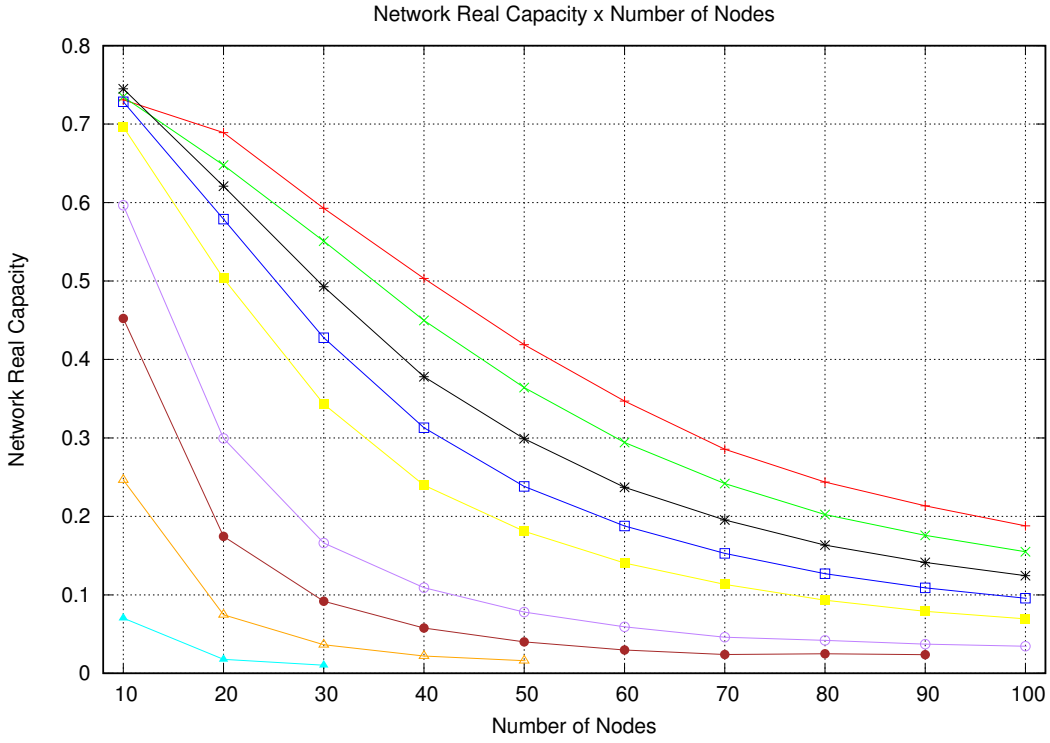


Figure 5.13: Network capacity for SM-based schedulings

Although we have an interesting result regarding the network capacity, it is still unclear how SM and TC performances differ from each other. To perform a numerical evaluation on this subject, we explicitly calculated the ratio between the average capacity and used it as a performance measure. Figure 5.14 depicts the advantage of SM over TC, plotting the average ratio of capacities as a function of the number of nodes for different area sides. Here, simpler cases experience more advantages in multicoloring. The performance gain in using the SM is up to 30% over the traditional coloring, although this value is reduced to 4% for families of networks with more incidence of SM-based schedulings.

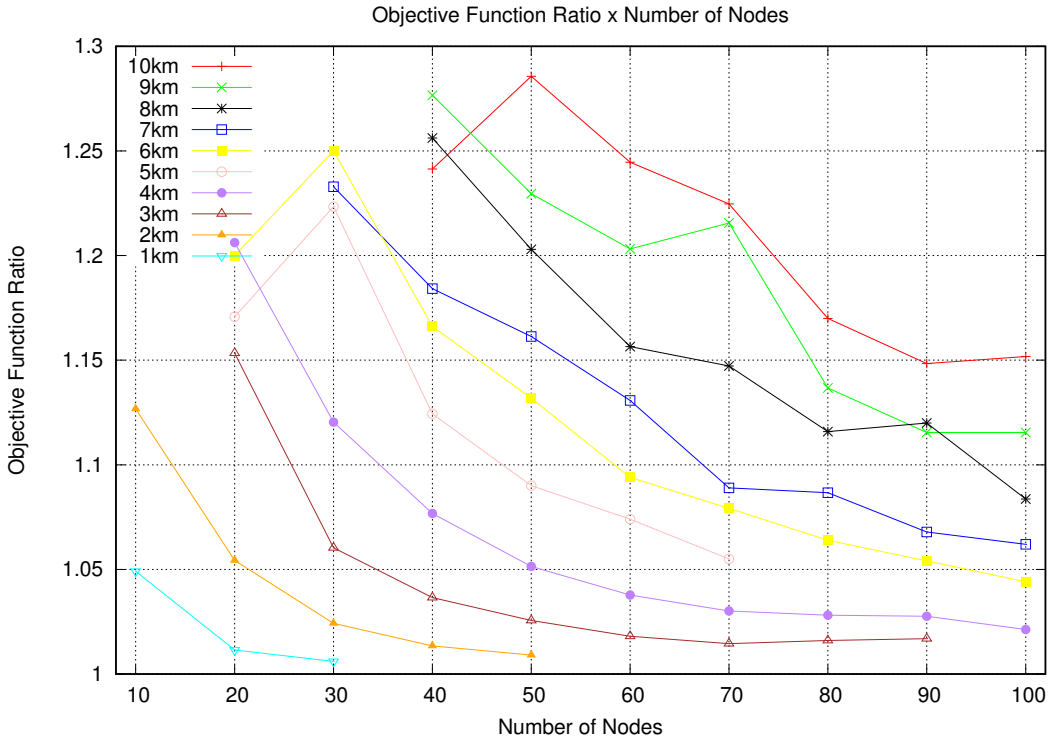


Figure 5.14: Number of Weak and Strong Multicoloring

Chapter 6

Conclusion

This work proposes an optimization model for the Link Scheduling Problem based on Fractional Edge-Coloring considering the Physical Interference Model. The Physical Interference Model is used to make the model more realistic. However, at the same time it makes the problem harder to solve, it provides some useful properties that help devising new solution approaches.

The Fractional Edge-Coloring problem was defined as an optimization problem. The definition was adapted to also model the Scheduling problem under the physical interference constraints. This adaptation is almost straightforward given the strong equivalence between these two problems. Unfortunately, a direct consequence of this adaptation is that most known solving techniques are no longer applicable.

Because we ran out of tools, we decided to tackle the problem by strictly building and solving the optimization model. In order to build those models, an exponential number of feasible sets of links need to be found. The hereditary unfeasibility property was used to support the development of a brute force enumeration algorithm with a better performance in practice, allowing us to apply this technique to a very selected family of networks.

We implemented the optimization model using the Gurobi solver on a resource limited computational environment. This technique was applied to a large number of different families of random networks respecting the system limitations. These families are characterized by parameters that control the transmission properties. After applying the model to the networks, we obtained some handy quantities used to assess the technique applicability and performance.

The results shown that the LP model is better than the IP, since it potentially offers shorter schedulings and can be solved faster. For the families of networks we used, many networks had to be dropped because the solution process demanded more resources than those available. However, for those that fit the limitations, the enumeration and solution

time was no higher than 20 minutes.

We noticed that the multicoloring scheduling occurrence is intimately related to the network link density. WM-based schedulings are found in 100% of the networks in some families while SM-based schedulings reach 40% in the densest ones. The most representative performance gains were observed in networks where the multicoloring is not a common result. This gain can represent at most 25% of improvement over TC-based schedulings. Finally, as expected, network capacities tend to decrease with the increasing number of nodes.

6.1 Final Remarks

We summarize the main contributions of the present work:

- The first contribution concerns the theory behind fractional coloring. This is a not so popular topic so the material was extracted from different sources and synthesized here. We also provide an interpretation for the optimization formulation components which helps its comprehension and possibly motivates its usage as a powerful modeling tool.

- A technique to find the optimal scheduling of PIM-based networks, consisting of solving an LP model. The modeling behind the technique is an important contribution because it illustrates how fractional coloring can be used to approach problems. Interestingly, this technique can handle the combinatorial issues well while exploring more realistic scenarios.

- The enumeration algorithm presented here, although applied to a very specific wireless networks scope, its usage is not limited to it. Any problem with combinatorial nature and hereditary infeasibility property can use this algorithm to search for all feasible sets of any objects and somehow support other problems' solutions and modelings.

6.2 Future Work

- It is possible to guarantee an application-driven QoS service by determining which links are more important than the others and answering how to produce schedulings that consider this information. Once this importance is defined by any ranking strategy, our scheduling technique can be used to model the problem such that specific links are scheduled more than others. The equality constraints can be

adapted to change each link's activation proportion, given a fixed transmission period.

- The formulation used here is only one of the many possible ways to model the Fractional Edge-Coloring problem. Moreover, it is possible that other formulations result in better solution approaches; either in terms of efficiency, or time complexity. We already have a simple alternative Combinatorial Optimization model for this problem but it needs to be improved and matured.

- We actually ran the experiments for many different values of the several parameters used. Although we did not notice significant differences, there is a huge amount of data describing these experiments. Proper data mining tools could be used to extract useful information from this data and create new intuition on the schedulings and general network structural-induced properties.

- It would be very interesting to observe what happens outside the computational limits we imposed to the experiments. We already have ideas on how to implement the system for networks with an arbitrary number of links. Furthermore, the project could be submitted to run on a supercomputer so the RAM limitation would also be removed.

Referências Bibliográficas

[1] KUNZ, T. "Carleton University Wireless mesh network". Disponível em: <`http://kunz-pc.sce.carleton.ca/mesh/newpage.htm`>.

[2] SHI, Y., HOU, Y. T., LIU, J., et al. "Bridging the Gap between Protocol and Physical Models for Wireless Networks", *IEEE TRANSACTIONS ON MOBILE COMPUTING*, v. 12, n. 7, pp. 1404–1416, July 2012.

[3] BJÖRKLUND, P., VÄRBRAND, P., YUAN, D. "A Column Generation Method for Spatial TDMA Scheduling in Ad Hoc Networks", *Elsevier Science*, 2003.

[4] GANDHAM, S., DAWANDE, M., PRAKASH, R. "Link scheduling in wireless sensor networks: Distributed edge-coloring revisited", *J. Parallel Distrib. Comput.*, v. 68, pp. 1122–1134, mar. 2008.

[5] DEZFOULI, B., RADI, M., WHITEHOUSE, K., et al. "DICSA: Distributed and concurrent link scheduling algorithm for data gathering in wireless sensor networks", *Ad Hoc Networks*, v. 25, pp. 54–71, set. 2015.

[6] HAJEK, B., SASAKI, G. "Link Scheduling in Polynomial Time", *IEEE Transactions on Information Theory*, v. 34, n. 5.

[7] FANG, Z., BENSAOU, B. "Fair Bandwidth Sharing Algorithms based on Game Theory Frameworks for Wireless Ad-hoc Networks", *IEEE INFOCOM*, mar. 2004.

[8] WAN, P.-J., FRIEDER, O., JIA, X., et al. "Wireless Link Scheduling under Physical Interference Model", *IEEE INFOCOM*, mar. 2011.

[9] LEE, J. *A First Course in Linear Optimization.* Reex Press, 2016.

[10] WOLSEY, L. A. *Integer Programming.* Wiley-Interscience, 1998.

[11] VIZING, V. G. "On an estimate of the chromatic class of a p-graph", *Diskret. Analiz.*, v. 3, pp. 25–30, 1964.

[12] SCHRIJVER, A. *Combinatorial Optimization*, v. B. 24 ed. Heidelberg, Springer-Verlag, 2003.

[13] SCHEINERMAN, E. R., ULLMAN, D. H. *Fractional Graph Theory*. John Wiley Sons, 2008.

[14] PADBERG, M. W., RAO, M. R. "Odd minimum cut-sets and b-matchings", *Mathematics of Operations Research*, v. 7, n. 1, pp. 67–80, 1982.

[15] EDMONDS, J. "Maximum Matching and a Polyhedron With O,1-Vertices", *JOURNAL OF RESEARCH of the Na tional Bureau of Standards-B. Mathematics and Mathematical Physics*, v. 69B, n. 1,2, pp. 125–130, dez. 1964.

[16] GRÖTSCHEL, M., LOVÁSZ, L., SCHRIJVER, A. *Geometric Algorithms and Combinatorial Optimization*. Berlin, Springer, 1988.

[17] RAPPAPORT, T. S. *Wireless communications principles and practices*. Prentice-Hall, 2002.

[18] GUPTA, P., KUMAR, P. "The capacity of wireless networks", *IEEE Trans. Inf. Theory*, v. 46, pp. 388–404, 2000.

[19] VIEIRA, F. R. J., DE REZENDE, J. F., BARBOSA, V. C. "Scheduling wireless links by vertex multicoloring in the physical interference model", *Computer Networks*, v. 99, pp. 125–133, 2016.

[20] GUROBI OPTIMIZATION, I. "Gurobi Optimizer Reference Manual". 2016. Disponível em: <`http://www.gurobi.com`>.