



SAMUEL TERRA VIEIRA

OTIMIZAÇÃO DO POSICIONAMENTO DE PONTOS DE ACESSO *WIRELESS*

FORMIGA - MG

2017

SAMUEL TERRA VIEIRA

OTIMIZAÇÃO DO POSICIONAMENTO DE PONTOS DE ACESSO WIRELESS

Monografia apresentada ao Curso de Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Área de concentração: Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - Campus Formiga
Ciência da Computação

Orientador: Prof. Msc. Everthon Valadão dos Santos

FORMIGA - MG

2017

Nascimento, Eduardo Simões.

5.13 Modelagem e Protótipo de um Sistema para Gerenciamento de
N353m Reuniões / Eduardo Simões Nascimento. -- Formiga: IFMG, 2017.
101p. : il.

Orientador: Prof^a. Dr^a. Paloma Maira de Oliveira

Trabalho de Conclusão de Curso – Instituto Federal de Educação,
Ciência e Tecnologia de Minas Gerais – Campus Formiga.

- 
1. Reuniões – Ensino – Ata.
 2. Laravel.
 3. PHP.
 4. Bootstrap
- I. Título

CDD 005.1



Ficha catalográfica elaborada pela Bibliotecária Ms. Naliana Dias Leandro CRB6-1347

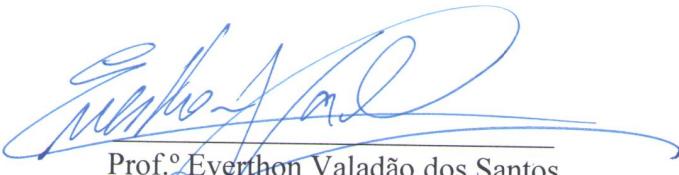
SAMUEL TERRA VIEIRA

**OTIMIZAÇÃO DO POSICIONAMENTO DE
PONTOS DE ACESSO WIRELESS**

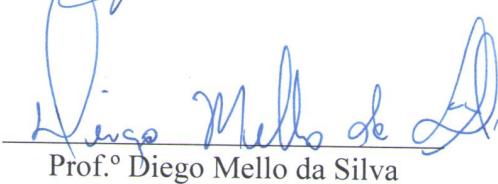
Trabalho de Conclusão de Curso apresentado ao
Instituto Federal de Minas Gerais - Campus
Formiga, como requisito parcial para obtenção do
título de Bacharel em Ciência da Computação.

Aprovado em: 14 de novembro de 2017.

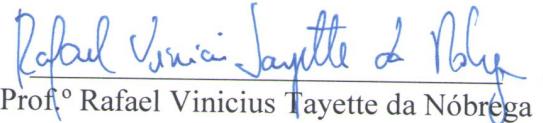
BANCA EXAMINADORA



Prof.º Everthon Valadão dos Santos



Prof.º Diego Mello da Silva



Prof.º Rafael Vinicius Tayette da Nóbrega



Prof.º Wallace de Almeida Rodrigues

Agradecimentos

Agradeço a Deus, pois sem Sua ajuda, direção e o Seu agir em minha vida, não teria capacidade e o empenho para chegar até aqui; por se fazer presente até nos momentos em que os desafios foram tão grandes quanto minha vontade, por me ter dotado de saúde, sabedoria e disposição para alcançar mais uma etapa em minha vida.

Agradeço aos meus pais que, com toda humildade e simplicidade, me ensinaram a ser uma pessoa decente, a respeitar e buscar meus sonhos de forma honesta e dentro do meu tempo, mesmo que seja com muito trabalho árduo.

Agradeço ao responsável pela parte de TI do campus, Roger Ferreira, e ao engenheiro civil do campus, Alysson Geraldo, por terem gentilmente cedido as plantas-baixas. Agradeço ao professor Everthon Valadão, a paciência e compreensão que teve comigo durante o período em que me acompanhou e que estivemos juntos realizando este trabalho. Esteve sempre presente como um grande professor e amigo, sempre se mostrando comprometido em dar o seu melhor para os alunos.

“Se vi mais longe foi por estar de pé sobre ombros de gigantes.”

Isaac Newton

Resumo

A proposta deste trabalho é desenvolver um *software* para *Wireless AP Placement* que utilizará modelo(s) da propagação do sinal *Wi-Fi* de acordo com as características físicas do ambiente, aplicando o *Simulated Annealing* como metaheurística para visar ao aprimoramento da cobertura de sinal, tendo como caso de uso as dependências do IFMG *campus* Formiga. Tal *software* fornece um melhor posicionamento dos *access points Wi-Fi* para o ambiente analisado, bem como informa o percentual de cobertura de sinal para tal quantidade de *access points*. O trabalho realizado foi disponibilizado de forma gratuita utilizando licença pública geral GNU GPL. Ressaltamos que ele possibilita testar disposições de *access points* sem o custo operacional de fisicamente movê-los, de maneira a propor uma disposição espacial dos mesmos que forneça uma maior cobertura e intensidade de sinal dentro do ambiente simulado. Para trabalhos futuros, a fim de se obter resultados que possibilitem uma melhor busca pelos pontos de acesso para dois ou mais *access points*, novas técnicas poderiam ser utilizadas para o cálculo da função objetivo da metaheurística, ou até mesmo implementar outras metaheurísticas.

Palavras-chave: *Wireless*. Propagação. Posicionamento. *Wi-Fi*. *Simulated Annealing*. CUDA.

Abstract

This work aims to develop a software for Wireless AP Placement that will use Wi-Fi signal propagation model (s) according to the physical characteristics of the environment, applying the Simulated Annealing as a metaheuristic for the improvement of signal coverage, taking as a case of use the dependencies of the IFMG - Formiga campus. Such software provides a better positioning of the Wi-Fi access points for the analyzed environment and informs the percentage of signal coverage for such amount of APs. The work was made available free of charge using the GNU GPL general public license. We emphasize that it makes it possible to test dispositions of APs without the operational cost of physically moving them, in order to propose a spatial arrangement of them that provides a greater coverage and signal intensity within the simulated environment. For future work, in order to obtain results that allow a better search for access points for two or more APs, new techniques could be used to calculate the objective function of metaheuristics, or even implement another metaheuristics.

Keywords: Wireless. Propagation. Positioning. Wi-Fi. Simulated Annealing. CUDA.

Listas de ilustrações

Figura 1 – Comportamento do <i>SimulatedAnnealing</i> durante a exploração do espaço de busca.	44
Figura 2 – Representação do ambiente a partir do arquivo <i>DXF</i> contendo a planta-baixa.	55
Figura 3 – Medição da intensidade de sinal <i>vs.</i> modelos de sua propagação.	57
Figura 4 – Ajuste de curvas da 2P-Logística, 3P-LogNormal e 3P-LogLogística.	59
Figura 5 – Qualidade do ajuste (GOF) da distribuição 3P-LogLogística.	60
Figura 6 – Qualidade do ajuste (GOF) da distribuição 3P-LogNormal.	61
Figura 7 – Previsão de RSSI da 4P-LogLogística para distâncias em $\log_{10}(x)$ metros.	62
Figura 8 – Previsão de RSSI da 4P-Logística para distâncias em metros.	63
Figura 9 – Comparação dos modelos de propagação LogDistance e NP-Logístico.	78
Figura 10 – Decaimento RSSI de <i>Wi-Fi</i> de acordo com a distância ao <i>accesspoint</i>	79
Figura 11 – Canais não sobrepostos para WLANs de 2,4 GHz.	79
Figura 12 – Absorção do sinal por cada parede atravessada.	80
Figura 13 – Interseção de retas.	80
Figura 14 – Representação do ambiente a partir do arquivo <i>DXF</i> contendo a planta-baixa do bloco A do IFMG <i>campus</i> Formiga.	81
Figura 15 – Simulação da propagação utilizando 256 cores no mapa de calor.	81
Figura 16 – Regiões com sinal abaixo da sensibilidade máxima dos equipamentos no bloco C.	82
Figura 17 – Grafo do ciclo de execução do algoritmo.	83
Figura 18 – Divisão interna da GPU dos blocks, grids e threads.	84
Figura 19 – Captura realizada no Bloco C, para um *access point* "18:8B:9D:69:E8:B2" posicionado em (x=660,y=260) e irradiando no Canal 11 (2.462 GHz).	85
Figura 20 – Simulação realizada com dados empíricos nas mesmas configurações do bloco C.	85
Figura 21 – Simulação da propagação de sinais de microondas no bloco A utilizando versão inicial do algoritmo.	88
Figura 22 – Simulação da propagação inicial de sinais de microondas no bloco A utilizando versão inicial do algoritmo.	89
Figura 23 – Comportamento da função objetivo do <i>SimulatedAnnealing</i> na busca do melhor ponto.	90
Figura 24 – Simulação da propagação de sinais de microondas no bloco A utilizando 1 AP, sua absorção e limiar de sensibilidade ao longo das salas e corredores do segundo piso do bloco A.	91

Figura 25 – Interpretação em forma de gráfico de pizza do resultado dado para a otimização de um <i>accesspoints</i> no bloco A.	92
Figura 26 – Simulação da propagação de sinais de microondas no bloco C utilizando 1 AP.	92
Figura 27 – Representação em gráfico de pizza do resultado dado para a otimização de um <i>accesspoints</i> no bloco C.	93
Figura 28 – Simulação da propagação de sinais de microondas no bloco A utilizando 2 APs.	94
Figura 29 – Representação do resultado dado para a otimização de dois <i>accesspoints</i> no segundo piso do bloco A.	95
Figura 30 – Simulação da propagação de sinais de microondas no bloco C utilizando 2 APs.	96
Figura 31 – Representação do resultado obtido para a otimização de dois <i>accesspoints</i> no bloco C.	97
Figura 32 – Simulação da propagação de sinais de microondas no bloco C utilizando 3 APs com potência de -25 dB.	97
Figura 33 – Representação do resultado para a otimização de três <i>accesspoints</i> no bloco C.	98
Figura 34 – Simulação da propagação de sinais de microondas no bloco A utilizando 3 APs com potência de -25 dB.	99
Figura 35 – Representação do resultado para a otimização de três <i>accesspoints</i> no bloco A.	100

Lista de quadros

Lista de tabelas

Tabela 1 – Fatores para parâmetro para o <i>Simulated Annealing</i> na primeira iteração.	69
Tabela 2 – Candidatos a parâmetros na primeira iteração.	69
Tabela 3 – Fatores para parâmetro para o <i>Simulated Annealing</i> na segunda iteração.	70
Tabela 4 – Candidatos a parâmetros na segunda iteração.	70
Tabela 5 – Fatores para parâmetro para o <i>Simulated Annealing</i> na terceira iteração.	70
Tabela 6 – Candidatos a parâmetros na terceira iteração.	71
Tabela 7 – Parâmetros definitivos utilizados na metaheurística.	71

Lista de abreviaturas e siglas

2D	2 dimensões
3D	3 dimensões
AP	<i>Access Point</i>
API	<i>Application Programming Interface</i>
CAD	<i>Computer-aided Design</i>
CPU	<i>Central Processing Unit</i>
HPC	<i>High Performance Computing</i>
CUDA	<i>Compute Unified Device Architecture</i>
dB	<i>Decibel</i>
dBm	<i>Decibel Miliwatt</i>
DWG	<i>Drawing</i>
DXF	<i>Drawing Exchange Format</i>
EMI	<i>Electromagnetic Interference</i>
FOSS	<i>Free Open Source Software</i>
Gb/s	<i>Gigabits por segundo</i>
GB	<i>Gigabits</i>
GHz	<i>Gigahertz</i>
GNU	<i>Gnu's Not Unix</i>
GPL	<i>General Public License</i>
GPS	<i>Global Positioning System</i>
GPU	<i>Graphics Processing Unit</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
ISM	<i>Industrial, Scientific and Medical</i>

LAN	<i>Local Area Network</i>
MHz	<i>Megahertz</i>
MIMO	<i>Multiple Input Multiple Output</i>
PCIe	<i>Peripheral Component Interconnect Express</i>
RF	<i>Radio Frequency</i>
SA	<i>Simulated Annealing</i>
T-R	<i>Transmitter-Receiver</i>
TI	Tecnologia da Informação
Wi-Fi	<i>Wireless Fidelity</i>
WLAN	<i>Wireless Local Area Network</i>

Listas de símbolos

Δ	Variação da função objetivo
μ	Média aritmética de uma distribuição
δ	Delta minúsculo
ξ	Ksi
σ	Sigma
α	Fator de decaimento da temperatura
γ	Gamma

Sumário

1	INTRODUÇÃO	25
1.1	Justificativa	26
1.2	Objetivos	27
1.3	Estrutura do Trabalho	27
2	FUNDAMENTAÇÃO TEÓRICA	29
2.1	Wireless AP Placement	29
2.2	Tecnologias Wi-Fi (WLANs IEEE 802.11)	29
2.3	Propagação de sinais de rádio	30
2.4	Modelos de propagação	32
2.4.1	Propagação no espaço livre (modelo de <i>Friis</i>)	32
2.4.2	<i>Two-rays ground reflection</i>	33
2.4.3	<i>Log-distance path loss</i>	35
2.4.4	<i>One-slope</i>	36
2.4.5	<i>Wall and floor factor</i>	36
2.4.6	<i>Log-normal fading</i>	37
2.4.7	<i>Ray-tracing fading</i>	37
2.5	Trabalhos Relacionados	38
2.5.1	<i>TamoGraph Site Survey</i>	38
2.5.2	<i>Netscout AirMagnet Survey</i>	39
2.5.3	<i>Ekahau Site Survey & Planner</i>	40
2.5.4	<i>D-Link Wi-Fi Planner PRO</i>	41
3	MATERIAIS E MÉTODOS	43
3.1	A Metaheurística	43
3.2	Linguagem Python	46
3.3	Bibliotecas utilizadas	47
3.4	Programação Paralela	48
3.4.1	CUDA	48
3.4.2	CUDA Toolkit	49
3.4.3	Numba	49
3.4.4	JIT (<i>just-in-time</i>)	50
3.4.5	Anaconda Navigator	51
4	PROJETO E DESENVOLVIMENTO	53
4.1	Representação do ambiente	53

4.1.1	Arquivo <i>DXF</i>	53
4.1.2	Escala e precisão	54
4.2	Propagação dos sinais	56
4.2.1	Definição do modelo de propagação para sinais <i>Wi-Fi</i>	56
4.2.2	Ajuste do modelo de propagação	58
4.2.3	Simulação da propagação de sinais no ambiente	63
4.2.4	Atenuação do sinal ao atravessar paredes	64
4.3	Visualização dos dados	66
4.4	Heurística de otimização	67
4.4.1	Implementação do <i>Simulated Annealing</i>	67
4.4.2	Calibração dos parâmetros do <i>Simulated Annealing</i>	68
4.5	Avaliação da solução	71
4.5.1	Avaliação da solução com um AP	71
4.5.2	Aperfeiçoamento da função objetivo	72
4.5.3	Avaliação da solução para dois ou mais APs	72
4.6	Análise da utilização de recursos	73
4.7	Paralelização em GPU	74
4.8	Validação dos modelos	76
5	RESULTADOS E ANÁLISE	87
5.1	Simulação da propagação de sinais <i>wireless</i>	87
5.2	<i>Wi-Fi Placement</i> para 1 AP	89
5.3	<i>Wi-Fi Placement</i> para 2 ou mais APs	93
6	TRABALHOS FUTUROS	101
7	CONSIDERAÇÕES FINAIS	103
	REFERÊNCIAS	105

1 INTRODUÇÃO

A demanda por disponibilidade e cobertura de redes locais sem fio (WLAN), seja em ambiente corporativo ou doméstico, tem crescido vertiginosamente. À medida que o uso de *Wi-Fi* se torna trivial e cotidiano, suas tecnologias são aperfeiçoadas e os preços dos equipamentos para acesso sem fio se tornam mais acessíveis(MARQUES, 2006).

Em um ambiente onde computadores tipicamente realizam a sua comunicação através de rede local cabeada (LAN), caso se torna necessária a realização de alguma mudança física no ambiente, será inevitável o uso de arranhações, canaletas ou até obras na estrutura física do prédio. Conforme ocorrem alterações no ambiente de trabalho, variando desde a colocação dos móveis e divisórias a mudanças de sala, a limitação imposta pelos cabos se torna um problema de organização e planejamento das futuras mudanças. Quando há a necessidade de ampliar a rede para a acomodação de novos pontos de acesso para telecomunicação, a necessidade de se estender o atual cabeamento torna-se inconveniente. O obstáculo às alterações se torna ainda maior quando o ambiente é alguma construção em que não é viável realizar intervenções na parede (e.g., tombamento ou estética) para a redistribuição dos cabos, por vezes em edifícios, onde, sequer, houve planejamento de uma rede cabeada estruturada.

Considerando o acima exposto, é possível observar o quanto é conveniente o uso de redes sem fio (*wireless*) quando há a necessidade de ampliar a rede ou implantar, mesmo que temporariamente, o acesso à internet em alguma determinada área. Apesar da praticidade, não é viável ter uma rede *wireless* se sua cobertura de sinal não chegar boa parte da área necessária ou, então, se por mais perto que o ponto de acesso (*access point*) *wireless* esteja do computador, não provê uma qualidade de serviço satisfatória devido a um baixo nível de sinal ou interferências na mesma frequência do canal.

Isso posto, faz-se necessário um bom planejamento do posicionamento dos *access points wireless* de maneira a oferecer uma boa cobertura de sinal onde ouver demanda conectividade. É importante deixar claro que o tratamento de colisão de canais como sua interferência de *access points* próximos não está dentre os principais objetivos dessa pesquisa e, por conseguinte, não foi implementado. A proposta deste trabalho é desenvolver um *software* para *Wireless AP Placement*, que utilizará modelo(s) da propagação do sinal *Wi-Fi* de acordo com as características físicas do ambiente, aplicando metaheurísticas que visem ao aprimoramento da cobertura de sinal, tendo como caso de uso as dependências do IFMG *campus* Formiga. Tal *software* indicará um melhor posicionamento dos *access points Wi-Fi* para o ambiente analisado.

1.1 Justificativa



É importante ressaltar a validade prática desse trabalho a ser desenvolvido, uma vez que uma grande dúvida dos usuários residenciais e profissionais de TI é saber qual a melhor localização para o(s) *access point(s)* Wi-Fi. Alguma vezes, essa questão até passa despercebida para o usuário (doméstico ou empresarial), resultando em locais sem cobertura de sinal. Boa parte dos ambientes corporativos e residenciais utilizam um ou mais pontos de acesso sem fio para prover acesso à rede local e dela para a internet. Entretanto, dispositivos móveis e computadores tipicamente não observam uma qualidade satisfatória de sinal do *access point wireless*. Esse problema pode ocorrer pelo fato do *access point* não abranger toda a área necessária ou sofrer atenuações devido às características do ambiente.

Ademais, redes Wi-Fi podem sofrer interferências externas, tais como telefones sem fio ou “babás” eletrônicas que operem na mesma faixa de frequência (banda ISM de 2.4 GHz), motores elétricos ou outras redes sem fio que estejam próximos do *access point* e irradiem energia na mesma frequência do canal utilizado, seja ele na banda de 2.4 GHz (IEEE 802.11 b/g) ou 5.8 GHz (IEEE 802.11 a/ac). Por meio de uma análise do espectro de radiofrequência (RF) é possível detectar o nível dos sinais e/ou interferências em determinado local, entretanto, um problema facilmente observável que pode agravar a baixa qualidade observada no serviço da rede sem fio é a existência de pontos cegos na cobertura do sinal para determinado ambiente (RUBINSTEIN; REZENDE, 2002).

Como consequência de toda a atenuação e interferência sofrida, a rede wireless pode ficar inoperante ou dar a impressão de baixo desempenho. Por parte da maioria dos usuários finais, uma reclamação comum é a dificuldade de acessar determinado conteúdo e o acesso à rede parecer “lento”. De um modo geral, por serem leigos no assunto, muitos usuários de rede sem fio acabam culpando o provedor de acesso à internet ou equipe de TI da instituição pela má qualidade do serviço quando, na verdade, a solução para o problema passaria por uma inspeção no espectro de sinal do(s) *access point(s)* Wi-Fi (para identificar interferências) e um melhor posicionamento destes (para maximizar a cobertura do sinal e reduzir interferências entre *access points*).

Por fim, até onde pudemos verificar, não há disponível um *software* livre, gratuito e de código-fonte aberto para *Wireless AP Placement*, apenas soluções proprietárias, comerciais e custosas. Isto posto justifica-se o desenvolvimento da ferramenta proposta neste trabalho.

1.2 Objetivos

Após a contextualização do objeto de estudo deste trabalho e sua importância, sintetiza-se aqui seu objeto primário: projetar e implementar um *software* para planejamento do posicionamento dos *access point* (*Wireless AP Placement*), que receba como entrada uma representação do ambiente e informações dos *access points* disponíveis, realize simulações de propagação dos sinais (considerando as características do ambiente) para os posicionamentos de *access points* propostos por metaheurística que vise maximizar a cobertura do sinal *Wi-Fi* tendo, como caso de uso, as dependências do IFMG *campus* Formiga. São objetivos secundários e mais específicos os seguintes tópicos:

- Construir um modelo das dependências do IFMG *campus* Formiga, a partir da representação do ambiente fornecida ao *software*, e.g. praia(s)-baixa(s);
- Simular a propagação do sinal *wireless* de *access points Wi-Fi* através das dependências do *campus* Formiga, seguindo modelos de propagação de sinais sem fio;
- Utilizar metaheurística computacional para propor/verificar, via simulação, novas localizações para os *access points Wi-Fi*, visando maximizar a cobertura do sinal no *campus* (dentre outros fatores);
- Fornecer, ao final, uma solução proposta para o novo posicionamento dos *access points Wi-Fi*, se possível, auxiliada por algum método de visualização da cobertura do sinal *wireless* no ambiente;
- Disponibilizar o *software* de maneira livre e de código-fonte aberto (FOSS), distribuindo-o por meio de alguma licença que proteja a autoria do mesmo (e.g., Licença Pública Geral GNU — GPL).

1.3 Estrutura do Trabalho

Esta monografia de conclusão de curso é organizada em 8 capítulos. no capítulo 2, após a contextualização feita na introdução, são apresentados os conceitos teóricos necessários para a compreensão do trabalho. O capítulo 3 explícta a linguagem utilizada no desenvolvimento do trabalho, bem como as bibliotecas usadas e a definição da metaheurística. São abordados também os conceitos de programação paralela envolvidos no processo de criação da ferramenta. No capítulo 4 são tratados os tópicos sobre o desenvolvimento do projeto, desde a entrada de dados, propagação do sinal, validação do modelo utilizado, à paralelização do algoritmo. O capítulo 5 apresenta os resultados obtidos após as simulações da propagação de sinais *wireless* nas dependências do IFMG *campus* Formiga e uma análise dos resultados encontrados nas simulações com um, dois ou mais *access points*.

No capítulo 6 é feita uma alusão aos possíveis trabalhos futuros e, por fim no capítulo 7 são feitas as considerações finais deste trabalho, seguidas pelas referências bibliográficas utilizadas neste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

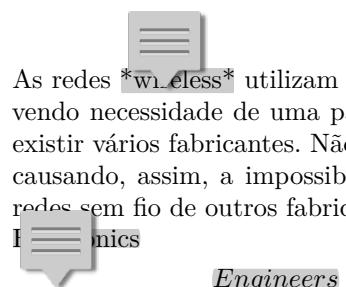
No levantamento do referencial bibliográfico realizado, observam-se um vasto número de trabalhos que tratam sobre a otimização do posicionamento de *access points wireless*. Utilizaremos, como principais fundamentações teóricas deste projeto, o trabalho de Battiti, Brunato e Delai: *Optimal Wireless Access Point Placement for Location-Dependent Services* e a obra *Comunicações sem fio - Princípios e Práticas*, de Theodore S. Rappaport.

2.1 *Wireless AP Placement*

De acordo com o trabalho publicado por Battiti, Brunato e Delai (R. Battiti M. Brunato et al., 2004), “*Optimal Wireless Access Point Placement for Location-Dependent Services*”, vários grupos de pesquisadores independentes têm proposto métodos para fazer estimativa a respeito da posição do usuário (DALSSOTO; SOUZA; BECKER, 2013), com base na intensidade dos sinais de rádio recebidos de múltiplos *access points* (NAJNUDEL, 2004). Partindo desta distinta aplicação, os pesquisadores propõem uma nova abordagem para o *AP Placement*, pois consideram que a localização “é um importante parâmetro que pode ser usado para determinar o comportamento do sistema” (R. Battiti M. Brunato et al., 2004, p. 1). Desse modo, a proposta deste TCC, que utilizará um modelo de propagação do sinal *Wi-Fi* em uma simulação do ambiente do IFMG *campus Formiga* é, mais uma vez, justificada pela necessidade de se ter um *software* livre e gratuito para *Wireless AP Placement*, que indique um melhor posicionamento dos *access points Wi-Fi*.

2.2 *Tecnologias Wi-Fi (WLANs IEEE 802.11)*

Quando se trata de redes *wireless*, é imprescindível dizer que, com o estabelecimento da família de padrões IEEE 802.11, se tornou possível a compatibilidade entre diferentes marcas de fabricantes de *access points wireless* e dispositivos móveis. De acordo com Franciscatti,



As redes **wireless** utilizam freqüências de rádio para se comunicar havendo necessidade de uma padronização dos equipamentos sem fio por existir vários fabricantes. Não havia uma padronização dessa tecnologia causando, assim, a impossibilidade de comunicação de dispositivos de redes sem fio de outros fabricantes. Assim o **Institute of Electrical and Electronics Engineers**

* (IEEE) formou um grupo de trabalho com o objetivo de definir os padrões de uso em redes sem fio, denominado 802.11. (FRANCISCATTI, 2005 apud BOF, 2010)

Desta forma, após a padronização definida pela IEEE, alguns padrões para WLAN foram estabelecidos (RIVERA, 2010; BANERJI; Singha Chowdhury, 2013), como:

- **IEEE 802.11a:** Definido em setembro 1999, operando na frequência de 5 GHz, uma largura de banda de 20 MHz, taxa de transmissão de até 54 Mbit/s e podendo ter um alcance de 35 metros *indoor* e até 5 quilômetros *outdoor*;
- **IEEE 802.11b:** Definido em setembro 1999, operando com uma frequência de 2.4 GHz, uma largura de banda de 22 MHz, taxa de transmissão de até 11 Mbit/s e com alcance de 35 metros *indoor* e 140 metros *outdoor*;
- **IEEE 802.11g:** Definido em junho de 2003, operando com uma frequência de 2.4 GHz, utilizando uma largura de banda de 20 MHz, com uma taxa de transmissão de até 54 Mbit/s e com um alcance de 38 metros *indoor* e 140 metros *outdoor*;
- **IEEE 802.11n:** Definido em outubro de 2009, operando com uma frequência de 2.4 GHz (802.11a) e 5 GHz (802.11g), com uma taxa de transmissão de até 72.2 Mbit/s utilizando uma largura de banda de 20 MHz e uma taxa de transmissão de até 150 Mbit/s com uma banda de 40 MHz, permitindo múltiplos fluxos espectrais (MIMO). O alcance vai de 70 metros *indoor* e 250 metros *outdoor*;
- **IEEE 802.11ac:** Definido em dezembro de 2013, operando com uma frequência de 5 GHz, utilizando uma largura de banda que vai de 20 MHz até 160 MHz, com uma taxa de transmissão de 87.6 Mbit/s até 866.7 Mbit/s, também com uso de MIMO.

Considerando os padrões supracitados, iremos focar especificamente na penetração dos sinais *Wi-Fi* na estrutura dos edifícios (paredes, andares).

2.3 Propagação de sinais de rádio

O sinal *Wi-Fi* produzido pelo *access point* nada mais é, de um modo simplista, que uma onda de rádio que se propaga no espaço. De acordo com Rappaport,

Os mecanismos por trás da propagação da onda eletromagnética são diversos, principalmente podem ser atribuídos a reflexão, difração e dispersão. (...) Devido a múltiplas reflexões de vários objetos, as ondas eletromagnéticas trafegam por diferentes caminhos de tamanhos variados. A interação entre essas ondas causa uma distorção de caminhos

múltiplos em um local específico, e as intensidades das ondas diminui à medida que a distância entre transmissor e receptor aumenta. (RAPPA-PORT, 2009, p. 72)

Por ser uma onda eletromagnética, a onda do sinal *Wi-Fi* como descrito por Torlak, apresenta tal comportamento, sofrendo reflexão, difração e dispersão (TORLAK, 2016). Outro aspecto que deve ser abordado acerca da onda do sinal *Wi-Fi* é a diferença entre modelos de propagação de larga escala (perdas) e pequena escala (atenuação). Os modelos de propagação em larga escala são caracterizados pela intensidade do sinal para grandes distâncias de separação do transmissor e receptor (podendo variar de várias centenas ou milhares de metros), enquanto modelos de propagação em pequena escala são caracterizados pelas flutuações rápidas do sinal recebidos para distâncias muito curtas (variando de alguns comprimentos de onda ou frações de segundos) (RAPPAPORT, 2009, p. 72).

Rappaport define três modelos básicos de propagação que são usados para a previsão da intensidade do sinal recebido a determinada distância do transmissor, numa larga escala. O modelo de propagação no espaço livre (*Friis*) é utilizado quando transmissor e receptor possuem uma linha de visão desobstruída, ou seja, não há obstáculos entre eles que interrompam ou alterem o caminho da transmissão do sinal. O modelo de propagação no espaço livre oferece uma noção da ordem de magnitude do sinal recebido, mas é demasiado otimista pois raramente há um único caminho entre a antena transmissora e a antena receptora: em situações reais, haverá reflexão do sinal no solo. Para grandes distâncias e antenas altas, o modelo de reflexão no solo é razoavelmente preciso para prever a intensidade do sinal recebido. Esse modelo é baseado na ótica geométrica e considera o caminho direto e o caminho refletido (modelo de dois raios), que muitas vezes é no solo. E, por fim, tem-se o modelo de difração (por gume de faca) que torna possível propagar os sinais de rádio através de obstruções, bem como ao redor da superfície da terra, além do horizonte. Contudo, a força do campo recebido diminui rapidamente quando o receptor se aproxima do obstáculo em direção à região obstruída (sombra), porém, o campo de difração ainda existe e normalmente tem força suficiente para produzir um sinal útil (RAPPAPORT, 2009, p. 72-83).

A propagação no interior é dominada pelos mesmos mecanismos (reflexão, difração e dispersão), porém as condições são muito variáveis, podendo variar, por exemplo, até mesmo se as portas e janelas estiverem fechadas ou dependendo do local onde as antenas são montadas. Dentro do mesmo contexto, existem várias características físicas e elétricas que podem influenciar na propagação do sinal, como qual o tipo de ambiente (escritório ou casa) e de que é feita a construção (madeira, tijolo, concreto ou até ferragem). Durante a propagação, pode também ocorrer a perda do sinal entre andares de um edifício, obedecendo à lei de potência da distância, a qual leva em consideração o tipo de prédio, arredores e uma variável aleatória normal que representa o desvio padrão (RAPPAPORT,

2009, p. 104-108). Com isso, na próxima seção são apresentados os modelos de propagação que serviram como base da simulação da propagação de sinais *wireless*.

2.4 Modelos de propagação

Quando se deseja realizar um bom desempenho e planejamento da cobertura do espectro *Wi-Fi*, é indispensável o conhecimento do meio de transmissão e qual modelo se deve usar para obter um resultado mais realista. Em sistemas *wireless* o meio de propagação utilizado é o canal de rádio, de forma que as características e efeitos sobre todas as informações trafegadas são de uma natureza complexa, fazendo com que medições empíricas sejam de suma importância. Com as medições é possível ver como é o comportamento de modelos em pequena e larga escala, além de ser possível determinar a variação da potência do sinal devido ao movimento de pessoas no ambiente ou atingir obstáculos fixos, como paredes, pisos, vidros, móveis, dentre outros.

Então, se faz necessário uma boa escolha de quais modelos de propagação utilizar quando se quer obter uma boa representação do espectro e que ao mesmo tempo esteja condizente com a realidade. Quanto maior for a precisão desejada, mais detalhes sobre o ambiente de propagação devem ser modelados. Nesta seção são comentados sucintamente sete modelos que foram estudados para a realização deste trabalho.

2.4.1 Propagação no espaço livre (modelo de *Friis*)

O modelo *Friis free-space path loss* ou geralmente tratado como modelo de propagação no espaço livre de Friis é usado para prever a intensidade do sinal recebido quando a antena transmissora e a antena receptora possuem um caminho de linha de visão limpa, ou seja, um caminho desobstruído de qualquer objeto ou edificação (LUO, 2013). Tal modelo é em geral usado em sistemas de comunicação por satélite e em enlaces de rádio de microondas com linha de visão. Assim como os modelos *outdoor*, o modelo de propagação no espaço livre de Friis, pressupõe que a potência recebida diminui com uma função da distância entre a antena transmissora e a antena receptora elevada a alguma potência, ou seja, uma função da lei de potência. A potência recebida através do espaço livre pela antena receptora separada da antena transmissora por uma distância d pode ser calculada pela seguinte equação:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \quad (2.1)$$

sendo:

- P_t a potência transmitida;

- $P_r(d)$ a potência recebida na distância d ;
- G_t é o ganho da antena transmissora;
- G_r é o ganho da antena receptora;
- d é a distância de separação das antenas;
- L é o fator de perda ($L \geq 1$), $L = 1$ indica nenhuma perda no *hardware* do sistema;
- λ é o comprimento de onda dado em metros.

O modelo de espaço livre de *Friis* é apenas uma previsão válida para uma potência recebida para uma distância d de separação entre as antenas. O campo distante que é criado entre as antenas pode ser chamado de região de *Fraunhofer*, onde essa região é definida como uma região além da distância de campo distante d_f , que está diretamente relacionada com a maior dimensão linear D de abertura da antena transmissora e com o comprimento de onda da portadora (antena receptora) (RAPPAPORT, 2009). Tal distância de *Fraunhofer* pode ser calculada pela seguinte equação:

$$d_f = \frac{2D^2}{\lambda} \quad (2.2)$$

O cálculo da potência recebida terá uma falha quando a distância é zero. Por este motivo, modelos de propagação em larga escala utilizam uma distância próxima, d_0 , com um ponto de referência de potência conhecido. A potência recebida, $P_r(d)$, em qualquer distância que a distância d seja maior que zero, pode estar relacionada com a potência recebida no ponto de referência d_0 (RAPPAPORT, 2009). A equação que calcula a potência recebida pode ser vista abaixo utilizando uma distância maior que d_0 :

$$P_r(d) = P_r(d_0) \left(\frac{d_0}{d} \right)^2 \quad (2.3)$$

O valor da distância de referência d_0 em sistemas práticos em antenas de baixo ganho, entre 1 e 2 GHz, normalmente é utilizado com sendo 1 metro em ambientes internos (*indoor*) e 100 metros ou 1 quilômetro para ambientes externos (*outdoor*), de forma que o resultado obtido pelas equações anteriores são múltiplos de 10, tornando os cálculos de perda de caminho factíveis em unidade de dB.

2.4.2 Two-rays ground reflection

O modelo *Two-rays ground reflection* é um modelo de propagação de rádio que prevê as perdas de trajetória entre uma antena transmissora e uma antena receptora. Em geral, as duas antenas têm altura diferente e raramente possuem uma linha de visão (LUO,

2013). O sinal é recebido de duas formas: por LOS (linha de visão) e o por *multipath* (multicaminho) formado predominantemente por uma única onda refletida no solo.

Quando a distância entre as antenas é menor do que a altura da antena transmissora, duas ondas são adicionadas de forma positiva para gerar maior potência e, à medida que a distância aumenta, essas  se somam de forma construtiva e destrutiva, proporcionando regiões de exaustão e  à medida que a distância aumenta além da distância crítica ou primeira zona de *Fresnel*, a potência cai proporcionalmente quatro vezes ao inverso da potência da distância. Essa é uma perda no caminho muito mais rápida do que é experimentada no *Friis free-space path loss model* (espaço livre de Friis).

Quando se tem valores muito altos para a distância, pode-se notar que a potência recebida e a perda no caminho se tornam independentes da frequência (RAPPAPORT, 2009). O modelo de reflexão de dois raios é uma formulação matemática de um tipo de interferência *multipath* quando a interferência é considerada como consistindo em dois caminhos: (I) do transmissor ao receptor diretamente, (II) do transmissor, refletido fora do chão, para o receptor.

A potência recebida a uma distância d do transmissor para o modelo *Two-rays ground reflection model* pode ser expressa como:

$$P_r(d) = P_t G_t G_r \frac{h_t^2 h_r^2}{d^4} \quad (2.4)$$

sendo:

- $P_r(d)$ a potência recebida com distância d ;
- P_t a potência transmitida;
- G_r o ganho da antena receptora;
- G_t o ganho da antena transmissora;
- h_t a altura da antena transmissora;
- h_r a altura da  receptora;
- d a distância do transmissor.

A perda do caminho para o modelo *Two-rays ground reflection model* pode ser expressa em dB com a seguinte equação:

$$PL(dB) = 40\log(d) - (10\log(G_t) + 10\log(G_r) + 20\log(h_t) + 20\log(h_r)) \quad (2.5)$$

2.4.3 Log-distance path loss

O modelo *Log distance path loss* é um modelo genérico e uma extensão do modelo de espaço livre de Friis. Ele é usado para prever a perda de propagação para uma ampla gama de ambientes, enquanto que o modelo Friis é restrito ao caminho desobstruído entre o transmissor e o receptor (LUO, 2013).

O principal critério ou característica deste modelo é  derar que a perda no caminho é logaritmicamente dependente da distância. Logo a perda no caminho calculada com uma distância d entre um transmissor e receptor (geralmente dado em quilômetros) . Na região mais distante do transmissor (onde $d \geq d_0$), se $PL(d_0)$ é a perda de percurso medida em dB a uma distância d_0 do transmissor, então a perda do caminho (a perda na potência do sinal em dB quando se desloca de distância d_0 a d) a uma distância arbitrária $d > d_0$ é dada pela expressão:

$$PL(d) = PL(d_0) + 10 \cdot n \cdot \log\left(\frac{d_0}{d}\right) \quad (2.6)$$

sendo:



- d é a distância dada em quilômetros em todos os casos;
- d_0 é a distância inicial de referência;
- $PL(d)$ é a perda no caminho para a distância d ;
- $PL(d_0)$ é um valor de perda de caminho para uma distância de referência;
- n é o expoente de propagação e indica a taxa na qual a perda de caminho aumenta com a distância (MATHURANATHAN, 2013).

Geralmente, para modelar ambientes reais, os efeitos *shadowing* (de sombreamento) não podem ser negligenciados. Se os efeitos de sombreamento são deixados de lado, a perda do caminho, quando representada em um gráfico que  representa a potência recebida e a distância, é simplesmente uma linha reta. Para adicionar um efeito de sombreamento e deixar o resultado final mais próximo da realidade, uma variável aleatória Gaussiana de média zero com desvio padrão σ é adicionada à equação. A perda real do caminho ainda pode variar devido a outros fatores, assim como os efeitos de reflexão, difração e dispersão (RAPPAPORT, 2009). Assim, o expoente da perda de caminho (a literatura define alguns em ambientes diferentes) e o desvio padrão da variável aleatória escolhida, devem ser bem conhecidos para uma boa modelagem da perda (LUO, 2013).

2.4.4 One-slope

O modelo *One-slope* é classificado como sendo um modelo empírico e assume que a perda no caminho dada em dBm é linearmente na distância logarítmica da distância d entre o transmissor e receptor (LUO, 2013):

$$PL(d) = L_0 + 10 \cdot n \cdot \log(d) \quad (2.7)$$

onde:

- d é a distância entre transmissor e receptor;
- $PL(d)$ é a perda no caminho para a distância d ;
- L_0 é a perda no caminho calculado em uma distância de 1 metro;
- n é o expoente de perda no caminho.

Claramente, este modelo baseia-se na perda do espaço livre e visa incluir todas as perdas devido a vários mecanismos de propagação pelo caminho usando um expoente n de perda. Por ser um modelo simples, se torna muito fácil de realizar sua implementação, mas se usado de forma imprecisa pode levar a grandes erros em ambientes internos, pois é possível que um grande número de objetos interfiram nos meios de propagação, ou seja, não será possível obter um resultado com uma precisão próxima da realidade.

2.4.5 Wall and floor factor

O modelo *Wall and floor factor* leva em conta a absorção em paredes e pavimentos e geralmente é usado em ambientes internos e considera a perda no espaço livre (LUO, 2013; RAPPAPORT, 2009). Ele se resume basicamente na perda no espaço livre de um ambiente interno somado com uma perda adicional relacionada a cada piso atravessado e o número de paredes interceptadas em uma linha direta entre o transmissor e receptor.

Segundo Xie, este modelo de propagação tem um desempenho melhor que o modelo *One-slope*, uma vez que proporciona mais graus de liberdade na consideração de obstáculos (XIE, 2013). A perda no caminho utilizando o modelo de propagação *wall and floor factor* pode ser calculada pela seguinte equação:

$$PL(d) = L_1 + 20 \cdot \log(d) + n_f \cdot L_f + n_w \cdot L_w \quad (2.8)$$

sendo:

- d é a distância entre transmissor e receptor;

- $PL(d)$ é a perda no caminho para a distância d ;
- L_1 é a perda no caminho calculado em uma distância de 1 metro;
- n é o expoente de perda no caminho;
- L_f e L_w são respectivamente perdas causadas pela penetração no piso e nas paredes;
- n_f e n_w são respectivamente os números de pisos e de paredes.

2.4.6 Log-normal fading

Este modelo de propagação estatístico provê uma estimativa aproximada do que representa a perda no caminho como uma função da distância e outros parâmetros como a frequência e a altura da antena (BUDGETS, 2013). Desta forma é possível obter uma previsão da qualidade da intensidade do sinal quando aumenta-se a distância entre o transmissor e receptor. Vale ressaltar  para se obter uma representação mais realista da perda do sinal, é necessária uma grande coleta de dados empíricos.

$$f(x; \mu; \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left[-\frac{(\ln(x) - \mu)^2}{2\sigma^2}\right] \quad (2.9)$$

Uma possível explicação para o motivo pelo qual este modelo utiliza uma distribuição *log-normal* é que, para cada caminho há muitos fatores que contribuem com a perda do sinal, incluindo a combinação de perda no espaço livre, difrações, reflexões, interferências de equipamentos, dentre outros motivos. Para cada uma dessas perdas, é utilizada uma variável aleatória que a representa (BUDGETS, 2013). A sua perda é dada em dB e é o somatório de todas essas perdas (também expressadas em dB). O teorema do limite central afirma que tal distribuição tenderá a uma distribuição normal.

2.4.7 Ray-tracing fading

Segundo (VALENZUELA, 1993), para esse modelo de propagação é realizado o traçado de raios em todas as direções possíveis do receptor ao transmissor, utilizando os princípios da ótica geométrica. Este mesmo conceito é utilizado em sistemas de realidade virtual para que possam se tornar visíveis todos os objetos dentro de um ambiente. Nos cálculos da simulação do ambiente são levadas em consideração a reflexão, difração, espalhamento do sinal, dentre outros fenômenos físicos.

Valenzuela também afirma que o modelo de propagação *Ray Tracing* assume que todos os objetos no ambiente de propagação são objetos refletores em potencial. Para isso, considera também somente os caminhos que realmente existem entre o transmissor e receptor. A complexidade do ambiente escolhido tem um forte impacto em seu consumo

de recurso computacional, uma vez que, quanto mais obstáculos forem adicionados, mais reflexões, difrações e cálculos serão feitos (VALENZUELA, 1993). Após a apresentação dos modelos físicos teóricos, na próxima seção são apresentados alguns *softwares* presentes no mercado, que realizam funções similares ao desenvolvido neste projeto.

2.5 Trabalhos Relacionados

Nesta seção são apresentados alguns *softwares* que estão presentes no mercado e oferecem serviços similares aos objetivos propostos nesse trabalho. São citadas ferramentas que realizam um trabalho semelhante ao desenvolvido aqui, como o *TamoGraph Survey*, *AirMagnet Survey*, *Ekahau Site Survey*, *D-Link Wi-Fi Planner* e *Xirrus Wi-Fi Designer*. Existem várias outras  ferramentas que trabalham no mesmo meio, porém, os *softwares* apresentados a seguir possuem conhecimentos aplicados neste trabalho e na área da Ciência da Computação.

2.5.1 *TamoGraph Site Survey*

O *TamoGraph*¹ é uma ferramenta de *site survey* usada para a coleta, visualização e análise de dados *Wi-Fi* 802.11 com padrões a/b/g, a/n/ac. É muito usada quando se tem a implantação e a manutenção de redes sem fio, uma vez que facilita tarefas que são demoradas e muitas das vezes, até complexas de se obter um bom resultado. O *TamoGraph* realiza como tarefas análises contínuas e relatórios de intensidade/qualidade do sinal, ruídos e  interferências, alocações de canais, taxas de dados transmitidos, dentre outros.

A  ferramenta aposta que, as empresas que fizerem o seu uso, poderão reduzir drasticamente o seu tempo e custos envolvidos em implantações e manutenções de redes *wireless*, melhorar o desempenho e cobertura da rede em todos os tipos de ambiente, desde ambientes *indoor* como prédios com escritórios, aeroportos e *shoppings*, até ambientes *outdoor* como pátios, praças, campos e estacionamentos.

A empresa *TamoSoft* considera ser praticamente impossível considerar todas as variáveis que possam afetar a saúde e o desempenho da rede. Segundo ela em seu site, alterar as condições do cenário, até mesmo de algo aparentemente menor, como um *notebook* conectado à rede sem fio de um escritório, pode afetar gravemente o seu desempenho, que também pode ser influenciado pela ampla proliferação de redes sem fio com fatores de interferência. Por estes fatores, a ferramenta da *TamoSoft* pode ser considerada como profissional e de essencial uso para empresas, inclusive para usuários comuns.

O *TamoGraph* pode ser executado em *Microsoft Windows 7*, *Windows 8*, *Windows 8.1*, *Windows 10*, *Windows Server 2008 R2*, *Windows Server 2012*, *Windows Server 2012*

¹ <<http://www.tamos.com/products/wifi-site-survey/>>

R2 e versão para *MacBooks*. Possui versões em 32 e 64 bits e requer um adaptador de rede *wireless* compatível. A sua licença mais básica custa US\$899.00 dólares, aproximadamente R\$2.850,00 e a licença profissional custa US\$ 1.199,00, aproximadamente R\$ 3.802,00, cotação 26/11/2017. Ambas as licenças são vitalícias e apenas para um usuário.

2.5.2 *Netscout AirMagnet Survey*

O *AirMagnet Survey*² é um *software* de *survey* para redes sem fio que propõe uma solução para *softwares* que realizam a análise de redes sem fio locais que necessitam projetar e planejar LANs sem fio com o padrão 802.11 a/b/g/n/ac com desempenho, segurança e conformidade. O *software* calcula a quantidade, a alocação e configurações ideais para a realização de uma rede local *wireless* com um bom desempenho.

A *Netscout*, empresa responsável pelo *AirMagnet*, diz que o seu produto vai além do que uma simples cobertura dos sinais de radiofrequência. Ele traça o desempenho de rede real do usuário final nos termos da velocidade de conexão, taxa de transferência e estatísticas do pacote e computa como resultado, um mapa completo do ambiente coberto pelo *Wi-Fi*, permitindo ao usuário implantar sua rede corretamente, já no primeiro momento, evitando custos de retrabalho e reclamações posteriores, além de ter fidelidade nos serviços prestados.

O *AirMagnet* permite que os usuários possam integrar analisadores de espectro profissionais para obter os dados do sinal *wireless* em uma única varredura, modelar cenários antes da implantação para estimar orçamentos, definir estratégias de migração para novas tecnologias, obter relatórios de pesquisa personalizados, executar inspeções internas usando dispositivos providos de tecnologia *GPS*, realizar inspeções de *VoiceOver* no local de implantação da rede *wireless* (para que esteja pronta para suportar serviços de voz), certificar a rede para os requisitos de aplicativos e rede dos usuários finais, com um planejamento detalhado da capacidade de usuários finais.

O *AirMagnet Survey* possui versões *Express* que oferecem uma versão mais simples de *survey* para padrão 802.11ac, permitindo que o usuário execute um exame básico do local de implantação da rede *wireless*, possibilitando mapear o sinal, ruído e até mesmo o desempenho de usuários. O *AirMagnet* possui também a sua versão *Pro*, que amplia ainda mais as capacidades oferecidas pela versão *Express*. Nela é adicionada a funcionalidade “*Planner*”, pela qual é possível realizar desde a implantação de um *access point* até o orçamento dos gastos, além de suporte para a implantação de vários andares, inspeções técnicas de ambientes externos (*outdoor*), verificação e análise de prontidão para serviços de voz, análise do espectro de radiofrequência e mais outros recursos.

É possível executar o *software* no ambiente *Windows* em todas as versões 64 bits

² <<http://enterprise-pt.netscout.com/products/airmagnet-survey>>

iguais ou superiores ao *Windows 7* e em ambientes *OSX* em que a versão é igual ou superior ao *Mac OS X v 10.5 (Leopard)*. O preço da licença para o uso do *software* não é informado no site da empresa. O orçamento deve ser feito pelo contato com representantes.

2.5.3 *Ekahau Site Survey & Planner*

A *Ekahau Wireless Design* é mais uma empresa que fornece um *software* para soluções sobre redes sem fio, batizado de *Ekahau Site Survey (ESS)*³. O *Ekahau Site Survey* propõe um *design* e análise experiente sobre a tecnologia *Wi-Fi*. A *Ekahau* define seu *software* como sendo um conjunto completo de ferramentas para projetar, analisar, otimizar e solucionar problemas de redes *wireless*.

O ESS não deixa de ser um instrumento para verificação e solução de uso fácil em redes *Wi-Fi*. Foi desenvolvido para engenheiros e arquitetos de redes sem fio (desde sistemas integrados até administradores da área de TI). A empresa ainda garante o alto desempenho e capacidade para qualquer rede *Wi-Fi* com padrões 802.11ac e n. Se caso uma rede ainda não esteja em seu desempenho ótimo ou ainda não foi implantada, o ESS irá sugerir automaticamente o devido posicionamento e as configurações ideais para o *access point*.

Com a ferramenta, também é possível criar automaticamente um plano da rede *Wi-Fi* de vários andares com base em requisitos de desempenho e capacidades especificados. Quase que de imediato, o ESS irá identificar o número ideal de *access points*, com os melhores locais para seus posicionamentos e seus respectivos canais, simulando o comportamento de como a rede irá ser executada antes de ir para o local. Com sua funcionalidade “*3D Planner*”, é considerado o espalhamento do sinal entre os andares do prédio para ajudar a minimizar a interferência dos canais.

Com o ESS é fácil realizar a análise em profundidade com mapas de calor. Nos mapas de calor é possível visualizar, por exemplo, a intensidade do sinal, a taxa de dados, perda de pacotes, a sobreposição de canais, espalhamento do espectro, dentre outras características. É possível também realizar a análise de capacidade mais abrangentes, por exemplo, todas as descobertas podem ser compiladas em um simples relatório utilizando um sistema de relatório desenvolvido pela *Ekahau*.

O *software* foi projetado para ser executado em ambientes *Windows* e *MacOS*, e é uma ferramenta caracterizada por ser, de fato, essencial para engenheiros de redes sem fio em empresas de todos os tamanhos. A sua licença *Standard* custa US\$2,295.00, aproximadamente R\$ 7.278,00 e vai até sua versão *Premium Pack* custando US\$ 5,649.00, aproximadamente R\$ 17.914,00; a versão *Pro Pack* com o valor de US\$ 5,995.00, aproximadamente R\$ 19.011,00, cotação 26/11/2017. Todas as licenças são anuais e devem ser

³ <<https://www.ekahau.com/products/ekahau-site-survey/overview/>>

 renovadas a cada ano.

2.5.4 D-Link Wi-Fi Planner PRO

 O *D-Link Wi-Fi Planner PRO*⁴, como indicado pelo nome, foi desenvolvido pela famosa empresa D-link. Com essa ferramenta, é possível ter uma visão do ambiente como um todo, antes da implantação da rede *Wi-Fi*. Isso faz com que o planejamento, a comunicação e a boa qualidade do serviço prestado entre WLAN e clientes sejam melhorados.

Para a execução do *software* é necessário criar uma pasta para o projeto. Logo após, o programa pede para ser carregado uma imagem que represente a planta do ambiente. A planta não tem a necessidade de ser exatamente igual, apenas é necessária uma imagem que possa ser o rascunho inicial para planta. Em seguida deve ser informada a escala, para estimar a medida da planta.

Para que a execução do programa seja possível, as zonas de coberturas e as zonas de exclusão tem de ser definidas. O próprio usuário pode marcar os obstáculos (portas e paredes) e indicar as zonas especiais como sendo espaços fechados para salas e escritórios. Isso faz com que o WFP tenha uma simulação mais precisa e próxima da realidade. Depois de feitas as configurações, um módulo chamado *AP Placement Advisor* irá fornecer uma sugestão sobre o número de *access points* e o posicionamento necessários para eles terem uma maior cobertura do local.

A empresa não fornece mais informações sobre quais os requisitos mínimos necessários para a utilização de sua ferramenta ou preços, mas para que seja possível obter mais informações é necessário obter cadastro como empresa e entrar em contato com a mesma [por email](#).

 Assim, no próximo capítulo são apresentados os materiais e métodos utilizados para a implementação do *software* desenvolvido neste trabalho.

⁴ <<http://tools.dlink.com/intro/wfp/>>

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os materiais e métodos utilizados para o desenvolvimento do projeto, tais como as bibliotecas do *Python*, o processo com o arquivo de *AutoCad*¹ para a representação da matriz de propagação, a paralelização do código, utilizando a GPU para um melhor desempenho da simulação, a metaheurística utilizada para a otimização do posicionamento de *access points* e o Projeto Fatorial 2^k para a calibração dos parâmetros da metaheurística.

Para definir qual modelo de propagação deveria ser utilizado, foram usados dois softwares: o *R-Project*² e o *Maple*³ para a realização de ajuste de curvas dos dados coletados empiricamente nos corredores do IFMG *campus* Formiga, mais especificamente sobre a planta dos andares do bloco C e bloco A.

3.1 A Metaheurística

Dada a complexidade do problema e o tamanho do ambiente simulado, para que fosse viável sugerir boas posições para a alocação do(s) *access point(s)* implementou-se o *Simulated Annealing* como metaheurística de otimização que teve como função objetivo maximizar a região coberta pela rede sem fio.

O *Simulated Annealing* é uma metaheurística para aproximar a otimização global em uma amplo espaço de busca. Este método foi proposto por Scoot Kirkpatrick em 1983 e foi utilizado para simular o processo de recozimento de metais cujo resfriamento rápido levava a produtos metaestáveis, ou seja, de maior energia interna e o esfriamento lento a produtos mais estáveis, estruturalmente fortes e de menor energia (LAARHOVEN; AARTS, 1987). Durante o recozimento, o material passa por vários estados possíveis com um tempo suficientemente longo para que qualquer elemento passe por todos os seus estados viáveis.

O *Simulated Annealing* realiza o processo de otimização buscando encontrar a melhor solução viável, considerando o objetivo do problema em questão, e o conjunto de restrições para aceitação da solução proposta. Na Figura 1 é possível ver de uma forma simples como é o funcionamento da metaheurística na busca de soluções para um sistema estável.

No eixo *x* temos o tempo gasto durante a busca, já no eixo *y* temos a tempe-

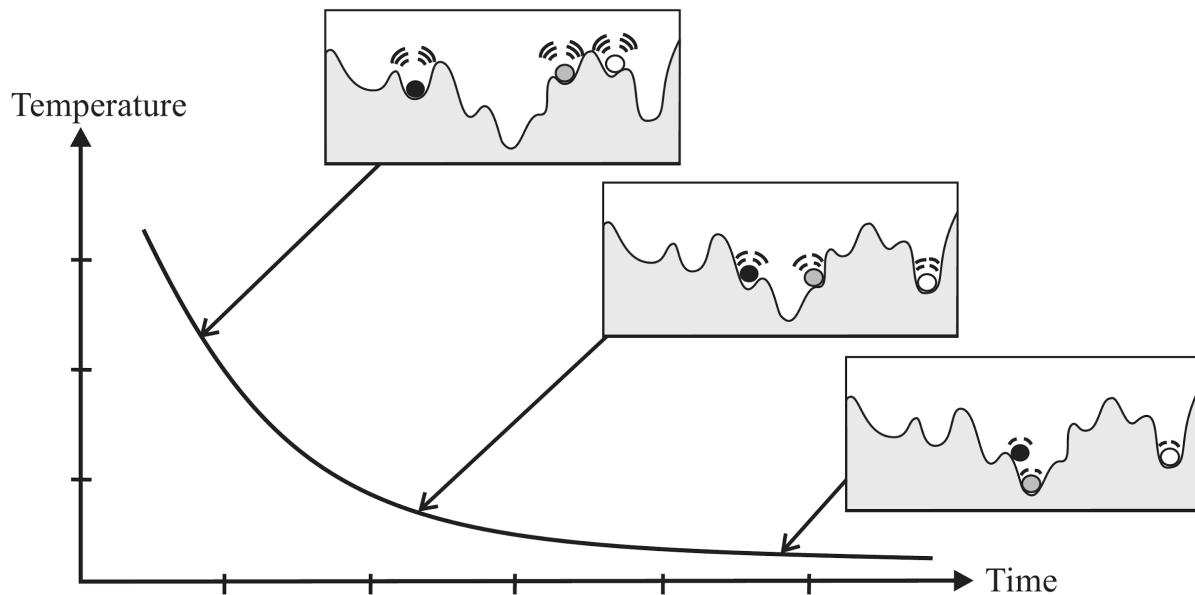
¹ <<https://www.autodesk.com.br/products/autocad/overview>>

² <<https://www.r-project.org/>>

³ <<https://www.maplesoft.com/products/Maple/>>

ratura que tende cair em função do tempo de busca. A metaheurística inicia com uma temperatura alta o que faz com que a probabilidade de aceitação de novas soluções seja também alta, evitando ótimos locais. No momento que a temperatura começa a cair, a chance de aceitar novas soluções diminui e o sistema tende a ficar cada vez mais estável obtendo uma solução definitiva.

Figura 1 – Comportamento do *Simulated Annealing* durante a exploração do espaço de busca.



Fonte: (LEDESMA; RUIZ; GARCIA, 2012).

Problemas no campo das heurísticas podem ser modelados como problemas de maximização e problemas de minimização de uma função objetivo, que neste caso é obter a maior cobertura e qualidade do sinal *wireless* na região informada.

Com um problema de otimização em mãos, encontrar soluções ótimas ou aproximadas do seu ótimo para problemas NP-difícil é um desafio nem sempre fácil de ser alcançado. O uso de heurística para auxiliar na busca por um lugar para o *access point* foi de fácil implementação e, como a maioria das heurísticas, produz boas soluções dentro de um tempo viável de acordo com os parâmetros estabelecidos.

O pseudocódigo da metaheurística com funções genéricas que foram implementadas neste trabalho é apresentado a seguir:

Estes são os identificadores utilizados:

- S_0 : Configuração Inicial (Entrada);

Algorithm 1 Simulated Annealing

```

1: procedure SIMULATEDANNEALING( $S_0, M, P, L, \alpha$ )
2:   /* Inicialização das variáveis */
3:    $S = S_0$ 
4:    $T_0 = TempInicial()$ 
5:    $T = T_0$ 
6:    $j = 1$ 
7:   /* Loop principal */
8:   /* Verifica se foram atendidas as condições de termino do algoritmo */
9:   repeat
10:     $i = 1$ 
11:     $nSucesso = 0$ 
12:    /* Loop Interno */
13:    /* Realização de perturbação em uma iteração */
14:    repeat
15:       $S_i = Perturba(S)$ 
16:       $\Delta_{Fi} = f(S_i) - f(S)$ 
17:      /*Teste de aceitação de uma nova solução*/
18:      if ( $\Delta_{Fi} \leq 0$ ) or ( $\exp(-\Delta_{Fi}/T) > Randomiza()$ ) then
19:         $S = S_i$ 
20:         $nSucesso = nSucesso + 1$ 
21:       $i = i + 1$ 
22:    until ( $nSucesso \geq L$ ) or ( $i > P$ )
23:     $T = \alpha \times T$ 
24:    /* Atualização do Contador de iterações */
25:     $j = j + 1$ 
26:  until ( $nSucesso = 0$ ) or ( $j > M$ )
27:  /* Saída do Algoritmo */
28:  return  $S$ 

```

- S_i : Configuração da Iteração i ;
- S : Configuração Final;
- T_0 : Temperatura Inicial;
- M : Número máximo de iterações (Entrada);
- P : Número máximo de Perturbações por iteração (Entrada);
- L : Número máximo de sucessos por iteração (Entrada);
- α : Fator de redução da temperatura (Entrada);
- $f(S_i)$: Valor da função objetivo correspondente à configuração S_i ;
- $nSucesso$: Contador de sucesso em uma iteração;
- i e j : Variáveis de controle de Loops.

Além dos indicadores acima, considerou-se as seguintes funções:

- *Perturba(S)*: Função que realiza uma perturbação na solução S ;
- *Randomiza()*: Função que gera um número aleatório uniforme no intervalo $[0, 1]$;
- *TempInicial()*: Função que calcula a temperatura inicial.

Mais adiante, é apresentado a descrição da metaheurística adaptada para o problema de otimização para alocação de *access point* nos ambientes do IFMG *campus Formiga*. Além disso, também serão apresentados técnicas e resultados de sua utilização.

3.2 Linguagem Python

A linguagem que foi utilizada para implementar todo o trabalho foi o Python. A linguagem Python é uma poderosa linguagem de programação e de fácil aprendizado. Surgiu no final dos anos 80 e foi criada por Guido Van Rossum. Na época, Guido trabalhava no Centro de Matemática e Ciência da Computação de Amsterdã, Holanda, no desenvolvimento de outras linguagens, quando percebeu que o desenvolvimento de utilitários para o sistema operacional Amoeba (projeto em que também trabalhava na época) utilizando a linguagem C estava tomando muito tempo e fazê-los em *Shell Script* não era viável. Precisava de algo que completasse o que cada linguagem deixava a desejar. Esses foram os principais motivos que fizeram com que o desenvolvimento do Python realmente tivesse início no ano de 1989 e nos primeiros meses de 1990 fosse a linguagem mais utilizada no departamento de Computação de Amsterdã e hoje por muitos desenvolvedores de *software* (SILVA, 2016).

O Python é uma linguagem interpretada. Isso significa que seu código é executado por um interpretador, e não compilado para linguagem de máquina para depois ser executada para um sistema e arquitetura específica, como acontece em algumas linguagens, como por exemplo a linguagem C (MAGNUN, 2014).

Não obstante, o Python não trabalha com tipagem de objetos, o que permite, no geral, um ótimo desempenho. Alguns processamentos que realizam a demanda de mais recursos, como o processamento de imagens, são feitos por bibliotecas que geralmente são escritas em C ou C++, inclusive com possíveis trechos em *assembly*, quando o alto desempenho é necessário (ROSSUM, 2009). Sendo assim, tais processamentos não fazem uso de tantos recursos num *script* escrito em Python. Em outra linguagem compilada seriam exigidos mais recursos.

Mesmo que o Python faça uso bem definido do tipo de dados que está manipulando, ele trabalha com tipagem dinâmica. Isso implica que uma variável possuirá características

cas de um tipo específico de sua declaração, até ser declarada novamente (MAGNUM, 2014). Pode-se ver então que, com o uso da tipagem dinâmica, são geradas flexibilidade e simplicidade no código de funções e classes do Python, reduzindo significativamente a quantidade de parâmetros em uma função.

A separação de blocos no Python não é feita com colchetes, chaves ou *begins* e *ends*. Toda a separação dos blocos é feita por tabulações. Isso força sempre o programador a manter o código mais organizado, além de reduzir consideravelmente o tamanho do código. Caso o programador que escreveu o código não mantenha a indentação, um erro de indentação é mostrado e ele não é executado.

Além dos tipos primitivos como *int*, *float*, *boolean*, também estão presentes no Python tipos especiais como listas, sendo listas  definidas entre colchetes e elementos mutáveis separados por vírgulas; tuplas sendo agora elementos imutáveis definidos entre parênteses, separados por vírgula e dicionários definidos entre chaves com tipos imutáveis variados. O conceito de Orientação a Objetos também está presente no Python, no qual todas as variáveis são definidas como objeto, até mesmo os tipos primitivos. Todos esses tipos especiais enriquecem ainda mais o poder que a linguagem tem (ROSSUM, 2009). O que em outra linguagem levaria tempo e linhas de código para ser feito, no Python é feito com simplicidade.

3.3 Bibliotecas utilizadas

Uma das vantagens de se usar o Python, é seu vasto número de bibliotecas embutidas e bibliotecas externas, as quais dão diversas facilidades ao resolver problemas simples do dia a dia até problemas mais complexos. Para o desenvolvimento do projeto foram utilizadas várias bibliotecas que estão embutidas no Python, que vieram complementar outras bibliotecas externas essenciais para o resultado esperado fosse obtido.

De início, foi utilizada a biblioteca *ezdxf*⁴. Tal biblioteca é um pacote do Python criado para manipular arquivos DXFs exportados do *AutoCad* independentemente de sua versão. Com ela, é possível criar, abrir, salvar e modificar arquivos com a extensão *.dxf* sem perder qualquer informação, podendo depois, ser utilizada em outros programas que utilizam a mesma extensão de arquivo. Tal biblioteca foi utilizada para a leitura da planta-baixa dos prédios e está presente desde o Python 2.7.

Outra biblioteca utilizada no projeto é a PyGame⁵. Seu desenvolvimento teve início no ano 2000 por Pete Shinners, sendo um código livre e é principalmente usada em aplicações que utilizam recursos multimídia e/ou programação gráfica, como em desenvol-

⁴ <<https://pypi.python.org/pypi/ezdxf>>

⁵ <<https://www.pygame.org/>>



vimento de jogos multiplataforma (independentemente do sistema operacional), utilizou-se para visualizar a solução gerada pelo *software*.

3.4 Programação Paralela

A computação paralela realizada por placas de vídeo é o uso de uma GPU juntamente com uma CPU para acelerar aplicações que necessitam de um alto recurso computacional (KIRK et al., 2007). Desde 2007, quando a NVIDIA criou o conceito de ‘computação acelerada’, o uso da GPU vem potencializando *data centers* de eficiência energética em laboratórios governamentais, universidades, corporações e empresas de médio e grande portes em todo o mundo. Elas desempenham um papel fundamental na aceleração de aplicações em plataformas que variam de inteligência artificial até carros, drones e robôs (NVIDIA, 2017b).

A seguir são descritos alguns conceitos e ferramentas que foram utilizadas para o desenvolvimento deste trabalho. Todas tiveram um papel essencial para que os objetivos propostos fossem alcançados.

3.4.1 CUDA

CUDA⁶ é um acrônimo de *Compute Unified Device Architecture*. É uma plataforma de computação paralela e um modelo de programação inventados pela NVIDIA (NVIDIA, 2017b). Tal plataforma faz com que as aplicações tenham aumentos significativos de desempenho computacional ao aproveitar a potência e os recursos da GPU⁷. Hoje há milhões de GPUs habilitadas para que trabalhem juntamente com CUDA. Muitos cientistas e pesquisadores vêm descobrindo inúmeras possibilidades para o uso da computação com GPU CUDA. Dentre as principais aplicações estão: a identificação de placas ocultas em artérias em pacientes com problemas de ataque cardíaco, análise do fluxo de tráfego aéreo junto ao *National Airspace System* (Sistema de Espaço Aéreo Nacional), aumento de desempenho para a simulação de visualização de moléculas (KIRK et al., 2007).

Há algum tempo era complicado escrever códigos para que fossem executados utilizando a GPU. Todo o assunto sobre programação paralela em GPU era focado em processamento de imagens; hoje esse conceito já não é mais restrito à placa de vídeo faz mais do que realizar o processamento de imagens; ela lida com um teraflop de desempenho de ponto flutuante e processa tarefas que vão de problemas de finanças à medicina.

Para que seja possível escrever um código que busque aproveitar o desempenho oferecido pelo *hardware* juntamente com a CUDA, será necessário utilizar o CUDA *Toolkit*,

⁶ <http://www.nvidia.com.br/object/cuda_home_new_br.html>

⁷ <<http://www.nvidia.com.br/object/what-is-gpu-computing-br.html>>

que oferece um ambiente de desenvolvimento abrangente para desenvolvedores C, C#, C++, Fortran, R e Python. O CUDA *Toolkit* inclui compilador, bibliotecas de matemática e ferramentas para depuração e otimização do desempenho do código (FARBER, 2011).

Hoje a NVIDIA oferece todas as ferramentas necessárias para o desenvolvimento utilizando seu *hardware*. Tanto a NVIDIA quanto a CUDA só tendem a crescer, à medida que cada vez mais empresas fornecem ferramentas, serviços e soluções. Além disso, áreas como a ciência e a medicina podem crescer ainda mais com suas pesquisas (NVIDIA, 2017a).

3.4.2 CUDA *Toolkit*

O CUDA Toolkit⁸ nada mais é que um *kit* de ferramenta disponibilizado pela NVIDIA. O *kit* fornece um ambiente de desenvolvimento para a criação de aplicações a serem otimizadas em uma GPU. Com o *kit* de ferramentas é possível otimizar e implementar códigos escritos utilizando CUDA em sistemas que aceitam a paralelização com GPU, *workstations*, centro de dados empresariais, plataforma baseadas em nuvem e super-computadores HPC (NVIDIA, 2017c). No pacote vem incluso bibliotecas otimizadas para GPU, ferramentas para depuração e otimização, um compilador C/C++ e uma biblioteca de tempo para ser usada nas aplicações que serão desenvolvidas.

As bibliotecas otimizadas possibilitam a otimização de vários tipos de aplicação, como por exemplo, aplicações que realizam cálculos matemáticos utilizando conceitos de álgebra linear, processamento de imagem e de vídeo, inteligência artificial e até análise de grafos. O que torna possível desenvolver aplicações personalizadas para diferentes linguagens é o uso da *API*. A *API* fornece um controle mais seguro e objetivo, especialmente sobre o contexto ou módulo que a aplicação está sendo desenvolvida. Uma chamada feita diretamente ao *kernel* é muito mais complexa de ser implementada, pois a configuração de execução e os parâmetros do *kernel* devem ser especificados com chamadas de funções muito bem explícitas. Outra grande vantagem da API é que seu uso é independente da linguagem.

A versão utilizada neste trabalho foi a CUDA *Toolkit* 9.0 e foi o que tornou possível a otimização do ritmo na sua versão final. O CUDA *Toolkit* foi instalado juntamente com os *drivers* de vídeo e é facilmente encontrado no site de desenvolvedores da NVIDIA.

3.4.3 Numba

O Numba⁹ é um compilador para *arrays* e funções numéricas em Python que proporciona a aceleração de módulos do código, escritos diretamente na linguagem Python.

⁸ <<https://developer.nvidia.com/cuda-toolkit>>

⁹ <<http://numba.pydata.org/>>

O Numba realiza a geração de código de máquina otimizado a partir do código escrito em `Python` puro, utilizando a infra-estrutura do compilador LLVM (instalado no momento da instalação do CUDA) (LAM; PITROU; SEIBERT, 2015). Com algumas anotações simples nos `comments` dos métodos, o código escrito utilizando vetores e matrizes pode ser otimizado em *just-in-time* com um desempenho semelhante ao C, C++ e Fortran, sem ter que alterar a linguagem de programação ou interpretador do Python.

Os principais pontos positivos da utilização do Numba são:

- geração de código *on-the-fly* (geração de código em tempo de execução);
- geração de código nativo para a CPU e *hardware GPU*;
- integração com a pilha de bibliotecas científicas do Python, graças ao Numpy.

3.4.4 JIT (*just-in-time*)

O JIT é uma anotação da biblioteca Numba utilizada como `@jit` acima do nome do método escrito em Python (RIGO, 2004). O código do método contido abaixo da `flag` será compilado em tempo de execução, utilizando o conceito *just-in-time*. O JIT possui a seguinte assinatura: `@jit(signature, nopython, nogil, cache, forceobj, locals)`. Os parâmetros não são obrigatórios, o que faz com que o programador `deixe` a cargo do compilador escolher as melhores configurações para a compilação. Uma explicação simples dos parâmetros podem ser vistos a seguir (NUMBA, 2017).

- ***signature*** é a assinatura do método pelo qual será gerado o código de máquina. A assinatura pode ser uma assinatura simples, sendo somente o tipo de retorno ou pode ser uma lista de assinaturas contendo cada tipo de dados dos parâmetros do método em questão. O tipo de retorno pode ser omitido, sendo mostrado apenas o tipo dos parâmetros e pode ser utilizado o tipo definido pelo Numba, por exemplo: `(numba.int32, numba.double)`. Caso queira deixar bem definido o tipo retorno, pode ser usado da seguinte forma: `numba.void (numba.int32, numba.double)`. Outra forma de definir é usando o nome simples do tipo dos dados, assim: `void(int32, double)`;
- ***nopython*** é um valor booleano quando seu valor for `true`; força o método a ser compilado no modo “*nopython*”. Neste modo de compilação, o Numba gera código que não acessa a API Python C. Este modo de compilação produz o código de desempenho mais alto, mas requer que os tipos nativos de todos os valores no método possam ser inferidos. Caso contrário, o `@jit` retornará automaticamente um erro se não puder ser usado;
- ***forceobj*** é um valor booleano quando seu valor for `true`. Força o método a ser compilado no modo objeto. Como o modo objeto é mais lento que o modo *nopython*,

isso é principalmente útil para fins de teste. Esta *flag* faz com que o modo de compilação Numba que gera o código que manipula todos os valores como objetos Python e usa a API Python C para executar todas as operações nesses objetos. O código compilado no modo objeto geralmente executará não mais rápido do que o código interpretado pelo Python, a menos que o compilador Numba possa aproveitar o mesmo código já compilado;

- ***nogil*** é um valor booleano que, quando seu valor for *true*, tenta liberar o bloqueio de interpretação global dentro do método compilada. O GIL (*global interpreter lock*) só será liberado se o Numba puder compilar a função no modo *nopython*, caso contrário, um aviso com erro de compilação será exibido;
- ***cache***, é um valor booleano que, quando seu valor for *true*, habilita um cache baseado em arquivo para encurtar os tempos de compilação quando a função já foi compilada em uma invocação anterior. O *cache* é mantido no subdiretório *__pycache__* do diretório que contém o arquivo de origem. Nem todas as funções podem ser armazenadas em cache, uma vez que algumas funcionalidades não podem ser sempre persistentes no disco. Quando um método não pode ser armazenado em cache, um aviso é emitido informando;
- ***locals***, é um dicionário que pode ser usado para forçar os tipos de dados Numba de variáveis locais particulares, por exemplo, se for necessário forçar o uso de variáveis de ponto flutuante com uma precisão específica. A documentação do Numba recomenda deixar o compilador definir os próprios tipos de variáveis.

Para a versão final do algoritmo deste trabalho, foi utilizada a anotação *@jit*, porém deixado com que o compilador escolhesse quais dados e forma de compilação usar. O conhecimento e estudo de cada parâmetro foi essencial para seguir com o desenvolvimento. Caso o programador decida definir qual a precisão utilizar e otimizar ainda mais o desempenho do código, é essencial um conhecimento mais aprofundado do compilador *just-in-time*.

3.4.5 Anaconda Navigator

A Anaconda¹⁰ é uma plataforma bastante popular de dados científicos para Python e é utilizada por 4,5 milhões de usuários em todo o mundo. Está sempre liderando projetos de código aberto como Numpy e SciPy, que atualmente formam a base de muitas aplicações no meio científico (NAVIGATOR, 2017). A Anaconda também pode ser utilizada e integrada com a linguagem utilizando o RStudio¹¹.

¹⁰ <<https://www.anaconda.com/>>

¹¹ <<https://www.rstudio.com/>>

A plataforma pode ser obtida através do *site* oficial. Após a instalação, se o usuário achar preferível usar uma interface gráfica, deve-se usar o *Navigator*; caso prefira usar o próprio terminal, basta usar o *conda*. É possível alternar entre eles (qualquer modificação feita em uma biblioteca será refletida no outro). O *Anaconda* acaba sendo a interface gráfica para o *conda*. É possível instalar, remover ou atualizar qualquer pacote científico da *Anaconda* com apenas alguns cliques, utilizando o *Navigator* ou apenas um único comando do *conda* no terminal. A versão do *conda* utilizada neste trabalho foi a versão 4.3.30 e a versão utilizada na interface gráfica, *Anaconda Navigator*, foi a 1.6.9.

De posse da metodologia desenvolvida neste capítulo, no capítulo seguinte é apresentado o projeto e desenvolvimento do *software*, desde a elaboração de dados, propagação do sinal, validação do modelo utilizado, à paralelização do algoritmo.

4 PROJETO E DESENVOLVIMENTO

Foram conduzidos estudos bibliográficos para compreensão dos modelos físicos de propagação de sinais *wireless* em pequena e grande escala, levando em conta as perdas de sinal e sua atenuação (ALMERS et al., 2007). Então foi feito um levantamento do estado da arte sobre simulação da propagação de sinais, tal como, por exemplo, a modelagem da propagação de sinais de rádio através de *Ray Tracing*¹ (YUN; ISKANDER, 2015), visando à definição da técnica a ser utilizada para a simulação da propagação dos sinais *wireless* (LENTZ, 2003; NAJNUDEL, 2004; SANDEEP et al., 2008). Uma vez definido o mecanismo de simulação, foi definida qual técnica de otimização seria aplicada para o problema da maximização da cobertura de sinal do(s) *access point(s)* *Wi-Fi*, bem como definida a função objetivo para avaliar as soluções propostas. Por fim, foi utilizado um método de visualização da intensidade do sinal *wireless* (RENSBURG; IRWIN, 2005; SULAIMAN et al., 2012) , por meio de um mapa de calor (*heat map*).

4.1 Representação do ambiente

Os modelos bidimensionais dos ambientes utilizados como caso de estudo foram o piso 2 do bloco A e os pisos 1, 2 e 3 do bloco C, onde atualmente os professores e alunos têm reclamado de baixa cobertura de sinal *Wi-Fi*. Especificamente, os ambientes da sala dos professores e da sala de estudos tem apresentado baixa cobertura de sinal *Wi-Fi*, com frequentes desconexões dos usuários que ali estão. Tipicamente, os pavimentos de edifícios e residências são representados através de *softwares CAD*, tais como o AutoCAD². Entretanto, *softwares CAD* tipicamente utilizam um formato proprietário (DWG no caso do AutoCAD) e como este trabalho se propõe a seguir a filosofia de *software livre* e de código-fonte aberto (FOSS³), optou-se por utilizar um formato de entrada interoperável chamado *DXF*⁴ (*Drawing Interchange Format*), que pode ser facilmente exportado a partir dos *softwares CAD* popularmente utilizados.

4.1.1 Arquivo *DXF*

O *software* desenvolvido neste trabalho deve receber como arquivo de entrada uma representação bidimensional do ambiente a ser simulado, esperando como tal um arquivo com extensão *DXF*. Arquivos *DXF* são comuns na exportação de desenhos e projetos de

¹ <[https://en.wikipedia.org/wiki/Ray_tracing_\(graphics\)](https://en.wikipedia.org/wiki/Ray_tracing_(graphics))>

² <<https://www.autodesk.com.br/products/autocad/overview>>

³ <<https://www.gnu.org/philosophy/floss-and-foss.html>>

⁴ <<https://www.loc.gov/preservation/digital/formats/fdd000446.shtml>>



peças para serem utilizadas em softwares de CAM e máquinas de corte automáticas. São populares por promoverem a troca aberta destes arquivos entre programas desenvolvidos por diferentes empresas⁵. Para interpretar o arquivo de entrada DXF, foi implementado um procedimento escrito na linguagem Python utilizando alguns recursos da biblioteca *ezdxf*, descrita na seção de materiais. Inicialmente, o procedimento realiza a leitura do arquivo DXF fornecido.

A partir do conteúdo do DXF, é obtido o *modelspace* no qual é possível buscar os valores (x, y) extremos do modelo 2D. Isto se faz necessário para que tais extremos sejam subtraídos no momento da leitura das paredes do edifício, a fim de evitar desperdício de processamento com as margens da planta-baixa e para que o desenho do ambiente possa se enquadrar perfeitamente na tela de visualização do mesmo. Tanto na busca dos valores limítrofes de x e y , quanto na leitura das linhas do modelo DXF, é feita uma busca em seu *modelspace* utilizando um laço de repetição. Na busca, é verificado o tipo de dados no *modelspace* é do tipo *LINE* (linha) e se a camada percorrida é do tipo *ARQ* (arquitetura). Durante a verificação de linhas na camada *ARQ*, são adicionados a uma lista de paredes as coordenadas do ponto inicial (x_1, y_1) e ponto final (x_2, y_2) de cada linha representada no arquivo DXF, pois cada linha na camada ARQ do modelo representa uma parede. Observe que cada coordenada (x, y) deve ser devidamente multiplicada pela escala definida pelo usuário, para que seja mantida a proporção entre o tamanho da matriz de simulação e o ambiente real simulado. Ao final do processamento do arquivo DXF, os valores máximos (e mínimos) das coordenadas (x, y) , bem como cada tupla de coordenadas das paredes é armazenada em memória pelo software, para que sejam utilizados pela heurística.

4.1.2 Escala e precisão

Observe que, a partir dos valores (x, y) limítrofes obtém-se as dimensões do modelo bidimensional e, desde que esteja disponível a informação sobre o tamanho real do edifício, pode-se calcular a escala da planta-baixa para o tamanho real do edifício. De posse das dimensões da planta-baixa e do edifício, pode-se representá-lo computacionalmente como um arranjo bidimensional, ou seja, uma matriz de tamanho Comprimento X Largura, podendo ser utilizada uma escala 1:1 do ambiente simulado para o ambiente real. Neste caso, uma célula da matriz representaria 1 m^2 no mundo real.

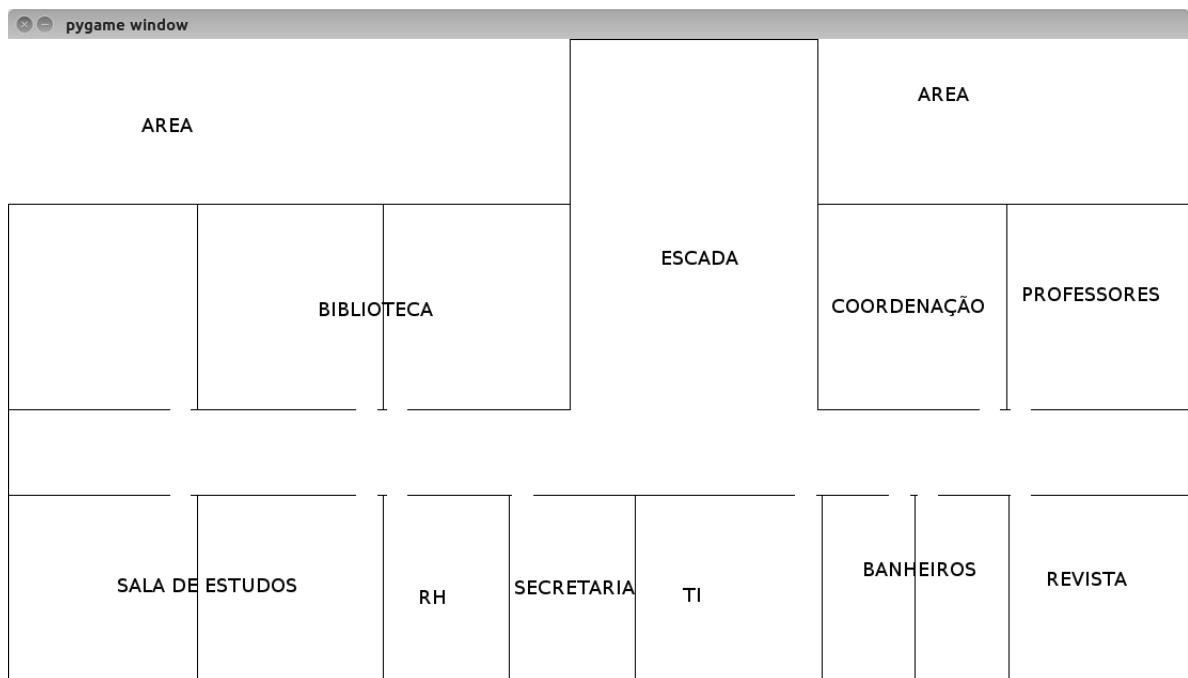
Entretanto, vale ressaltar que o software produzido é capaz de trabalhar com diferentes níveis de precisão na simulação de propagação de sinais, onde caso o usuário queira uma precisão de 0,5 m^2 por célula da matriz, bastaria informar tal configuração de forma que o software passe a utilizar um fator de escala de 2.0x a dimensão real

⁵ Disponível em <<https://www.otimizenesting.com.br/single-post/2016/09/29/Arquivo-DXF>>. Acesso em 29 out. 2017.

do edifício (pois precisão = 1/escala). Assim, o *software* pode trabalhar com a precisão desejada pelo usuário, podendo inclusive realizar uma simulação de propagação de ondas onde cada célula da matriz tenha a dimensão de um comprimento de onda ($\lambda = c/f$). Por exemplo, se o usuário desejar ter a precisão de 1 cm de comprimento de onda para simular a propagação de Wi-Fi na frequência de 2,4 GHz, cada célula representará⁶ uma área de $12,5 \text{ cm}^2$. Entretanto, vale ressaltar que quanto maior a precisão da simulação, maior serão as dimensões da matriz que representa o ambiente e, portanto, mais demorada será a simulação computacional.

As plantas-baixas representando os ambientes simulados neste trabalho foram gentilmente cedidas pela equipe de TI do *campus* Formiga (bloco A) e pelo setor de engenharia civil do *campus* Formiga (no caso dos blocos B e C). A Figura 2 ilustra o ambiente do bloco A do *campus* Formiga, tendo sido carregado automaticamente a partir do arquivo *DXF*. O texto descrevendo no que consiste cada sala do ambiente foi adicionado posteriormente à captura da imagem, para melhor compreensão e localização do leitor.

Figura 2 – Representação do ambiente a partir do arquivo *DXF* contendo a planta-baixa.



Fonte: Elaborado pelo autor.

Para a visualização dos resultados dos testes e uma melhor interpretação dos resultados, foi utilizado o PyGame para prover a visualização gráfica da matriz contendo os resultados dos cálculos de intensidade de sinal em cada ponto do modelo simulado, con-

⁶ $2.998 \times 10^8 \text{ m/s} \div 2.437 \text{ GHz} < \text{https://www.wolframalpha.com/input/?i=c+2.437+GHz}>$

forme proposto pelo modelo de propagação utilizado. Após a representação da matriz de propagação no PyGame, podem ser desenhadas também as paredes do ambiente simulado de acordo com a lista de linhas obtidas no processamento do arquivo *DXF*. A maneira como a representação visual do ambiente simulado foi realizada será tratada em detalhes em seção posterior.

4.2 Propagação dos sinais

Conforme assunto abordado na seção 2.4, se faz necessário uma boa escolha de qual modelo de propagação de sinais utilizar, para obter uma boa representação da intensidade dos sinais para aquela banda do espectro eletromagnético e, assim, simular algo mais próximo da realidade. Quanto maior for a precisão desejada para o valor da intensidade de sinal devido a perdas e sua atenuação, mais detalhes sobre o ambiente de propagação devem ser modelados (frequências, distâncias, paredes, pisos, materiais, etc).

4.2.1 Definição do modelo de propagação para sinais *Wi-Fi*

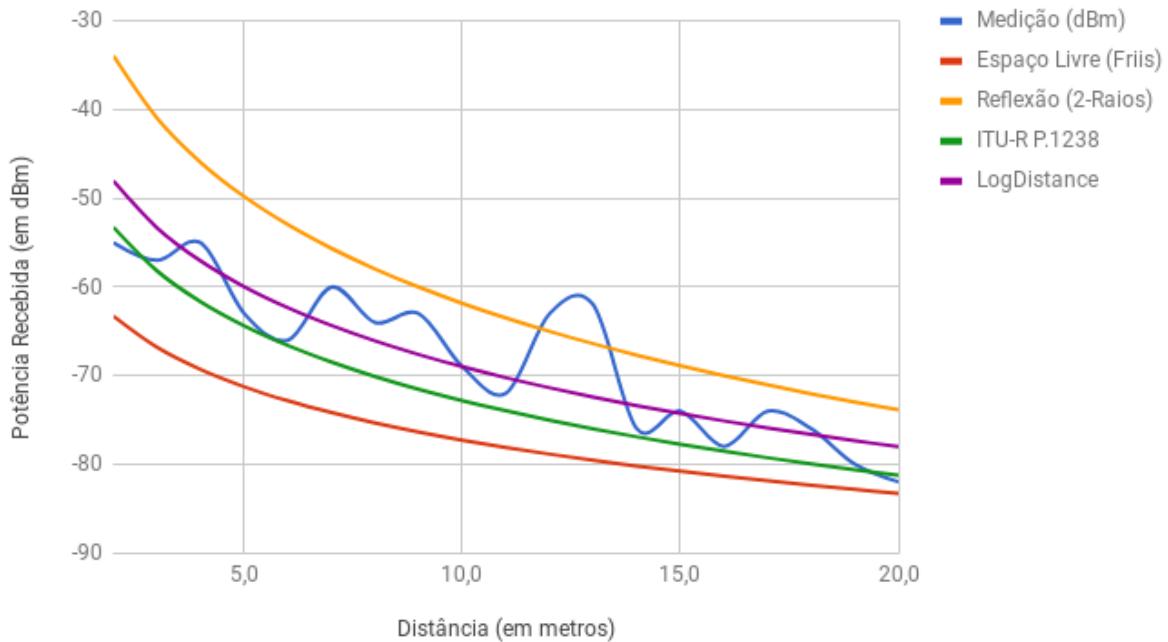
Foram conduzidos nas dependências do *campus* Formiga experimentos para medição do decaimento da intensidade do sinal *Wi-Fi*. Utilizou-se medições *indoor* nos corredores do *campus*. Para tal, foi posicionado no corredor um *access point* Cisco WAP200 (IEEE 802.11b/g) e configurado para utilizar um canal *Wi-Fi* que correspondesse a uma frequência que, naquele momento, estivesse livre de interferência de ~~outros~~ *access points* naquela região do *campus*. A potência máxima de transmissão que o *access point* suporta é -14 dBm (0,0398 mW), mas ele foi configurado para utilizar apenas 25% dessa potência. Tal configuração visou apenas reduzir a distância a ser percorrida na condução do teste, uma vez que ~~as~~ paredes e pisos absorvem ~~mais~~ ~~menos~~ o sinal de acordo com a frequência dele e ~~a~~ ~~intensidade~~, portanto ~~mantive-se~~ o comportamento esperado na propagação do sinal. O *access point* WAP200 utilizado como transmissor possui antenas com 2 dB_i de ganho e a interface *Wi-Fi* do *notebook* utilizado como receptor nos testes possui ganho de 1 dB_i. Foi utilizado o canal 11 (portanto com frequência de 2,462 GHz e λ de 12,18 cm), tendo sido realizadas medições após o campo distante da antena (campo de *Fraunhofer* = 148 cm).

As medições foram coletadas utilizando o comando *iwlist* da biblioteca *iw-utils*, que realiza a varredura (*scanning*) da faixa de espectro correspondente ao *Wi-Fi* e registra os valores de intensidade de sinal recebidos bem como o endereço MAC do(s) *access point*(s). A Figura 3 ilustra quão bem cada modelo de propagação considerado previu o valor da intensidade de sinal de acordo com o aumento da distância. Pela figura, observe que o modelo de reflexão de dois raios, por definição e de acordo com a altura das antenas (aproximadamente 90 cm, considerando também o suporte utilizado), seria apropriado

somente se as distâncias entre transmissor e receptor fossem superiores a 83,59 m, faixa demasiado próxima do limite de alcance de um enlace *Wi-Fi* típico. Também, observe que com o modelo de propagação no espaço livre (modelo de *Friis*), as estimativas foram abaixo do valor real observado, por tal modelo não considerar os fenômenos de reflexão, difração e dispersão que tipicamente ocorrem na propagação de ondas em um ambiente geometricamente complexo (paredes, piso, teto, portas, janelas, mobília, etc).

Figura 3 – Medição da intensidade de sinal *vs.* modelos de sua propagação.

Medição VS Modelos de propagação



Fonte: Elaborado pelo autor.

Pela Figura 3 é possível observar que, dos modelos de propagação analisados, o *LogDistance Path Loss* apresentou um boa concordância entre os valores de suas estimativas e os valores reais mensurados, sendo portanto considerado como o mais promissor. Entretanto, observe que a Figura 3 apresenta ainda apenas uma das várias medições realizadas, ilustrando portanto o decaimento exponencial da intensidade do sinal *Wi-Fi* ao longo de um corredor do bloco B. Neste caso, o modelo *LogDistance* foi calibrado para uma distância de referência (d_0) de 10 metros com a respectiva perda (PL_0) mensurada em -69 dBm , com expoente de perda de caminho manualmente ajustado como $\gamma = 3$ (*path loss exponent*).

4.2.2 Ajuste do modelo de propagação

Não pode-se basear nossa simulação em parâmetros de um modelo de configuração criado para medições coletadas em *apenas um corredor de um pavimento de um prédio campus** Formiga. *Se faz necessário calibrar o modelo de propagação para que seja mais abrangente para o campus**, devendo ser calibrado de acordo com múltiplas medições realizadas em diversas salas, corredores e áreas internas de nosso caso de estudo.

Para tal, as informações de distância em função da intensidade das várias medições conduzidas foram utilizadas como entrada para um ajuste de curvas estatístico, técnica de regressão que foi conduzida no software *R-Project*⁷. Inicialmente, realizou-se regressões com um ajuste de curvas estatístico utilizando os pacotes *fitdistrplus*⁸ e *FAdist*⁹ do software *R-Project*.

Pela comparação do ajuste de curvas ilustrado na Figura 4, observou-se que a distribuição Logística com apenas 2 parâmetros (logis) não consegue modelar bem o decaimento do nível de sinal observado na medições do RSSI no *campus* Formiga. Por outro lado, as distribuições 3P-LogNormal (lnorm3) e 3P-LogLogistica (llog3) se aproximaram bem do comportamento esperado para o decaimento da intensidade do sinal, estando tal fato em consonância com o exposto nas seções 2.4.6 (*Log-normal fading*) e 2.4.3 (*Log-distance*).

A Figura 5 e a Figura 6 apresentam em detalhes a qualidade do ajuste de curvas estatístico, obtido através das regressões realizadas. Ao comparar o GOF (*Goodness-of-fit*) entre duas ou mais distribuições, pode-se considerar os critérios de qualidade do ajuste AIC (*Akaike's Information Criterion*) e BIC (*Bayesian Information Criterion*).

Considerando o melhor ajuste das distribuições *LogNormal* e *LogLogistica*, ambas com 3 parâmetros, passou-se a considerar uma regressão estatística com distribuições estatísticas mais complexas, de 4 ou mais parâmetros, mas mantendo o comportamento logarítmico. Uma regressão **NP-Logística** foi conduzida com a utilização do pacote *nplr*, também no software *R-Project*.

Para obtermos um melhor resultado, considerando o bom desempenho das distribuições *LogNormal* e *LogLogistica*, instruímos o modelo de regressão Logístico a utilizar uma base logarítmica para os valores do eixo x , ou seja, a distância ao *access point* deve ser interpretada como $\log_{10}(x)$. Em uma livre interpretação, isso corresponderia a instruir a regressão logística a utilizar modelos NP-Logísticos. Assim, buscou-se partir dos bons resultados obtidos na análise prévia, mas tentando melhorá-los com a inclusão de um quarto ou quinto parâmetros de ajuste da distribuição. Obteu-se um melhor *fitness* com um modelo Logístico de 4 parâmetros (*4-Parameter Logistic* ou simplesmente 4PL),

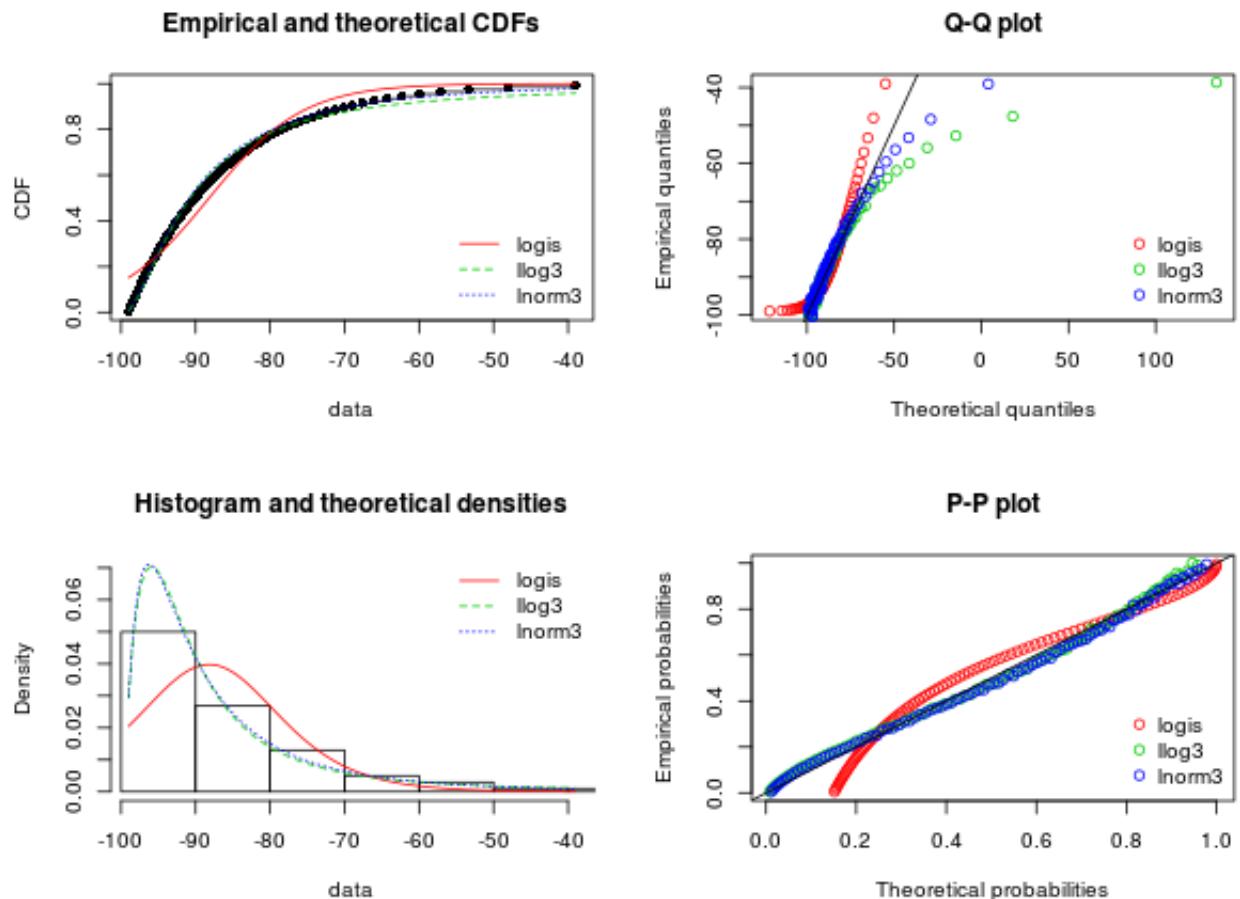
⁷ <<https://www.r-project.org/>>

⁸ <<https://cran.r-project.org/web/packages/fitdistrplus/index.html>>

⁹ <<https://cran.r-project.org/web/packages/FAdist/index.html>>



Figura 4 – Ajuste de curvas da 2P-Logística, 3P-LogNormal e 3P-LogLogística.



Fonte: Elaborado pelo autor.

conforme resultados apresentados a seguir. Conforme figura a e a , o melhor ajuste de curva (GOF) foi obtido com uma distânci ação 4P-Logística e com a distância fornecida em em escala logarítmica, como $\log_{10}(x)$ metros.

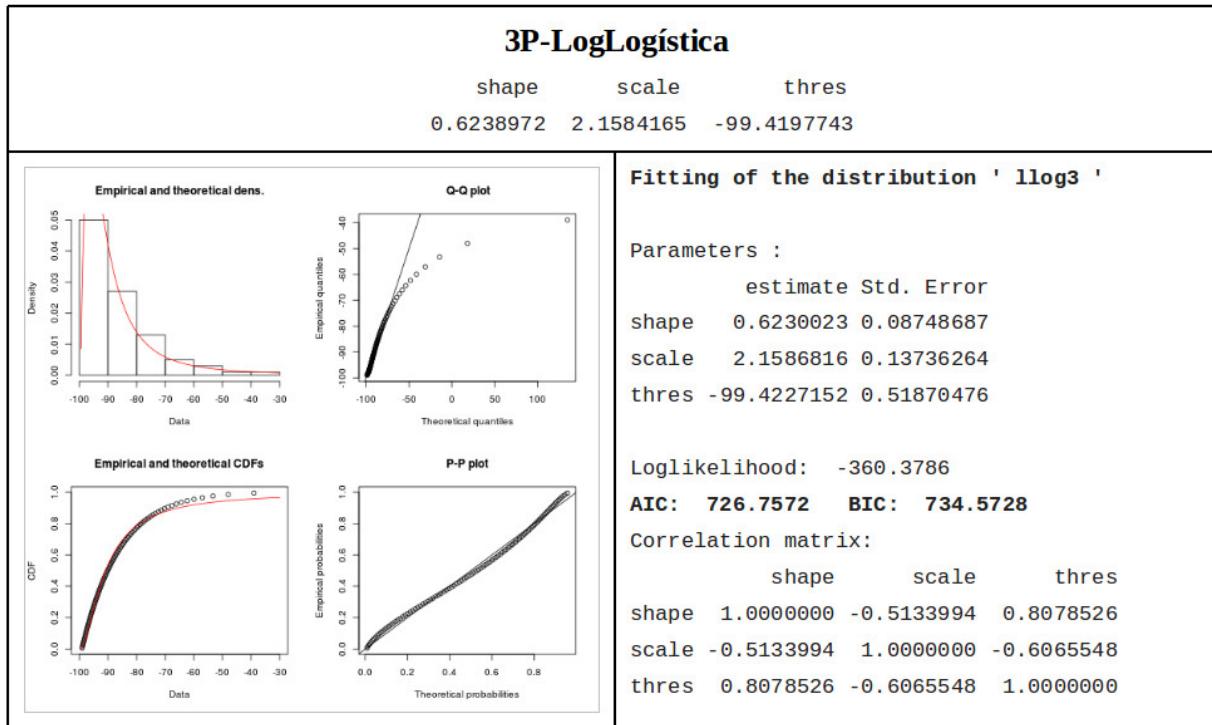
En m, a partir dos parâmetros da 4P-Logística (4PL ou logis4) os pela regressão, podemos configurar as constantes de sua equação e, a partir dela, obter a estimativa de decaimento do nível de sinal com o aumento da distância do access point, calibradas para a realidade do campus Formiga. A equação característica de uma NP-Logística de cinco parâmetros é,

$$f(x; A, B, C, D, E) = D + (A - D)/((1 + (x/C)^B)^E), \quad (4.1)$$

na qual os parâmetros A, B, C, D e E são obtidos ao aplicar-se a regressão logística para os valores de intensidade de sinal (em dBm ou mW). Observe que, fixando “E = 1”



Figura 5 – Qualidade do ajuste (GOF) da distribuição 3P-LogLogística.



Fonte: Elaborado pelo autor.

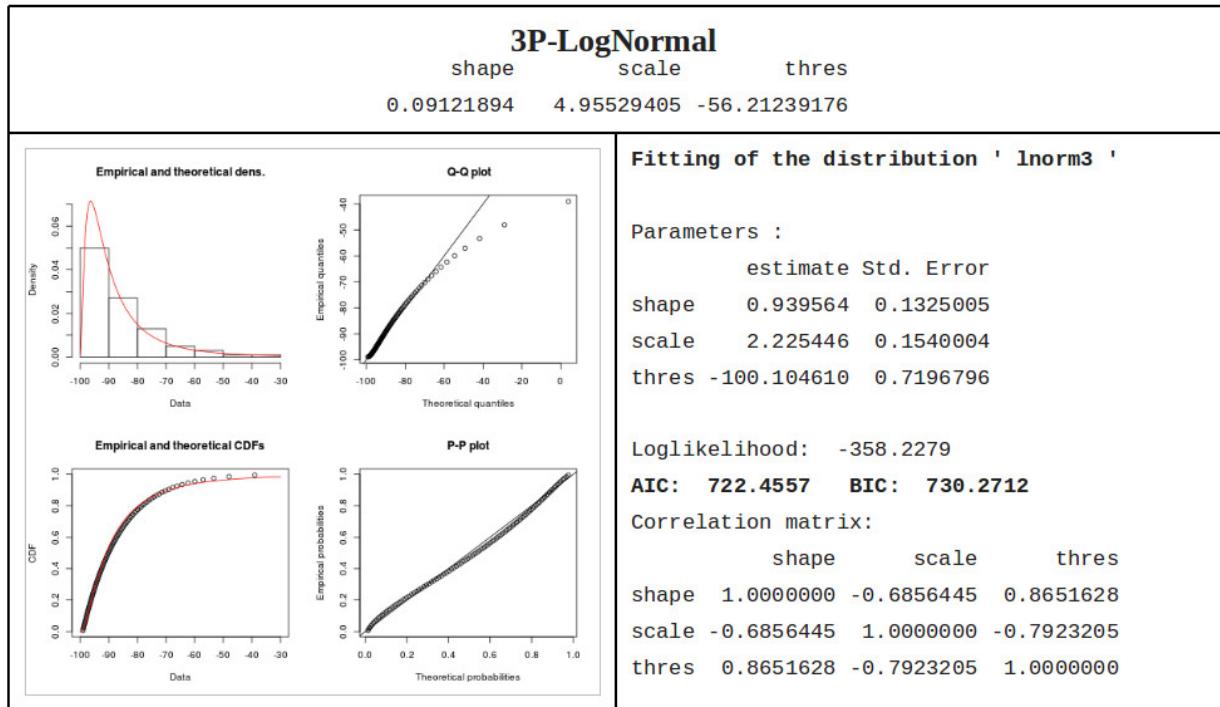


obtem-se a equação da 4PL (*4-Parameter Logistic*) e se fixa os dois “ $D = 0$ ” (referência), obtem-se a equação da 3PL (*3-Parameter Logistic*). Os cinco parâmetros da *llog* tem o seguinte significado:

- A = Mínima assintótica. Em medições da propagação de sinais de rádio, considere-o como o valor do sinal para uma distância de referência. Em propagação de sinais de rádio, tipicamente é o valor do sinal a 10 metros do transmissor. No caso do *Wi-Fi*, é um valor de referência aferido após o campo distante da antena (distância de *Far hoffe*), ou seja, intensidade do sinal aferida a 2 metros (2,4 GHz) ou 4 metros (5,8 GHz);
- B = Inclinação da curva, que pode ser positiva ou negativa. No caso de propagação de sinais de rádio, para obter um decaimento do sinal provavelmente B será positivo;
- C = Ponto de inflexão, definido como o ponto onde a curvatura muda de direção ou sinal. C é a concentração onde $y = \frac{(D-A)}{2}$. No caso do *Wi-Fi*, provavelmente será o valor do sinal aferido em torno 10 a 20 metros;
- D = Máxima assintótica. Em medições da propagação de sinais de rádio, considere-o como o valor do sinal para uma distância muito grande. No caso do *Wi-Fi*, pode-se



Figura 6 – Qualidade do ajuste (GOF) da distribuição 3P-LogNormal.



Fonte: Elaborado pelo autor.

utilizar um valor de referência aferido a 100 metros (limite da WLAN);

- E = Fator de assimetria. Para E=1, tem-se uma curva simétrica em torno do ponto de inflexão e, portanto, uma equação logística de quatro parâmetros.

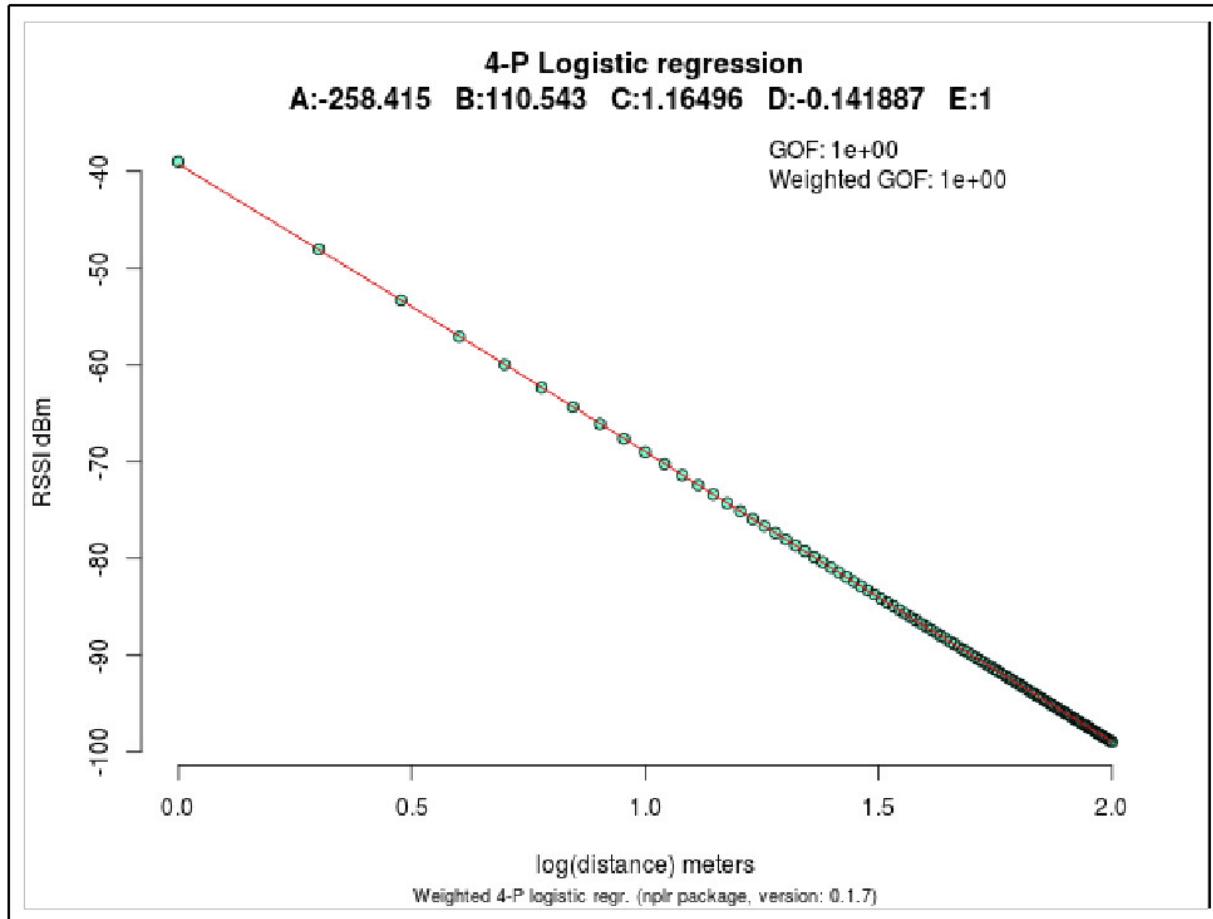
Para utilizar tal distribuição estatística como modelo de propagação, deve-se implementar no *software* sua **função de densidade de probabilidade** (PDF) para que possa retornar qual seria o valor da intensidade do sinal recebido (RSSI) em determinada distância do *access point Wi-Fi*. Durante a implementação, vale observar que ambas as distribuições *4P-Log* e *LogDistance* devem ser utilizadas como um modelo de **perda de sinal**, ou seja, o valor da estimativa obtido (*PL*) deve ser subtraído da potência de transmissão (*P_t*) do *access point* para obter a previsão da potência recebida (*P_r*):

$$Pr(x) = P_t - PL(x), \quad (4.2)$$

onde *P_r* é a potência recebida à distância *x* do transmissor; *P_t* é a potência do sinal do transmissor (ex.: -20 dBm) e *PL* é a perda do sinal ao longo do caminho entre o transmissor e receptor (calculada com a equação da 4PL ou da *LogDistance*). Inicialmente, implementou-se tais equações em uma planilha para visualizarmos graficamente



Figura 7 – Previsão de RSSI da 4P-LogLogística para distâncias em $\log_{10}(x)$ metros.



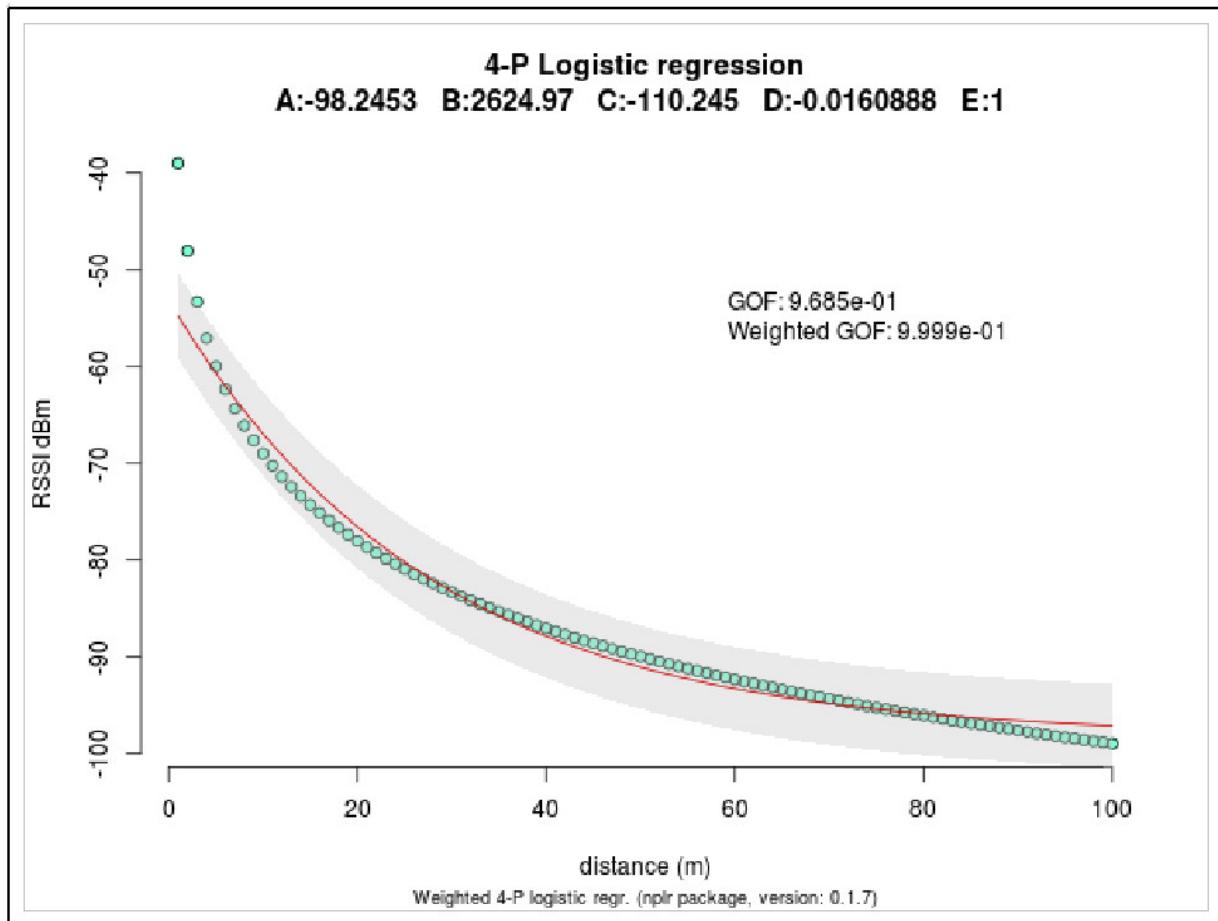
Fonte: Elaborado pelo autor.

uma comparação das medições de RSSI realizadas no campo com a expectativa de decaimento proposto pelos modelos *LogDistance* e *4P Logístico*. A Figura 9 ilustra o decaimento se real previsto por cada um dos modelos acima citados em relação aos valores reais de medição coletados.

Já na Figura 9 (b), é apresentada a extrapolação que os modelos de propagação de fornecem para distâncias de até 100 metros (limite de alcance de um sinal *Wi-Fi indoor*). Comparando as Figura 9 (a) e (b), é interessante notar que os dois modelos diferiram mais para curtas distâncias, escala onde acontece muita variação do nível de sinal devido aos efeitos de múltiplas reflexões nas paredes, difração nas portas, dispersão e absorção pelos materiais. Isto está plenamente de acordo com a abordagem do assunto pela literatura, no que tange a modelos de propagação de pequena e larga escala (RAPPAPORT, 2009).



Figura 8 – Previsão de RSSI da 4P-Logística para distâncias em metros.



Fonte: Elaborado pelo autor.

4.2.3 Simulação da propagação de sinais no ambiente

Como dito anteriormente, para a realização dos cálculos, o ambiente foi representado como uma matriz bidimensional, representando em escala o ambiente a ser simulado. Cada posição $[x, y]$ da matriz é um ponto candidato a hospedar o *access point* (AP). Uma vez determinada a nova localização do *access point*, para cada posição da matriz é necessário realizar o cálculo do modelo de propagação. Na configuração do software produzido, pode-se informar a posição atual do *access point* no ambiente real, ou deixar que o software realize a exploração a partir de uma posição aleatória. Para preencher os valores da matriz representando a propagação do sinal de microondas, deve-se realizar o cálculo determinado pelo modelo de propagação, conforme tratado na seção anterior. Tipicamente, como os modelos de propagação fazem uso da distância entre o transmissor e receptor, esta deve ser calculada utilizando as coordenadas do *access point* na simulação e do ponto para o qual deseja-se calcular a estimativa da intensidade do sinal ali recebido.

(RSSI).

A Figura 10 exibe uma representação gráfica da matriz com os valores de intensidade de sinal conforme o decaimento exponencial da intensidade do sinal recebido (RSSI), com o aumento da distância de cada ponto da matriz com o *access point Wi-Fi* (ponto central).

Neste ponto, vale notar que o escopo deste trabalho não considera interferências entre canais *Wi-Fi* de vários *access points*, de maneira que para cada *access point* simulado é gerada uma matriz numérica individual, que receberá as estimativas de valores para a intensidade de sinal em Decibel *miliWatt* (*dBm*). O propósito deste trabalho é sugerir um melhor posicionamento dos *access points* e a decisão de não terem sido modeladas interferências entre *access points* diferentes é baseada no fato de que a tecnologia *Wi-Fi* possibilita coexistência de alguns *access points* em um mesmo ambiente, desde que o canal central de cada *access point* esteja suficientemente espaçado dos demais. Por exemplo, nos padrões IEEE 802.11b/g/n a largura de banda é de 20 MHz e portanto podem coexistir até quatro *access points* em um mesmo ambiente, conforme ilustra a Figura 11.

4.2.4 Atenuação do sinal ao atravessar paredes

Para obter um resultado que fosse condizente com a realidade, ou seja, que representasse quanto o sinal *Wi-Fi* degradava à medida que se propagava através de cômodos em sequência, foi adicionada uma técnica para realizar estimativas da absorção do sinal ao atravessar paredes. Na planta-baixa do edifício, todas as linhas (que representam paredes e divisórias) são armazenadas em uma estrutura de dados no início do programa, antes de iniciar a simulação. Tais linhas se mostraram úteis para simular de maneira mais próxima da realidade o valor da intensidade do sinal em determinado ponto, atravessando o ambiente simulado desde o *access point* até tal ponto.

Segui-se a relagem utilizada pela literatura para modelos de propagação em pequena escala, onde o valor esperado para o sinal em determinado ponto é subtraído do valor em *dB* que representa a energia absorvida por parede multiplicado pela quantidade de paredes atravessadas pelo sinal. A literatura sugere que o valor de absorção varia entre 8 e 13 para paredes de concreto (RAPPAPORT, 2009), de maneira que o valor definido para uso do *software* produzido na simulação do *campus Formiga* foi de 8 *dB* (e de acordo com análise experimental). Vale observar que, por consistir em uma constante que pode ser configurada no *software*, a ordem de magnitude da absorção de energia pelas paredes do edifício pode ser ajustada para que a simulação da propagação possa se aproximar da realidade daquele ambiente.

Para ilustrar o resultado de aplicar tal fator de correção, observe na Figura 12 a mesma simulação da figura anterior, porém agora contabilizando a absorção do sinal

Wi-Fi a cada parede atravessada no trajeto do *access point* ao ponto destino.

Para ter uma noção de quantas paredes o sinal sofreria atenuação, foram utilizadas equações de geometria euclidiana.  Momento em que se faz necessário o cálculo da absorção das paredes para um ponto $[x, y]$ qualquer da matriz, a lista de linhas representando as paredes da planta-baixa deve ser percorrida. A técnica utilizada foi a intersecção de retas num espaço cartesiano (*Line-Line Intersection*), buscando definir se os dois segmentos de reta  possuem um ponto em comum (ponto de interseção). A definição de interseção de retas é dada por retas concorrentes,  seja, se possuírem um ponto em comum, então a intersecção das duas retas será este ponto em comum. Assim, viabiliza-se verificar o cruzamento do sinal com paredes a um baixo custo computacional, se comparado a testar por força bruta cada ponto interno ao retângulo formado pela parede.

Os segmentos de reta candidatos consistem nas linhas previamente obtidas do arquivo *DXF*, contendo a planta-baixa. Cada linha no *DXF* possui duas coordenadas bidimensionais, para o ponto inicial e o ponto final do segmento de reta. O outro segmento de reta é formado entre o *access point* e o ponto para o qual está sendo calculada a intensidade do sinal. A Figura 13, ilustra o segmento de reta formado entre um dado ponto (x_1, y_1) na matriz e o local do *access point* (x_2, y_2) , cruzando com um outro segmento de reta formado pelas extremidades da parede (x_3, y_3) e (x_4, y_4) . Caso haja ponto de interseção (x, y) significa que o sinal terá atravessado tal parede e soma-se um à quantidade de paredes atravessadas. Caso contrário, se não houver ponto de interseção entre os dois segmentos de reta, significa que o sinal cruzou livremente o espaço entre o *access point* e o destino sem cruzar com tal parede.

Entretanto, apesar de tal verificação ser menor aceita para um ponto e as paredes candidatas, deve-se lembrar que é um cálculo a mais a ser efetuado para cada um dos $N \times M$ pontos da matriz para cada uma das K paredes, ou seja, $N \times M \times K$ cálculos a serem efetuados e, portanto, com custo de tempo da ordem de $O(n^3)$. E como o algoritmo de otimização irá avaliar centenas de posicionamentos para o *access point*, à medida que explora o espaço de soluções, o custo computacional sobre para $N \times M \times K \times P$, ou seja, como todas estas quantidades possuem uma mesma ordem de magnitude (centenas), o algoritmo passa a ter custo temporal de $O(n^4)$. Assim, com a adição dessa nova modelagem na propagação do sinal *Wi-Fi*, o tempo de execução do algoritmo extrapolou os costumeiros segundos e beirou as dezenas de minutos, com isso, tem-se um cenário propício para buscar uma forma de otimizar esta solução. A seguir, será apresentado o processo utilizado para a visualização dos dados.

4.3 Visualização dos dados

A princípio, foi implementada a etapa que trata a representação do ambiente, ao utilizar a matriz de propagação para a geração do modelo 2D correspondente à planta-baixa fornecida (KASE et al., 2005; THAKUR; BANERJEE; GUPTA, 2009). Tal modelo pode representar algumas características físicas do ambiente, tais como paredes e portas (MARSCHALLINGER, 1996), no qual o sinal pode atenuar em um maior ou menor grau sua intensidade de propagação. A Figura 4 ilustra uma representação gráfica das características da planta baixa do bloco A¹⁰, onde pode-se observar a presença de portas e paredes. Foi registrado o uso que cada cômodo possui atualmente no *campus* Formiga.

Com o preenchimento da matriz de propagação de acordo com a posição do *access point* e com o respectivo valor da intensidade do sinal, se tornou difícil a visualização de quanto o sinal ficava a medida que a distância aumenta e mais paredes eram atravessadas, através de uma simples inspeção numérica. Tendo a matriz preenchida com a intensidade de sinal (*dB*), suficiente para a realização dos cálculos com a função objetivo da heurística de otimização, fica praticamente a visualização a olho nu, não sendo viável distinguir dentre milhares de valores numa matriz, se pudesse ser considerada uma simulação boa ou ruim.

Considerando a vasta quantidade de ferramentas e bibliotecas disponibilizadas pela linguagem Python, foi necessário utilizar um método para transformar a matriz de intensidade de sinais em algo um tanto quanto visível e de fácil interpretação. Diante disso, o PyGame foi utilizado na visualização da matriz resultante. Com a obtenção dos valores máximos e mínimos dentro da matriz de propagação, é fácil definir uma faixa de valores para calibrar uma escala de referência (gradiente de cores) para a exposição dos resultados. Para padronizar e viabilizar a comparação visual de resultados diferentes, o gradiente de cores foi normalizado para um valor mínimo de referência de -100 *dB*, referente à maior sensibilidade registrada pelos equipamentos de comunicação *Wi-Fi*. Esta visualização dos resultados pode ser vista na Figura 15.

Foi implementado um método capaz de explorar os recursos disponibilizados pelo PyGame para ilustrar a matriz resultante. Tal método, percorre a matriz obtendo o valor calculado em *dBm* da posição $[x, y]$ e mapeia uma cor de acordo com o gradiente de cores escolhido. Adicionalmente, foi utilizada condição que avalia o valor do sinal em determinado ponto com uma constante que representa a sensibilidade máxima definida pelos equipamentos de *Wi-Fi*. Os valores de sensibilidade dos diversos equipamentos presentes no *campus* Formiga do IFMG variam entre -85 e -100 *dB*. Assim, o valor calculado para a situação da propagação for menor que a sensibilidade do *access point*, na visualização do mesmo ele é considerado como um “ponto cego” ou uma “região de sombra”, uma vez

¹⁰ A modelagem de ambiente utilizada para as simulações apresenta as mesmas divisões de cômodos do arquivo AutoCAD disponibilizado para o este trabalho.

que não há intensidade de sinal suficiente para obter uma boa qualidade de serviço (ou sequer conectar na rede *Wi-Fi*). A Figura 16 ilustra as regiões de sombra no ambiente simulado com apenas um *access point* centralizado transmitindo com uma potência de -17 dB.

O algoritmo foi desenvolvido de tal forma que seja possível durante a exploração do espaço de soluções pelo *Simulated Annealing*, exibir a matriz de intensidade de sinal a cada iteração para acompanhar o funcionamento do algoritmo, comportando-se como uma animação em tempo real. Entretanto, como o desenho dos milhares de pontos pelo PyGame consiste em uma operação que consome alguns segundos, tal processo de animação fica desativado por padrão para não retardar o cômputo da solução final que o usuário aguarda.

4.4 Heurística de otimização

Para o cumprimento dos objetivos e a obtenção de um bom resultado, foi implementada a metaheurística, que irá explorar o espaço de soluções de posicionamento dos *access points* no modelo espacial do ambiente, visando à maximização da cobertura do sinal (dentre outras possíveis métricas). Conforme exposto no capítulo de materiais e métodos, foi utilizada a metaheurística proposta por Scoot Kirkpatrick (KIRKPATRICK et al., 1983). O *Simulated Annealing* foi utilizado para a realização do processo de otimização, buscando encontrar a melhor solução viável, considerando o objetivo do problema em questão e o conjunto de restrições para aceitação da solução proposta.

4.4.1 Implementação do *Simulated Annealing*

Inicialmente a metaheurística buscava o melhor ponto apenas para um *access point*. Com o cumprimento dos objetivos tornou-se possível a busca e otimização de mais de um *access point*. O *Simulated Annealing* inicia com os valores predefinidos no momento da sua chamada. Para o método é informado: o número de *access points* a serem utilizados no ambiente, o número máximo de iterações, o número máximo de perturbações por cada iteração, o número máximo de sucessos por iteração e o α como o fator de redução da temperatura. Na seção 4.5.3 podem ser vistos mais detalhes sobre como a função objetivo foi implementada.

Com a estratégia de perturbar mais de um *access point* por vez, o *Simulated Annealing* inicia alocando todos os *access points* na posição central da planta baixa. A primeira iteração é avaliada (*access points* no centro) para que então seja realizada a busca e novas perturbações sejam feitas. O *Simulated Annealing* fica em um *loop* principal, no qual se verifica se foram atendidas as condições de término do algoritmo (se extrapolou o máximo de iterações ou se nenhum ponto com ótimo local foi encontrado). Outro *loop*

interno realiza a perturbação dos *access points* em forma de lista circular. Com os valores de $f(S_i)$ (resultado da avaliação da função objetivo na i -ésima iteração do *Simulated Annealing*) e $f(S)$ (melhor resultado da avaliação da função objetivo iteração do *Simulated Annealing*) é possível determinar o valor de ΔF_i (diferença da avaliação da função objetivo atual com a avaliação da função objetivo ótima encontrada até o momento). Caso o valor de ΔF_i seja menor ou igual a zero, há uma redução de energia, a qual implica que a nova solução é melhor que a anterior. Então é aceita a solução e $f(S_i)$ passa a ser a nova solução corrente. A aceitação desse tipo de solução é mais provável em altas temperaturas e bastante improvável em temperaturas reduzidas. Para reproduzir essas características, geralmente usa-se, para calcular a probabilidade de se aceitar a nova solução, uma função conhecida por critério de aceitação de *Boltzmann* (AARTS; KORST, 1988), em sua expressão o valor de T é a temperatura atual e que regula a probabilidade de soluções com pior custo e Δ representa a variação da função objetivo, isto é, $\Delta = f(s') - f(s)$. Tal expressão pode ser escrita como:

$$e^{(-\Delta/T)} \quad (4.3)$$

Com a aceitação, é atualizada a lista das melhores posições para a alocação dos *access points* e também a variável $f(S)$ com o valor de melhor função objetivo, o número de sucessos representando a vizinhança, que também é fator de parada do *Simulated Annealing*, e o número de perturbações por iteração é incrementado. O método é finalizado quando a temperatura chega a um valor próximo de zero, situação esta em que a chance de se aceitar uma solução vizinha pior é extremamente baixa, ou seja, quando o sistema estiver estável (LAARHOVEN; AARTS, 1987).

Por fim, a melhor solução encontrada pela metaheurística é retornada, indicando a proposta do novo posicionamento dos *access points* e a cobertura do sinal *Wi-Fi* alcançada, caso tal solução fosse implantada no ambiente real. Além disso, é fornecida uma visualização gráfica do resultado a ser obtido com a solução proposta utilizando o PyGame.

4.4.2 Calibração dos parâmetros do *Simulated Annealing*

Definir quais os parâmetros utilizar na metaheurística não é uma tarefa fácil, então para isso, qualquer ferramenta que auxiliasse neste processo seria útil. Para a escolha de parâmetros utilizar para os valores foi empregado um Projeto Fatorial 2^k com foco no ajuste do *Simulated Annealing*.

Com a aplicação do planejamento de experimentos é possível definir uma sequência de coletas de dados experimentais a fim de atingir um objetivo. Dentre os métodos de planejamento experimental disponíveis na literatura, o planejamento fatorial é o mais

indicado quando se deseja estudar os efeitos de duas ou mais variáveis de influência, sendo que em cada tentativa ou réplica, todas as combinações possíveis dos níveis de cada variável são investigadas (NETO; SCARMINIO; BRUNS, 1996). O caso mais simples de planejamento factorial descrito na literatura é aquele em que cada fator k está presente em apenas dois níveis (o experimento aplicado neste trabalho), ou seja, em um experimento com k fatores (ou variáveis) e dois níveis, obtendo-se 2^k experimentos a realizar.

Os valores dos parâmetros do *Simulated Annealing* foram definidos quando ainda estava sendo utilizado apenas um *access point*. Para o teste, foram considerados os valores da quantidade de vizinhos, temperatura inicial, o fator de resfriamento e a máxima de perturbações. O valor do máximo de iterações foi definido em 500, a posição do *access point* foi definida como sendo [u, v]. Foi desenvolvido um *script* em Python que automatizasse o processo. Com quatro parâmetros que se deseja saber os valores e são utilizados dois níveis de profundidade, foi utilizado o conceito de tabela verdade, contendo 2^4 linhas fazendo com que todas as possibilidades de parâmetros sejam testadas.

Com a coleta das informações obtidas pelo *script* em Python, os valores foram adicionados em uma planilha do Excel cedida pelo orientador. A tabela realiza todos os cálculos necessários para saber qual parâmetro é mais propício a se utilizar. A Tabela 1 apresenta a configuração da primeira iteração do planejamento 2^k com os fatores e níveis bem definidos.

Tabela 1 – Fatores para parâmetro para o *Simulated Annealing* na primeira iteração.

		FATORES		FATORES		FIXO			
k=	4	(nº vizinhos)	B (temperatura inicial)	C (fator resfriamento)	D (perturbações)	Iterações			
n=	Níveis	1 2	20 40	1 2	100 200	75% 95%	1 2	10 20	1000 1000

O resultado computado é dado na Tabela 2:

Tabela 2 – Candidatos a parâmetros na primeira iteração.

	CANDIDATOS A PARÂMETRO			
Índices	vizinhos	temperatura	resfriamento	perturbações
Índice Objetivo	1,4%	1,0%	1,2%	1,0%
Tempo	0,9%	0,8%	1,2%	1,0%
MÉDIA	1,2%	0,9%	1,2%	1,0%

Com o resultado acima, é possível notar que o melhor candidato é o número de vizinhos e o resfriamento. O valor dado em porcentagem representa o quanto promissor é ser tomado como um valor de parâmetro para o *Simulated Annealing*. Ainda que o valor de ambos tenha sido o mesmo, o número de vizinhos acaba sendo menor, pois somou uma fração menor do tempo total. O número de vizinhos foi aumentado para ter um

resultado melhor ainda; os valores da temperatura inicial também foram aumentados e o fator de resfriamento foi decrementado junto com o número de perturbações. Modificando os parâmetros, se torna propício a um novo melhor conjunto de valores do *Simulated Annealing*. Os novos valores da segunda iteração do planejamento factorial 2^k podem ser vistos na Tabela 3.

Tabela 3 – Fatores para parâmetro para o *Simulated Annealing* na segunda iteração.

		FATORES		FATORES		FIXO	
k=	4	A (n de vizinhos)	B (temperatura inicial)	C (fator resfriamento)	D (perturbações)	Iterações	
n=	Níveis	1 2	40 80	1 2 150 300	1 2 75% 85%	1 2 10 15	1000 1000

O resultado dado pela configuração da Tabela 3 pode ser visto na Tabela 4:

Tabela 4 – Candidatos a parâmetros na segunda iteração.

	CANDIDATOS A PARÂMETRO			
Frações	vizinhos	temperatura	resfriamento	perturbações
Função Objetivo	1,4%	1,0%	1,3%	1,0%
Tempo	1,2%	0,9%	1,2%	1,1%
MÉDIA	1,3%	0,9%	1,2%	1,0%

Como na iteração anterior, o melhor candidato a manter o seu valor é o número de vizinhos. Com a análise feita, o número de vizinhos no segundo nível e a temperatura inicial também obtiveram um bom resultado no segundo nível. O valor do resfriamento no segundo nível manteve sua média, enquanto o número de perturbações em ambos os níveis, teve um resultado inferior ao anterior, sendo necessário modificá-lo na próxima iteração. Na Tabela 5 é possível ver como ficaram definidos os parâmetros para a terceira iteração e último teste.

Tabela 5 – Fatores para parâmetro para o *Simulated Annealing* na terceira iteração.

		FATORES		FATORES		FIXO	
k=	4	A (n de vizinhos)	B (temperatura inicial)	C (fator resfriamento)	D (perturbações)	Iterações	
n=	Níveis	1 2	80 60	1 2 300 600	1 2 80% 85%	1 2 5 10	1000 1000

De acordo com a configuração da Tabela 5 são dados os candidatos a parâmetros expostos na Tabela 6.

Os resultados apresentados na Tabela 6 passaram por três refinamentos. Assim, como dito no início desta seção, o teste foi realizado com o intuito de aprimorar os parâmetros do *Simulated Annealing* com apenas um *access point*. A estratégia para a versão final e com múltiplos *access points* é definir também como um fator (variável), o maior do



Tabela 6 – Candidatos a parâmetros na terceira iteração.

CANDIDATOS A PARÂMETRO				
Frações	vizinhos	temperatura	resfriamento	pertubações
Função Objetivo	1,4%	1,0%	1,3%	1,0%
Tempo	1,2%	0,9%	1,1%	1,1%
MÉDIA	1,3%	0,9%	1,2%	1,0%

Tabela 7 – Parâmetros definitivos utilizados na metaheurística.

Raio de perturbação	Número máximo de iterações	Número máximo de pertubações	Número máximo de vizinhos	Decaimento da temperatura (Alpha)	Temperatura inicial
WIDTH * (0.025) (2.5% da largura da matriz)	600 * (n de APs)	5	80	85%	300 * (n de APs)

raio de perturbação e fazer com que ele seja uma porcentagem da largura da matriz de ~~l~~gação, tornando-o dinâmico de acordo com o tamanho da pl~~a~~A Tabela 7 mostra ~~de~~ma forma melhor como ficaram os parâmetros utilizados na me~~a~~heurística.

A resolução foi dada em metros e transformada de uma forma compatível com o processamento e a duração esperada para ~~isar~~ cada solução. Os parâmetros definidos para a alocação de vários *access points* foi um compilado de parâmetros utilizados no *Simulated Annealing* com um *access point* juntamente com ~~tes~~ feitos repetidas vezes.

4.5 Avaliação da solução

Nesta seção apresenta-se como foram implementadas as técnicas utilizadas para avaliações com um ou mais *access points*, bem como o aperfeiçoamento da função objetivo.

4.5.1 Avaliação da solução com um AP

Para a implementação do *Simulated Annealing*, se faz importante a decisão de uma boa função objetivo. A função ~~tivo~~ está diretamente ligada à eficácia com que o *Simulated Annealing* irá buscar seu otimo global. Inicialmente a função objetivo foi implementada de forma bem prática: como a matriz gerada continha em cada posição o resultado da potência do sinal, conforme ele se propagava pelo ambiente, para representar uma indicação numérica da cobertura do sinal *Wi-Fi* foi feita a soma de todos os valores da matriz. Num primeiro momento, tentamos somar os valores da potência do sinal em *mW* mas em função do decaimento exponencial do sinal com o distanciamento do transmissor, a precisão de um número de ponto flutuante chegava rapidamente ao limite de seus 32 bits. Tão pouco adiantou aumentar a precisão para 64 bits ou mais. Curiosamente, uma função objetivo baseada na soma dos valores em miliwatts dos sinais simulados era demasiado custosa de se calcular e trouxe pouco benefício quanto à exploração do espaço de soluções pelo *Simulated Annealing*.

Considerando o acima exposto, decidiu-se por manter os valores da matriz resultante em dB , uma escala logarítmica que representa a ordem de magnitude da intensidade do sinal. Assim, apesar de não ser conceitualmente correto somar valores de potência em dB , nosso objetivo não era ter o valor exato em mW da soma de toda a energia irradiada no ambiente simulado, mas sim ter uma noção da ordem de magnitude do quanto de sinal útil aquela simulação do ambiente apresentava. Observe que as   sem fio tipicamente apresentam sensibilidade mínima em torno de $-85 dB$ a $-100 dB$ (em torno de $3,162 \times 10^{-12} W$ a no mínimo $1 \times 10^{-13} W$), assim sendo, quanto mais “positivo” fosse o valor da função objetivo avaliada, maior seria o montante da potência dos sinais que cobriam o ambiente simulado. Levando em considerações tais fatos, experimentou-se com diversas maneiras de sumarizar a matriz em um só valor para ser avaliado como função objetivo e curiosamente, aquele que apresentou o melhor custo-benefício foi justamente a soma simples de todos os valores da matriz, mesmo que as informações nela armazenada estivessem em dB . Numa inspeção visual da solução proposta pelo *Simulated Annealing*, com a coloração dos valores de dB em um gradiente de cores de acordo com o posicionamento do *access point*, pode-se observar que a partir de um ponto aleatório, a metaheurística  conduzia as soluções candidatas para a região central do ambiente 2D, como seria desejado.

4.5.2 Aperfeiçoamento da função objetivo

Assim que foi implementada uma primeira versão das etapas de entrada e processamento do arquivo `.ax`, foram conduzidos testes no simulador e na otimização, de maneira a checar a proximidade de dados reais coletados com a simulação e o quanto a otimização conseguiria melhorar a solução, direcionada por meio de sua função objetivo. Tal continuidade na condução dos testes permitiram alterações e aperfeiçoamentos na função objetivo da metaheurística de otimização e na simulação das características do ambiente e da propagação dos raios (sinais de RF). Foram comparados os resultados simulados com resultados reais coletados nas dependências do *campus*.

4.5.3 Avaliação da solução para dois ou mais APs

Como desde o princípio, a implementação da função objetivo para apenas um *access point* visava maximizar a cobertura do sinal *Wi-Fi* para o ambiente. Isso não foi diferente para a simulação de mais um *access point*. Foram utilizadas técnicas de manipulação de grandezas em *dB* para obter o resultado esperado. Como dito na seção 4.5.1, a manipulação com a soma dos dados em *dB* não é conceitualmente apropriada. Como uma proposta de solução para esse problema, antes de realizar a soma dos valores da matriz, a potência do sinal em questão era convertido de *dB* para *mW*, fazendo com que a operação ficasse correta e uma verificação da potência recebida com a sensibilidade do *access point*

também foi feita para penalizar zonas em que  al era tão fraco que não se tornava uma  válida. Com a conversão dos valores  e a verificação da qualidade do sinal  adicionada, o tempo de execução do algoritmo aumentou consideravelmente, fazendo com que outra alternativa fosse buscada.

Diante de uma opção que demandava mais tempo para que fosse processada, outra estratégia para a função objetivo  implementada. Quando se tem mais um *access point* para buscar os seus respectivos  ótimos, somar suas matrizes de resultados também não é algo que dê um resultado correto. Foi, então, implementado um método que realizava a sobreposição das matrizes de  da propagação. Para o *Simulated Annealing* saber se a escolha era ruim ou boa,  transformar essa escolha em um número. Nesse sentido,  acordo com as n matrizes de resultado de n *access points*, foi necessário realizar suas .

Tal método recebe como parâmetro uma lista ( ou *array*, ou vetor, em Python são a mesma estrutura de dados) contendo a matriz resultado da simulação de todos os *access points* e uma variável *size* que corresponde à quantidade de *access points* no ambiente. Posteriormente é feito um laço de repetição percorrendo a lista de matrizes e guardado as matrizes máximas utilizando o método *maximum()* da biblioteca Numpy retornando tal valor. O método que realiza essa operação pode ser visualizado a seguir.

```
def sobrepoem_solucoes_MAX(propagation_array , size):
    max = propagation_array [0]
    for i in range(1, size):
        max = np.maximum(propagation_array [i] , max)
    return max
```

 método retorna à matriz resultante e realiza a avaliação da lista de *access points* fazendo uso da matriz para obter um valor a se comparar no processo de busca de um ponto melhor no *Simulated Annealing*. Os cálculos para a penalização de área  de o sinal tem uma qualidade de  a sensibilidade do equipamento não foram implementados nessa estratégia.

4.6 Análise da utilização de recursos

A fim de buscar uma melhor visualização do fluxo de execução do algoritmo, foi utilizada uma ferramenta de *profile*, chamada *cProfile*, para geração do arquivo com extensão *.cprof* e o software *pyprof2calltree* para realização desta análise. Com tais ferramentas é possível obter análises da porcentagem de tempo gasto em cada método do algoritmo, de acordo com o tempo total. Como é possível ver a seguir, foi gerada uma imagem de saída utilizando o software *pyprof2calltree* com o fluxo de execução do método *Run*, que é res-

ponsável desde os cálculos de proporção da matriz/planta, leitura de arquivos de entrada, até a etapa final, com a exibição do resultado em forma gráfica, utilizando o PyGame.

Assim, cada retângulo do grafo da Figura 17 representa um método implementado, sendo eles referentes aos métodos implementados utilizando Python e aos métodos de bibliotecas utilizadas no desenvolvimento da aplicação. Também pode ser visto que cada aresta no grafo possui uma barra de porcentagem representando visualmente sua fatia de tempo gasto na aplicação como num todo, juntamente com o número de vezes que o método foi chamado podendo assim concluir que os cálculos numéricos são realizados rapidamente e o que demanda tempo é a representação gráfica dos dados. Para realizar a interpretação do arquivo de saída, basta utilizar o comando `pyprof2calltree -k -i PlacementAPs.cprof` no terminal com o arquivo gerado ao fim da simulação, nesse caso, `PlacementAPs.cprof`.

4.7 Paralelização em GPU

A decisão de fazer uso da GPU para ajudar no desempenho foi eficiente. A transição de um código sequencial para um código paralelizado foi feita aos poucos. No início, foi importado da biblioteca numba o módulo `jit` adicionado sob todos os cabeçalhos dos métodos a anotação `@jit`. Apenas com isso, o código será compilado em código de máquina no momento que for executado (*just-in-time*). A seguir é possível ver um exemplo de um método que foi compilado para código de máquina utilizando a anotação `@jit`,

```
@jit
def calc_distance(x1, y1, x2, y2):
    return sqrt(pow((x1-x2), 2.0)+pow((y1-y2), 2.0))*precisao
```

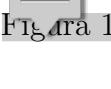
Após utilizar as anotações nos métodos, foi escrito o método que demandava mais recursos e demandava todo o tempo. Utilizando o resultado obtido do *Profile*, foi possível ver que o método era o que realizava a avaliação da solução em ponto dentro do *Simulated Annealing*. Neste momento, o método calcula a intensidade do sinal para cada ponto da matriz e, para cada ponto, calcula a absorção das paredes, percorrendo uma lista com todas as paredes do ambiente. Quanto maior o detalhamento do ambiente, mais linhas o ambiente terá e maior será a lista de paredes a se verificar absorções.

Posteriormente à decisão do método a deixar a cargo da GPU realizar os cálculos, todo o código foi transcrito de forma a dividir todo o trabalho entre os núcleos de CUDA disponíveis. No código a seguir é possível ver, no método que realiza a propagação do sinal, que uma matriz utilizando a biblioteca numpy é criada, as dimensões dos *blocks* e *grids* (os valores foram escolhidos empiricamente, dependem do *hardware* utilizado e do problema a ser trabalhado). Logo após, a matriz é enviada para a GPU para ser compartilhada

entre *blocks* e *threads*. O método que realiza o cálculo da intensidade do sinal é chamado de acordo com os *blocks* e *grids* definidos. Então, a  que foi enviada para a GPU é trazida de volta e retornada para continuar com os .

```
@jit
def simula_propagacao_gpu(pointX, pointY):
    g_matrix = np.zeros(shape=(WIDTH, HEIGHT), dtype=np.float32)
    blockDim = (48, 8)
    gridDim = (32, 16)
    d_matrix = cuda.to_device(g_matrix)
    simulate_kernel[gridDim, blockDim](pointX, pointY,
                                       d_matrix, floor_plan)

    d_matrix.to_host()
    return g_matrix
```

Na  é possível compreender facilmente como é a divisão dos *blocks*, *grids* e *threads*.

O equipamento utilizado para o  utiliza uma placa de vídeo NVIDIA modelo GeForce GT 740M¹¹ com 2GB de memória dedicada; possui 384 núcleos CUDA, com um *clock* básico  93 MHz, 1.300 milhões de transistores e com vazão do barramento PCIe de até 80 Gb/s.

O método “*simulate_kernel_gpu*” foi escolhido para trabalhar com a matriz de propagação. Nele, a matriz é percorrida de acordo com os valores dos *blocks* e *grids* informados no método anterior. Para cada célula da matriz é executado o método “*propagation_model_gpu*”, o qual, segundo o próprio nome já diz, realiza o cálculo do modelo de propagação utilizando a GPU e está diretamente ligado ao “*propagation_model*” que obtém valores do modelo de propagação e os miliwatts a serem subtraídos do cálculo da perda por paredes. O código referente ao “*simulate_kernel_gpu*” pode ser visto abaixo.

```
@cuda.jit
def simulate_kernel(apX, apY, matrix_results, floor_plan):
    startX, startY = cuda.grid(2)
    gridX = cuda.gridDim.x * cuda.blockDim.x
    gridY = cuda.gridDim.y * cuda.blockDim.y
    for x in range(startX, WIDTH, gridX):
        for y in range(startY, HEIGHT, gridY):
            matrix_results[x][y] = propagation_model_gpu(x, y,
                                                          apX, apY, floor_plan)
```

¹¹ Disponível em <<https://www.geforce.com/hardware/notebook-gpus/geforce-gt-740m>>. Acesso em: 29 out. 2017.

O cálculo do valor do sinal naquele ponto é $O(k)$ pois consiste em um operação envolvendo multiplicação, divisão, exponenciação/logaritmo, portanto, assintoticamente é $O(1)$. Em cada simulação deve-se calcular o valor para a propagação em cada ponto de uma matriz $N \times M$. Considerando que uma das dimensões é maior que a outra, pode-se dizer que o custo é $N \times N$: $O(n^2)$ e $O(1 \times n^2) = O(n^2)$ para aplicar a operação em todos os pontos da matriz. Mas, entre cada ponto da matriz (PM) e o *access point*, podem haver k paredes, que deve-se verificar se há ou não interseção entre a reta AP-PM e a reta formada pela parede. Tal operação, utilizando a equação de geometria analítica para interseção de retas, custa $O(1)$ para cada parede e $k \times O(1)$ para verificar cada parede. Como k é suficientemente grande em relação a n , podemos considerar o custo de verificar se há interseção com paredes como $O(n)$. Para fazer essa verificação de cada parede para cada ponto da Matriz, temos então: $O(n) \times O(n^2) = O(n^3)$.

Quanto ao *Simulated Annealing* (SA), a cada rodada ele aleatoriza uma solução vizinha (v) e avalia-a (fo), e como o SA tem uma quantidade finita e decremental de iterações em função do fator de resfriamento, a quantidade de vizinhos explorados é em torno de $v = \log(n)$. Assim, a complexidade do *Simulated Annealing* é em torno de $O(fo * v)$, ou seja: $O(fo \times \log(n))$. Como a função objetivo (fo) consome $O(n^3)$, teríamos: $O(n^3 * \log(n))$.

É importante notar que a anotação deste método é diferente das demais que utilizam o `@jit` para gerar código de máquina. A anotação `@jit` é o caminho geral do compilador, que pode ser orientado opcionalmente para um dispositivo CUDA. A anotação `@cuda.jit` é efetivamente o dialeto de núcleo de mais baixo nível do CUDA para Python que a empresa *Continuous Analytics* desenvolveu. Assim é possível receber suporte para variáveis embutidas do CUDA como o `threadIdx` e especificadores de espaço de memória compartilhada. Caso seja necessário escrever um *kernel* CUDA (como o método `simulate_kernel_gpu`) em Python e compilá-lo e executá-lo (como ocorreu nesse caso), deve-se usar a anotação `@cuda.jit`. Caso contrário, se for necessário apenas acelerar algum método existente, deve-se usar somente `@jit` com uma anotação alvo do CUDA.

4.8 Validação dos modelos

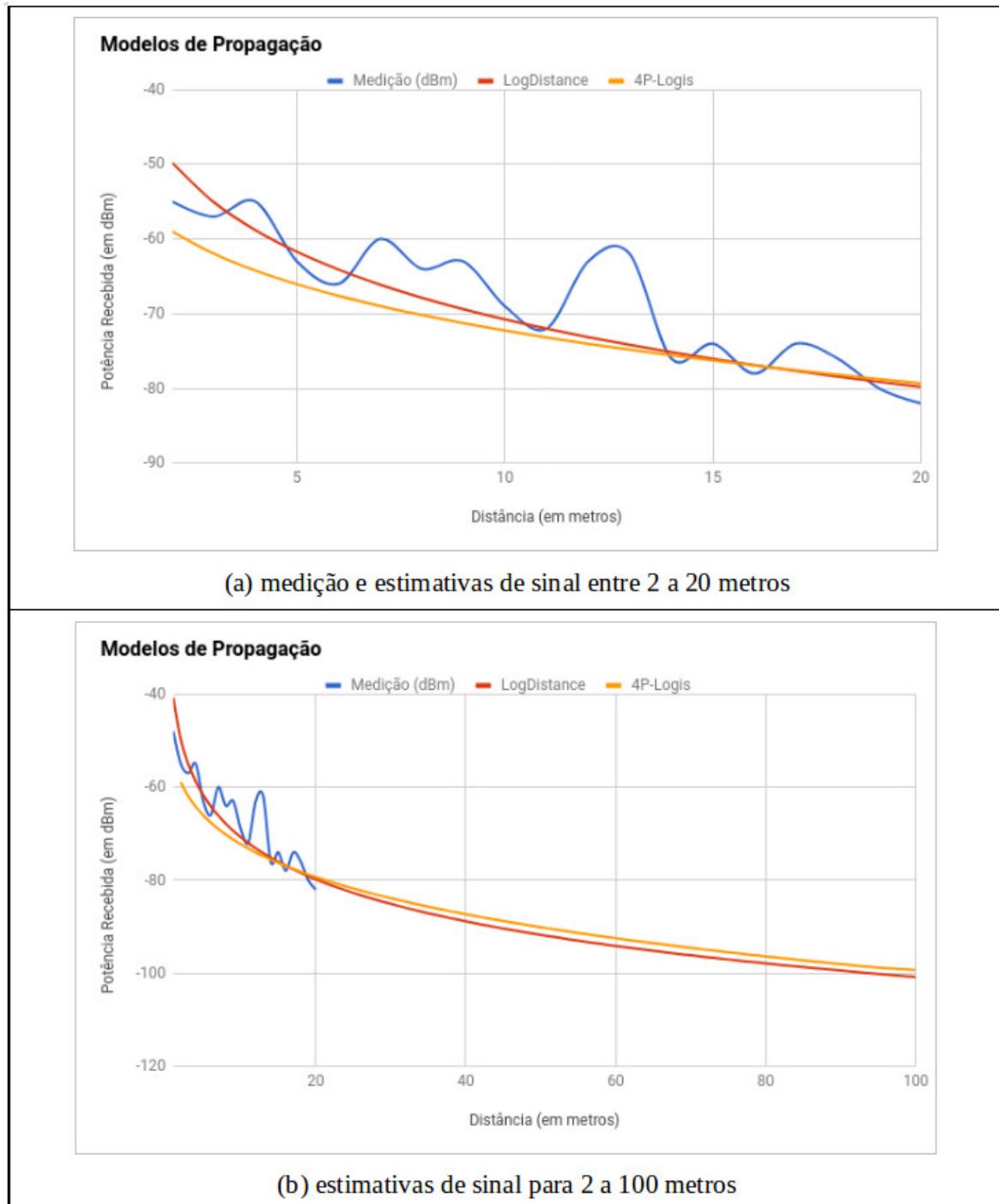
Durante e após a implementação dos modelos de perda e atenuação do sinal *wireless* e modelagem de propagação dos sinais pelo ambiente, foram comparados os resultados da simulação com o resultado real esperado. Para tal, pode-se utilizar o atual posicionamento dos *access points Wi-Fi* no *campus* e realizar experimentos de *Wireless Site Survey*, ou seja, medições reais da intensidade de sinal em determinados pontos do ambiente analisado.

Tais resultados foram então comparados e podem ser visto com a Figura 19 e a Figura 20 com os resultados previstos pela simulação (através da visualização gráfica da

propagação no ambiente), para que os modelos de propagação tracterísticas do modelo espacial pudessem ser criados, num processo de melhoria contínua visando ao aumento da precisão da solução apresentada. É válido reiterar que os parâmetros do *Simulated Annealing* e da função *NI Log* foram calibrados para melhor representar o ambiente do IFMG *campus* Formiga, de acordo com seus materiais de construção, espessura de suas paredes, pisos e teto, desta forma, é notável que os valores estão de acordo com as medições realizadas (BATISTA, 2017).

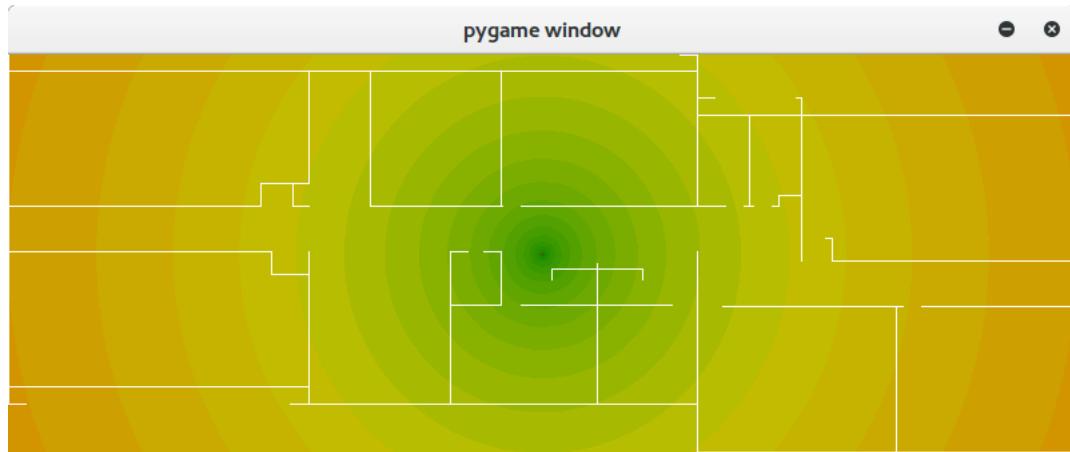
Assim é possível ver como é notável a aplicação de que os modelos teóricos obtiveram o comportamento relativamente parecido com a coleta e simulação realizada empiricamente, fazendo com que o modelo utilizado esteja realmente validado com a realidade. A seguir são apresentados os resultados obtidos após as simulações da propagação de sinais *wireless* nas dependências do IFMG *campus* Formiga e uma análise dos resultados encontrados nas simulações com um, dois ou mais *access points*.

Figura 9 – Comparação dos modelos de propagação LogDistance e NP-Logístico.



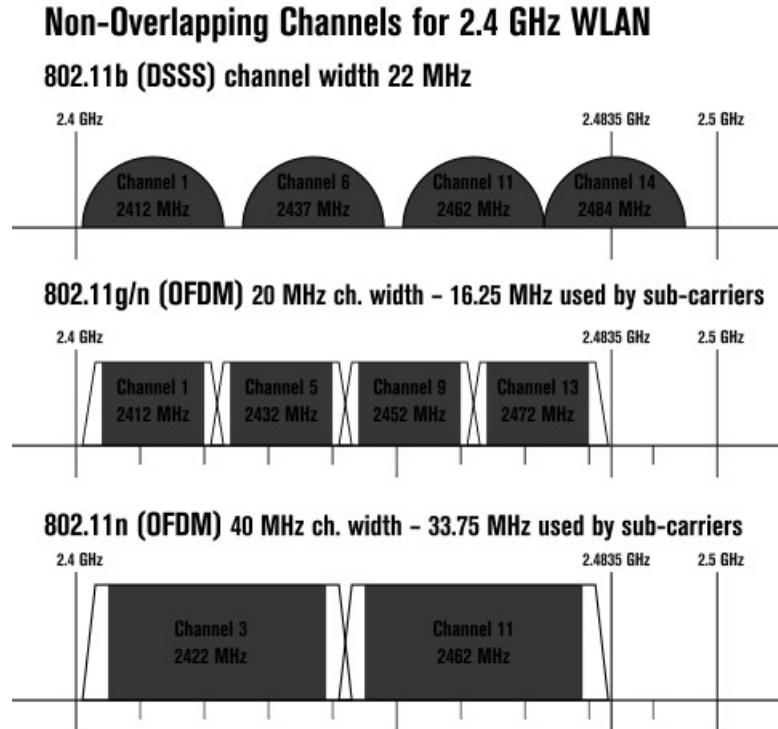
Fonte: Elaborado pelo autor.

Figura 10 – Decaimento RSSI de $Wi-Fi$ de acordo com a distância ao $accesspoint$.



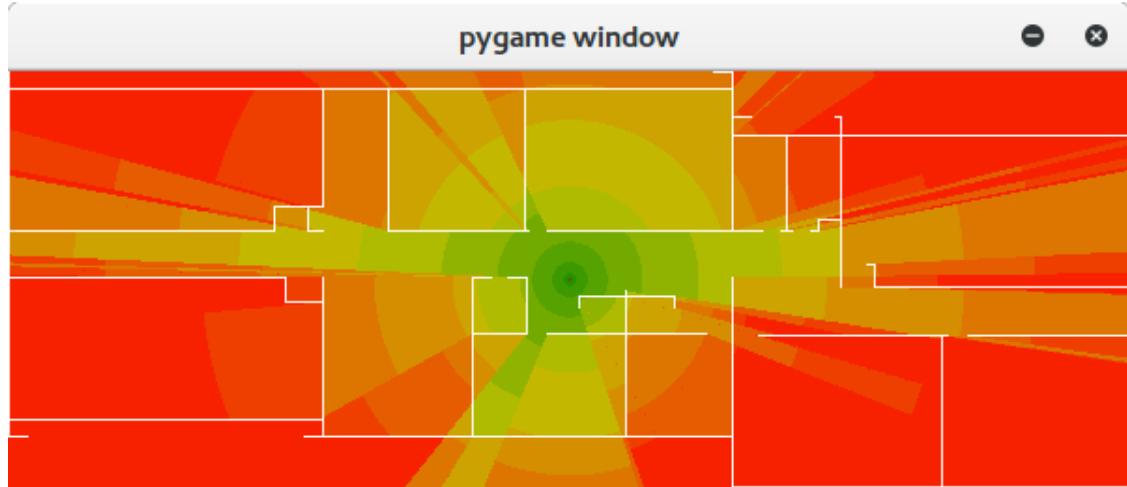
Fonte: Elaborado pelo autor.

Figura 11 – Canais não sobrepostos para WLANs de 2,4 GHz.



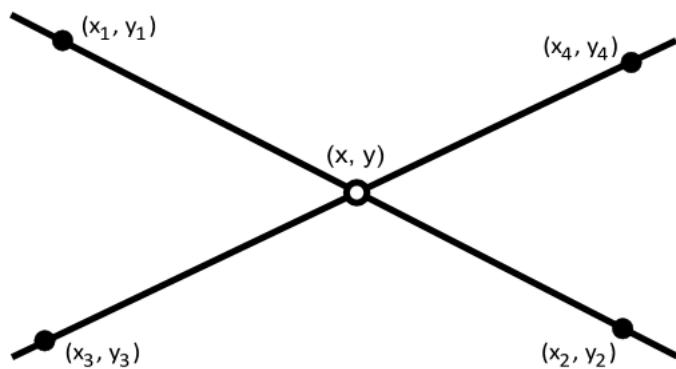
Disponível em: <<https://en.wikipedia.org/wiki/File:NonOverlappingChannels2.4GHzWLAN-en.svg>>. Acesso em 29 out. 2017.

Figura 12 – Absorção do sinal por cada parede atravessada.



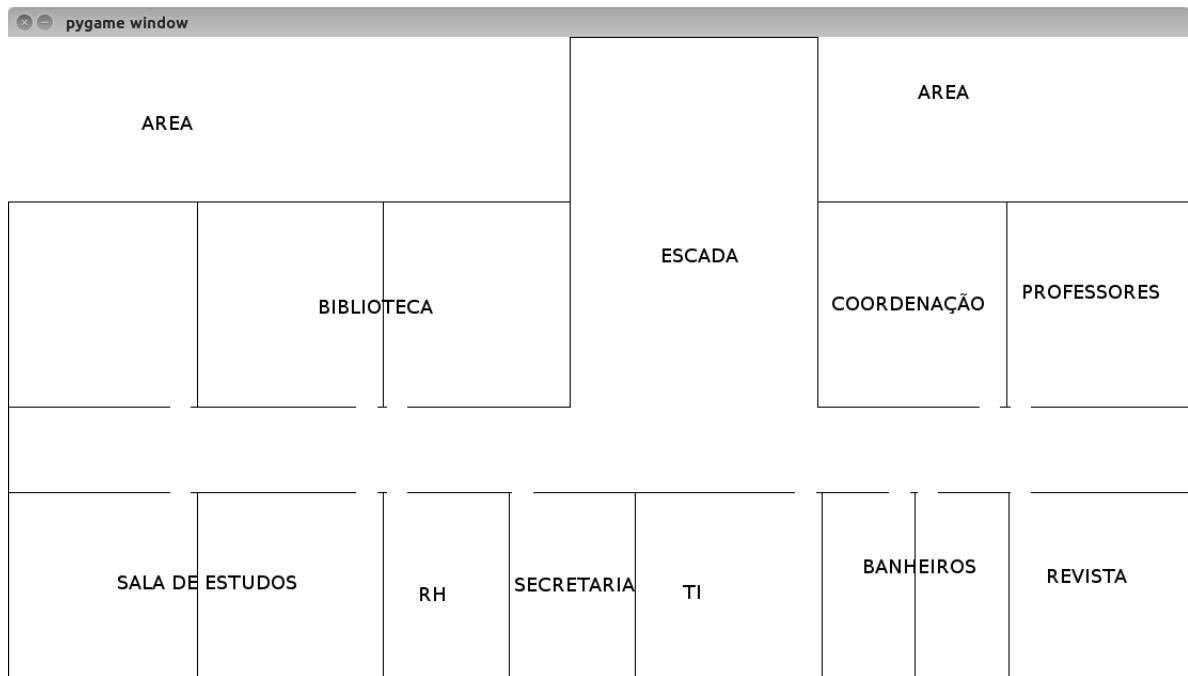
Fonte: Elaborado pelo autor.

Figura 13 – Interseção de retas.



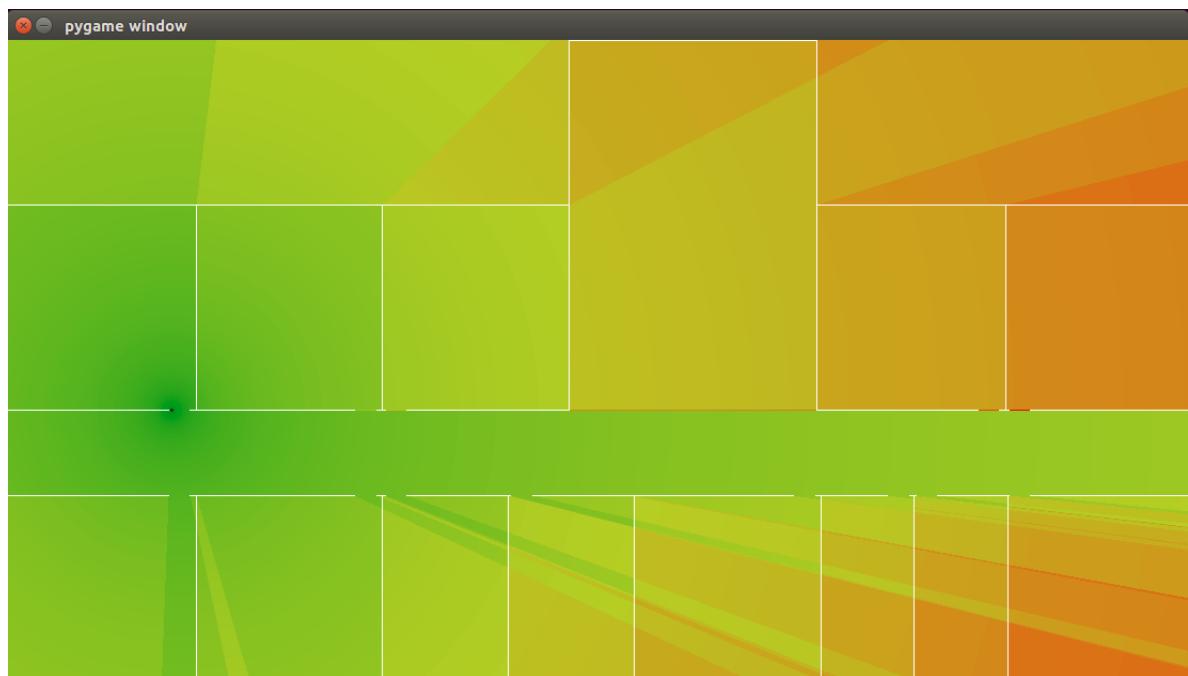
Fonte: Elaborado pelo autor.

Figura 14 – Representação do ambiente a partir do arquivo *DXF* contendo a planta-baixa do bloco A do IFMG *campus Formiga*.



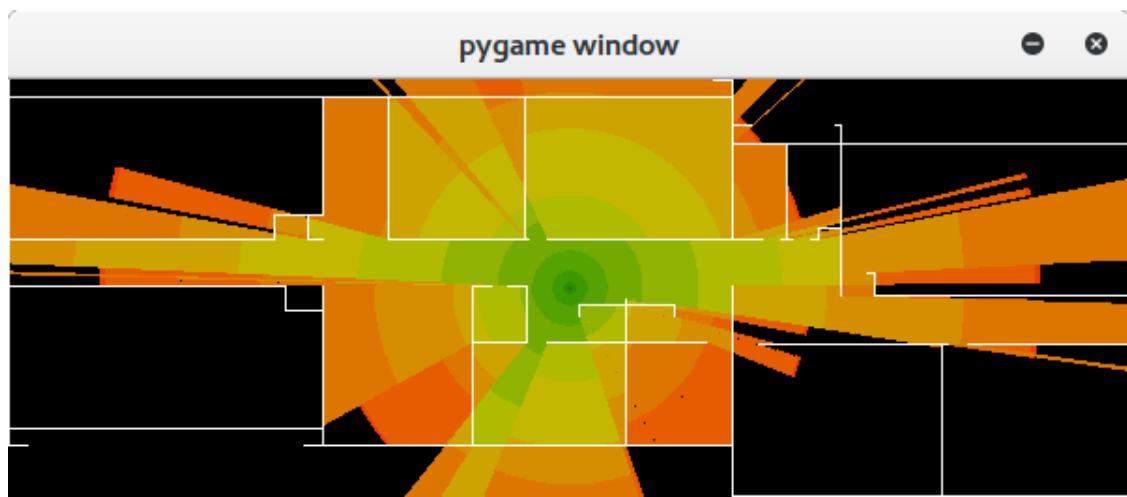
Fonte: Elaborado pelo autor.

Figura 15 – Simulação da propagação utilizando 256 cores no mapa de calor.



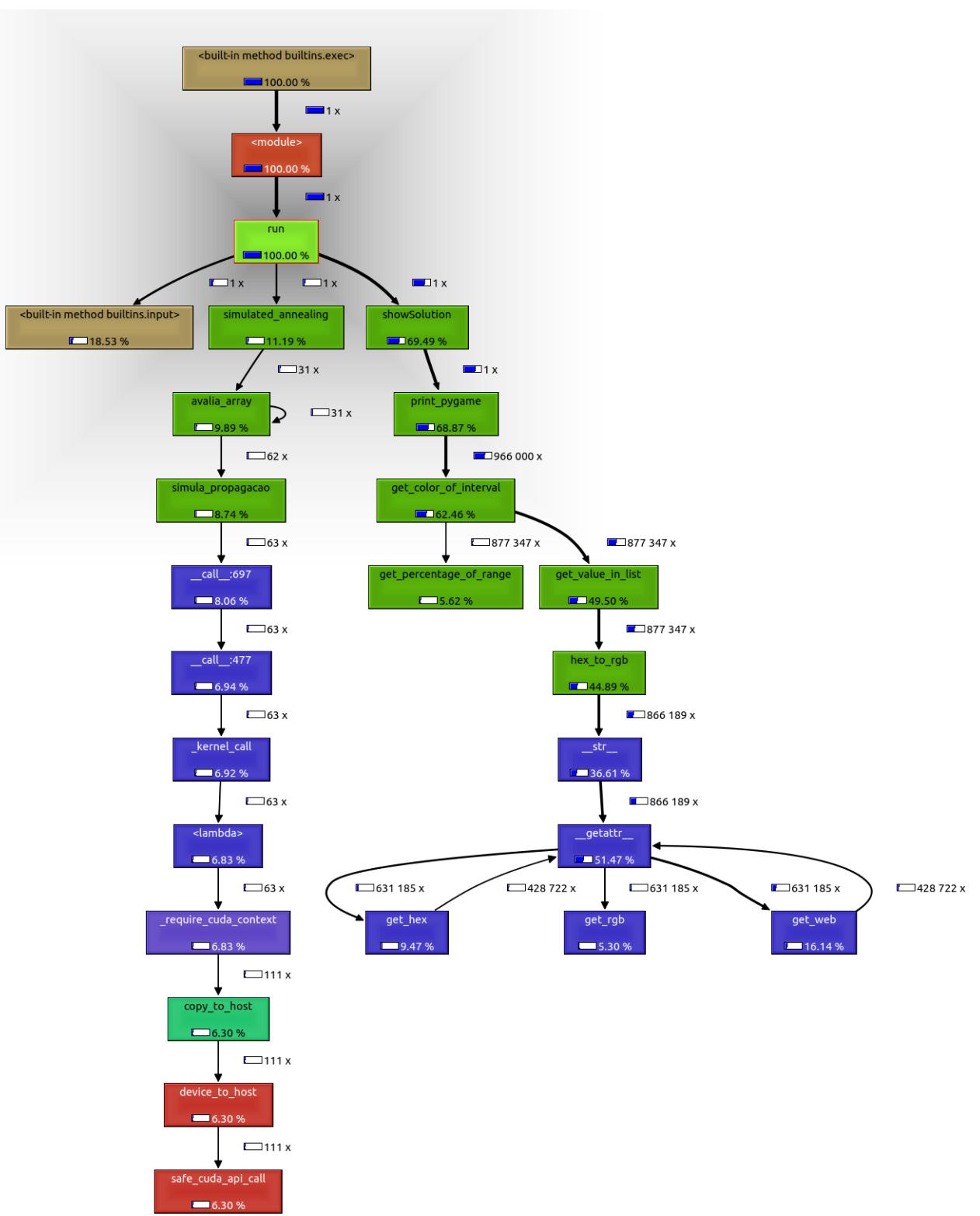
Fonte: Elaborado pelo autor.

Figura 16 – Regiões com sinal abaixo da sensibilidade máxima dos equipamentos no bloco C.



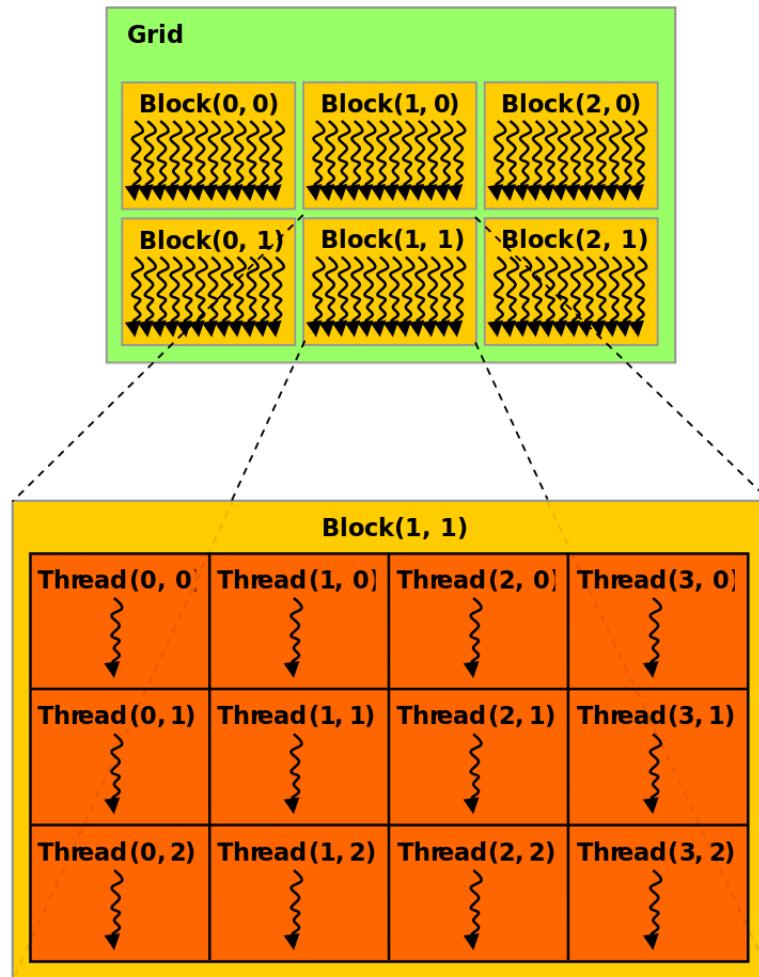
Fonte: Elaborado pelo autor.

Figura 17 – Grafo do ciclo de execução do algoritmo.



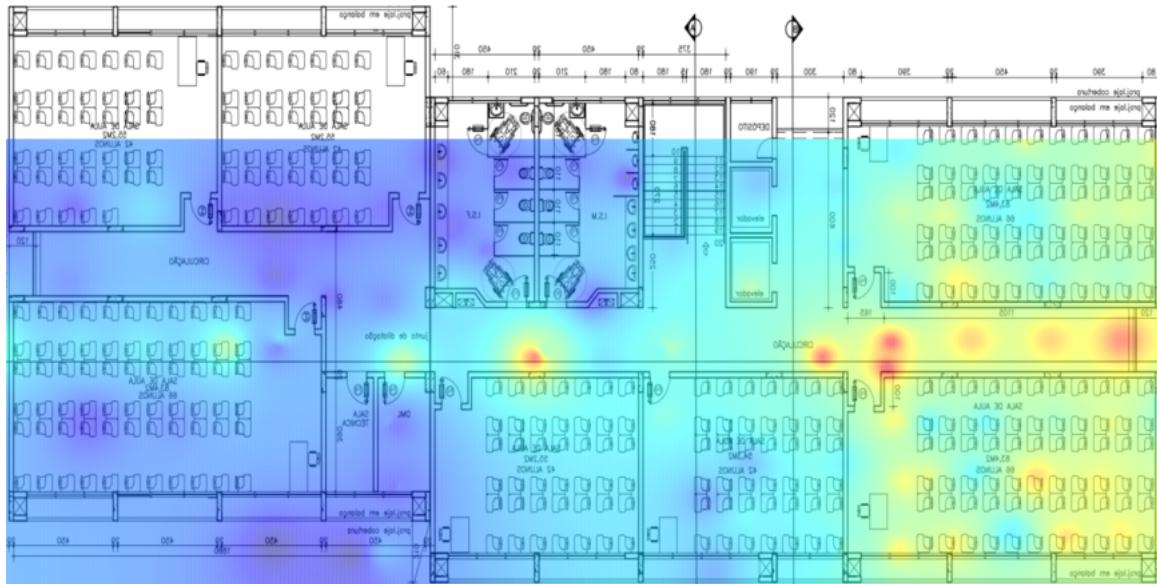
Fonte: Elaboração do autor.

Figura 18 – Divisão interna da GPU dos blocks, grids e threads.



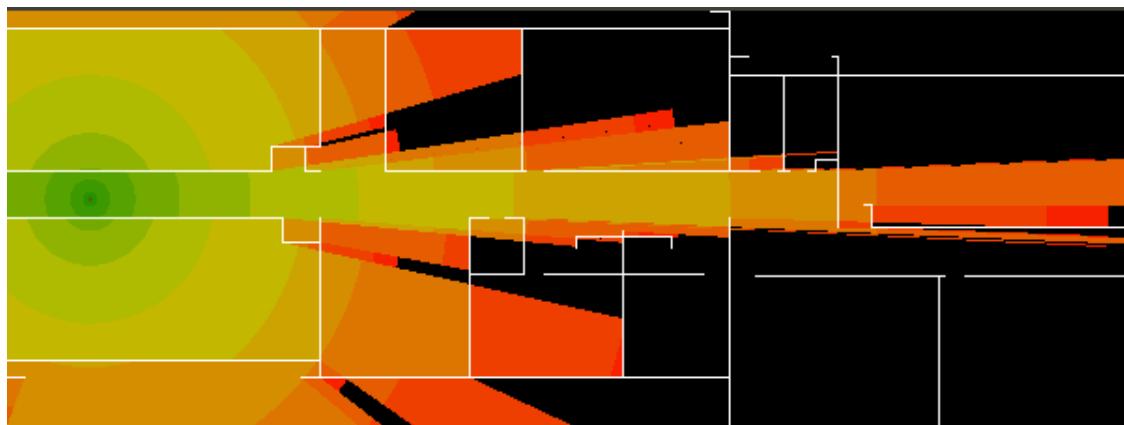
Disponível em: <https://en.wikipedia.org/wiki/Thread_block>. Acesso em 28/10/2017

Figura 19 – Captura realizada no Bloco C, para um *access point* "18:6B:9D:69:E8:B2" posicionado em (x=660,y=260) e irradiando no Canal 11 (2.462 GHz).



Fonte: (BATISTA, 2017).

Figura 20 – Simulação realizada com dados empíricos nas mesmas configurações do bloco C.



Fonte: Elaboração do autor.

5 RESULTADOS E ANÁLISE

Neste capítulo são mostrados os resultados obtidos a partir da implementação de toda a fundamentação teórica e a explicação dos procedimentos desenvolvidos. Apresenta-se também uma análise de custo computacional e um ajuste de curvas utilizando os softwares *cProfile* e o *R-Project*, respectivamente.

5.1 Simulação da propagação de sinais wireless

As primeiras simulações gráficas realizadas tiveram como objetivo apenas exibir o comportamento do espectro do sinal *Wi-Fi*, considerando tanto o seu decaimento com a distância quanto sua absorção pelas redes, assim, demonstrar graficamente o seu valor dentro da matriz de propagação. A Figura 21 mostra a simulação da propagação de sinais de *microondas* na planta baixa do bloco A do IFMG *campus Formiga*. Para a obtenção de tal resultado, a simulação e dos próximos resultados que são mostrados neste capítulo, utilizou-se o código em Python¹ que faz uso da GPU.

Na Figura 22 tem-se quatro capturas, nas quais as imagens (a), (b) e (d) não possuem as portas ou janelas implementadas, ou seja, os cômodos são tratados como ambientes fechados. Já a captura (c), por sua vez, possui o resultado gráfico da simulação e tem aberturas nas paredes, simulando as portas. Observe que quanto maior é o número de paredes, com a atenuação do sinal, sombreamentos são criados.

Com a execução do algoritmo, além de se ter a representação gráfica do ambiente através do Python, as informações da configuração da execução, tais como os resultados, são impressos no terminal. Abaixo é possível ver como é chamado o algoritmo e uma saída de exemplo dos dados.

```
$ python PlacementAPs.py
```

Simulando ambiente com: 400 x 149 pixels

Escala de simulação: 1 px : 0.1275 metros

Quantidade de APs: 2

Potencia de cada APs: -25 dB

FO global best: 5.762e+02

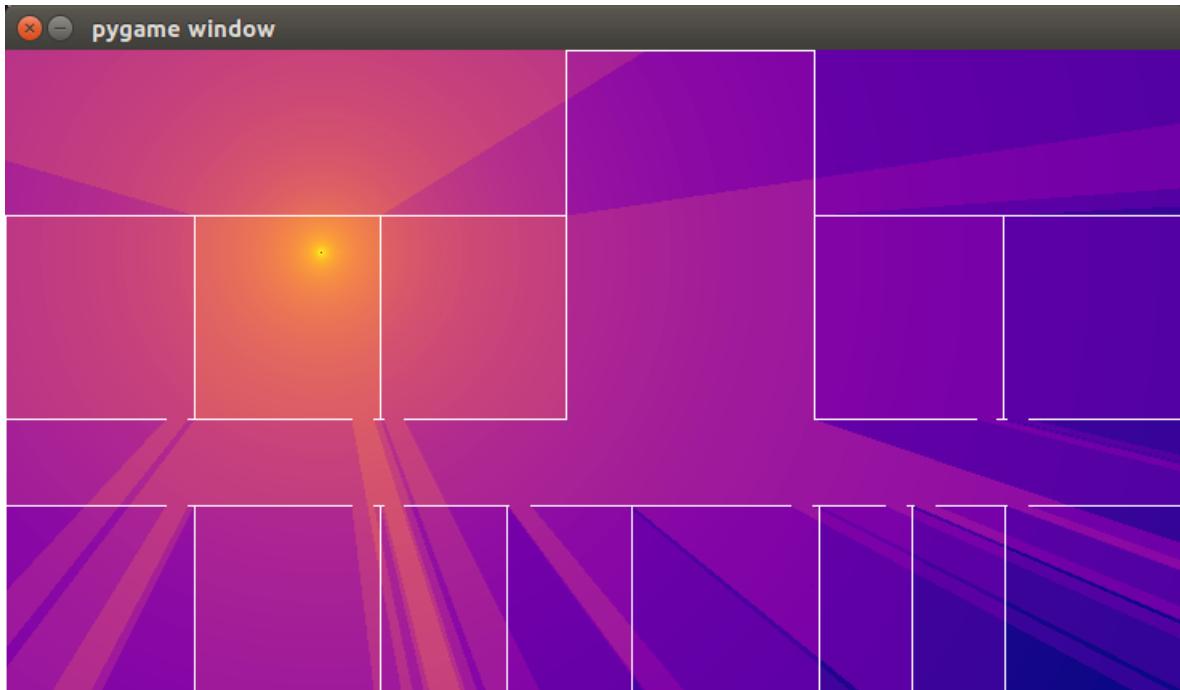
COBERTURA DE SINAL WI-FI:

87.62% com boa cobertura (sinal forte)

¹ <<https://github.com/samuelterra22/tcc/blob/master/PlacementAPs.py>>



Figura 21 – Ilustração da propagação de sinais de microondas no bloco A utilizando versão inicial do algoritmo.



Fonte: Elaboração do autor.

12.38% de zonas de sombra (abaixo da sensibilidade)

Cobertura por FAIXAS de intensidade de sinal

sinal Otimo	24.3%
sinal Bom	26.3%
sinal Ruim	37.0%

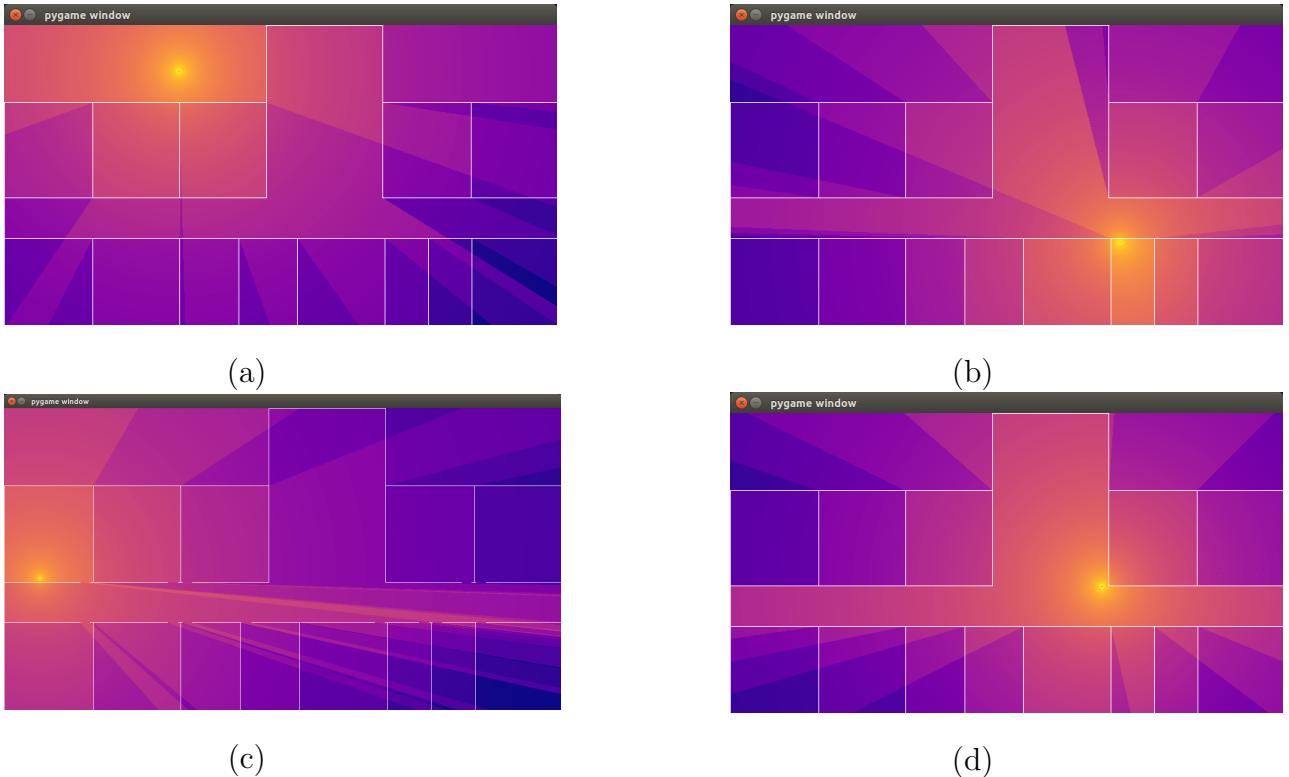


Ainda com a saída de dados, ao fim da execução da otimização da posição dos *access points*, é gerado um gráfico utilizando a biblioteca *matplotlib.pyplot* do Python. Com o gráfico mostrado na Figura 23, é possível ver claramente como a função objetivo da metaheurística se comportou durante a busca pelo espaço de soluções. Observe que, com os seus algoritmos baixos, o *Simulated Annealing* fugiu de seus ótimos locais e, no final da execução, retornou à sua melhor avaliação.



Neste sentido, é importante ressaltar que a Figura 21 e a Figura 22 representam os resultados dos primeiros testes executados. O gradiente de cores utilizada foi o plasma, variando das cores laranja até o azul escuro, onde a cor azul representa as zonas com menor intensidade e a cor laranja as zonas de maior intensidade. Na próxima seção são mostrados os resultados de testes efetuados utilizando ainda um *access point* com uma nova escala de cores, mais ainda apropriada a interpretação imediata do que seria um

Figura 22 –  ilação da propagação inicial de sinais de microondas no bloco A utilizando versão inicial do algoritmo.



Fonte: Elaboração do autor.

sinal ótimo, bom e ruim.

5.2 Wi-Fi Placement para 1 AP

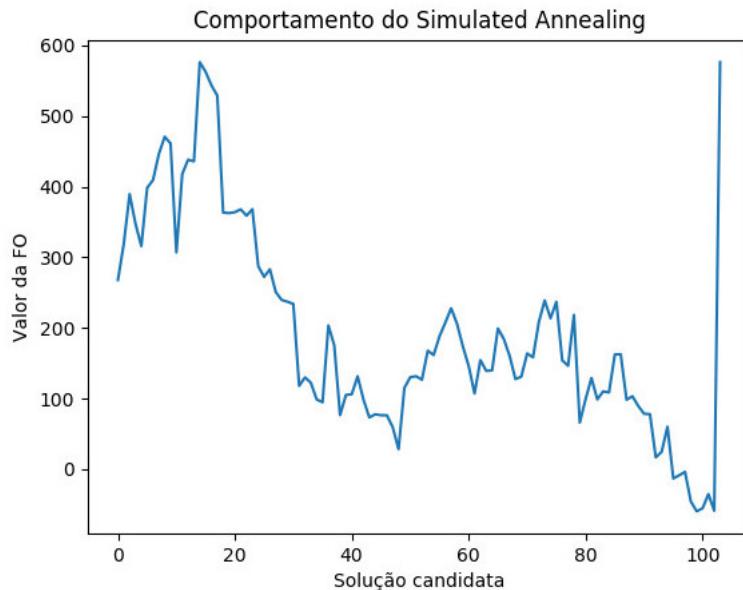
 A Figura 24, se comparada com a Figura 22, ~~po~~ ~~que~~ cores mais intensivas para os resultados obtidos, além de uma restrição na parte gráfica, destacada com a cor preta  áreas onde o sinal é menor que a sensibilidade típica de um equipamento Wi-Fi (seja ele um computador ou *access point*). De vez, a cor verde representa as zonas de maior intensidade do sinal e a cor vermelha, as zonas com sinal relativamente ruim.

 É possível notar que o *Simulated Annealing* ~~deixou~~ como um caso ótimo, nesta situação, um ponto próximo ao meio da planta. Com a posição escolhida, a cobertura do sinal se propaga  reas próximas à biblioteca e sala de estudo, que contém o maior fôrma de pessoas ~~neste~~  que utilizam o acesso à Internet. Na Figura 24, os pontos que ~~com~~ a desejar corresponde à potência máxima que consegue transmitir, neste caso, -25 dB.

 Como resultado disponibilizados via terminal, tem-se:

```
$ python PlacementAPs.py
```

Figura 23 – Componente da função objetivo do *SimulatedAnnealing* na busca do melhor ponto.



Fonte: Elaboração do autor.

Simulando ambiente com:

600 x 325 pixels

Escala de simulação:

1 px : 0.0800 metros

Quantidade de APs:

1

Potência de cada APs:

-25 dBm

FO global best: 4.432e+02

COBERTURA DE SINAL WI-FI:

74.32% com boa cobertura (sinal forte)

25.68% de zonas de sombra (abaixo da sensibilidade)

Cobertura por FAIXAS de intensidade de sinal

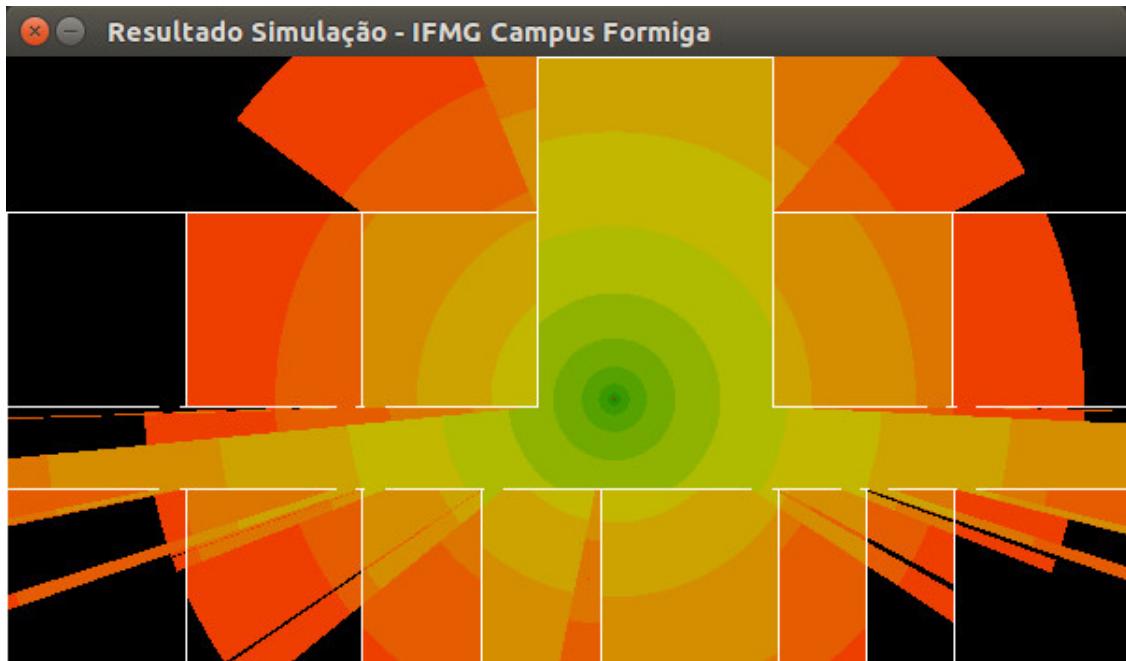
sinal Otimo 15.8%

Bom 26.0%

Ruim 32.5%

Com a Figura 25 podemos analizar, em forma de gráfico de pizza, a solução dada pelo algoritmo ao realizar a simulação com um *access point* no bloco A, com cada fatia correspondendo à quantidade do sinal propagado. Observe que as zonas de sombra (sem

Figura 24 – Simulação da propagação de sinais de microondas no bloco A utilizando 1 access point com potência de transmissão de $-25 dBm$. A figura absorção e limiar de sensibilidade ao longo das salas e corredores do segundo piso do bloco A.



Fonte: Elaboração do autor.

cobertura do sinal) tiver uma fatia correspondente a mais de um quarto de todo o espaço e, se somado ao sinal ruim, mais da metade da área não será atendida com uma boa qualidade de serviço.

As plantas dos pisos 3 do bloco C foram obtidas em formato DWG e convertidas para o formato .ax, ao final da implementação. Tais arquivos foram simulados e obtidas suas representações gráficas. A Figura 26 mostra a simulação de sinal baixa no bloco C utilizando 1 access point com potência de transmissão de $-25 dBm$.

A saída via terminal é a seguinte:

```
$ python PlacementAPs.py
```

Simulando ambiente com:

600 x 223 pixels

Escala de simulação:

1 px : 0.0850 metros

Quantidade de APs:

1

Potência de cada APs:

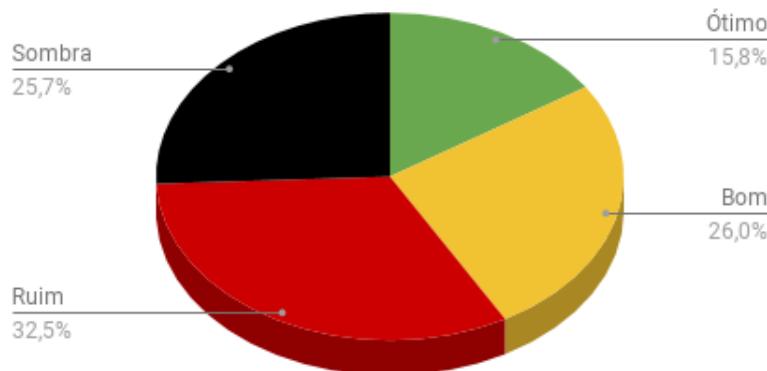
$-25 dBm$



```
FC global best: 2.089e+02
```

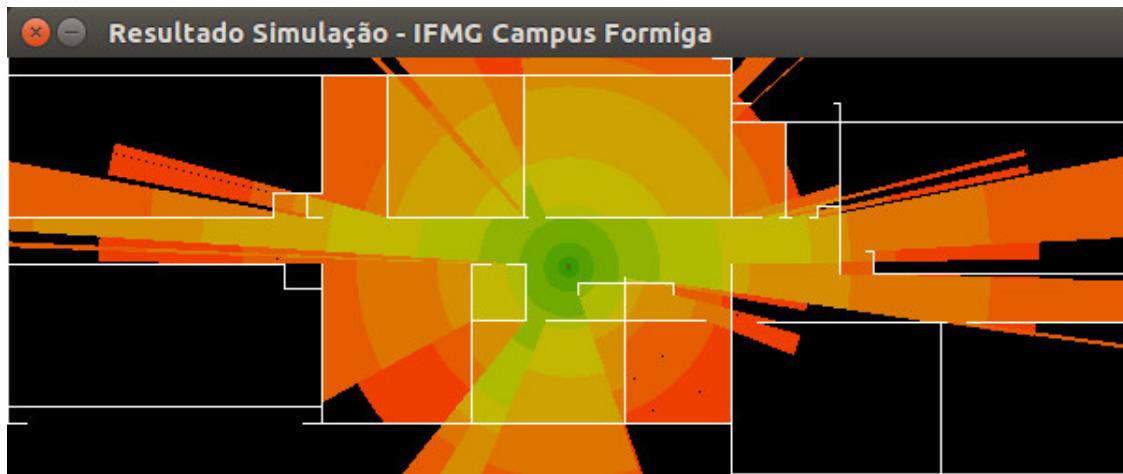
COBERTURA DE SINAL WI-FI:

   Figura 25 –   forma de gráfico de pizza do resultado dado para a otimização de um *accesspoints* no bloco A.



Fonte: Elaboração do autor.

Figura 26 – Simulação da propagação de sinais de microondas no bloco C utilizando 1 AP.



Fonte: Elaboração do autor.

50.89% com boa cobertura (sinal forte)

49.11% de zonas de sombra (abaixo da sensibilidade)

Cobertura por FAIXAS de intensidade de sinal

sinal Otimo 11.4%

sinal Bom 14.5%

sinal Ruim 25.1%

Na Figura 27 também pode ser visto como a zona de sombra esta presente com

a maior parte da simulação, o que fica notável que utilizando apenas um *access point* se torna impossível obter uma boa qualidade do sinal todas as salas de ambos os piso do bloco C.

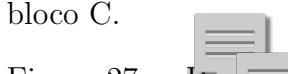
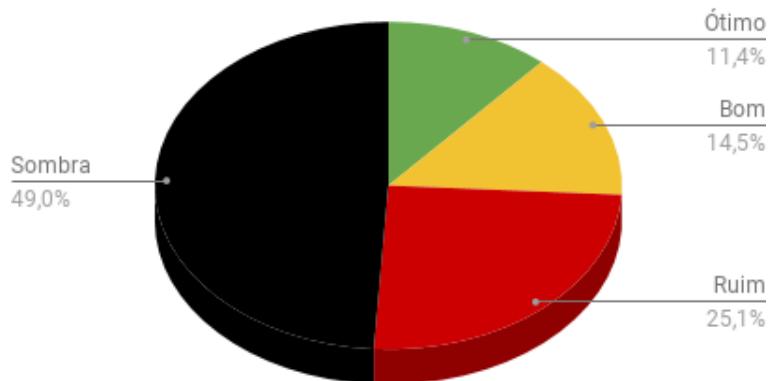


Figura 27 – Representação em gráfico de pizza do resultado dado para a otimização de um *access points* no bloco C.



Fonte: Elaboração do autor.



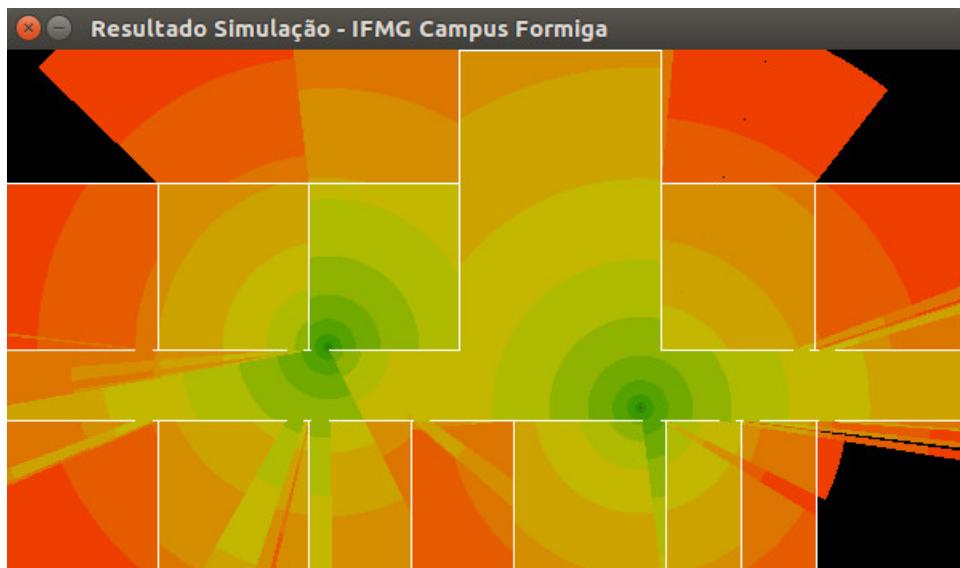
Como visto na Figura 24 e na Figura 26, quando utilizada a busca pelo ponto ótimo com um *access point*, ele tendeu a permanecer ao centro da planta. Com o resultado dado para esta simulação com um *access point*, os cômodos mais ao centro são bem atendidos com o sinal *Wi-Fi*, mas ao contrário dos mais afastados, não irão usufruir de uma boa qualidade de sinal e esse é retirado da apresentação gráfica e dos cálculos da avaliação da função objetivo. O resultado toma essa característica por não realizar uma verificação das áreas subjetivamente mais importantes. Caso tivesse essa funcionalidade implementada, a metaheurística poderia fazer com que o *access point* cobrisse as áreas mais críticas do ponto de vista da utilização do ambiente pelas pessoas.

5.3 Wi-Fi Placement para 2 ou mais APs

Feita a propagação do sinal *wireless* para um *access point* nos bloco A, foi adaptada, recalibrada e avaliada a implementação da simulação com dois *access points*. A Figura 28 mostra como foi o resultado da simulação do *Wi-Fi* no piso 2 do bloco A.

Como pode ser visto, a heurística pode então dar, como o resultado, dois pontos que visualmente aparecam ser muito bons. Na Figura 28 pode ser visto que os dois *access points* foram alocados pela metaheurística com uma distância de folga, fazendo com que possam aproveitar a propagação no ambiente. Com a energia sugerida pelo *Simulated Annealing*, utilizando um *access point* transmitindo a -25 dB, pode-se também

Figura 28 – Simulação da propagação de sinais de microondas no bloco A utilizando 2 APs.



Fonte: Elaboração do autor.

notar que algumas partes das salas nos extremos não estão recebendo uma qualidade de sinal como na sala em que os *client points* estão instalados, mas mesmo assim, a maior parte da área está recebendo sinal, havendo poucas zonas onde a potência recebida é inferior à sensibilidade do equipamento.

A seguir é possível visualizar o resumo da execução do algoritmo:

```
$ python PlacementAPs.py
```

Simulando ambiente com:

600 x 325 pixels

Escala de simulação:

1 px : 0.0800 metros

Quantidade de APs:

2

Potencia de cada APs:

-25 dBm



Fc global best: 6.112e+02

COBERTURA DE SINAL WI-FI:

91.12% com boa cobertura (sinal forte)

8.88% de zonas de sombra (abaixo da sensibilidade)

Cobertura por FAIXAS de intensidade de sinal

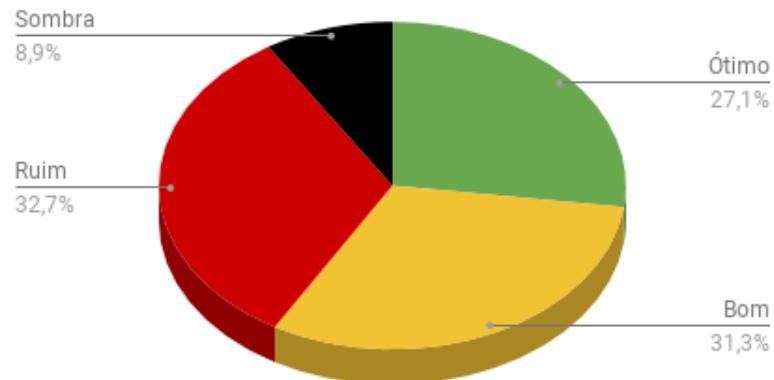
sinal Otimo 27.1%

sinal Bom 31.3%

sinal Ruim 32,7%

Na Figura 29 tem-se a representação da cobertura por faixa de intensidade do sinal [Wi-Fi icon] simulação de dois *access points* no segundo piso do bloco A. É válido ressaltar que utilizando dois *access points* na simulação, a cobertura é ainda maior, logo a fatia que indica a zona de sombra corresponde a apenas 8,9%, um valor inferior à metade do sombreamento quando havia somente um *access point*.

Figura 29 – Representação do resultado dado para a otimização de dois *access points* no segundo piso do bloco A.



Fonte: Elaboração do autor.

O mesmo teste foi realizado com a planta baixa do bloco C, porém agora que seja possível ter uma maior cobertura [Wi-Fi icon] área do prédio, a simulação foi realizado com um equipamento com potência de -17 dB. Na Figura 30, pode ser visto quais são os pontos sugeridos pelo *Simulated Annealing* na busca para a alocação dos mesmos.

Com o resultado dado pelo *Simulated Annealing* no bloco C, pode-se notar que foi possível cobrir praticamente toda a área. Tiveram algumas áreas com pontos cegos porém, podem ser consideradas de certa forma um mal necessário quando o objetivo é obter a cobertura do sinal. Um resultado ainda melhor pode ser encontrado se for feito um maior refinamento do *Simulated Annealing*. A saída dada pelo algoritmo é a seguinte:

```
$ python PlacementAPs.py
```

Simulando ambiente com:

600 x 223 pixels

Escala de simulação:

1 px : 0.0850 metros

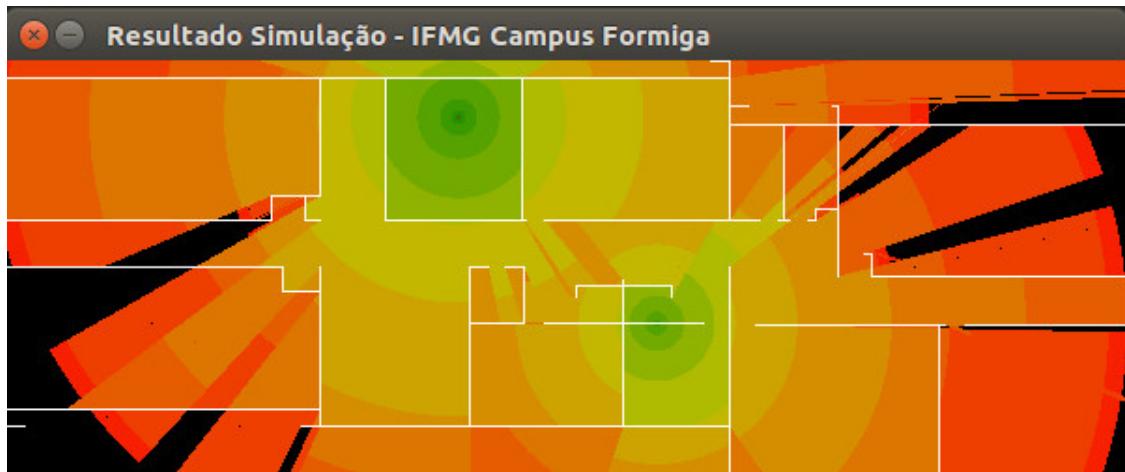
Quantidade de APs:

2

Potência de cada APs:

-17 dB

Figura 30 –  simulação da propagação de sinais de microondas no bloco C utilizando 2 APs.



Fonte: Elaboração do autor.

COBERTURA DE SINAL WI-FI:

89.76% com boa cobertura (sinal forte)

10.24% de zonas de sombra (abaixo da sensibilidade)

Cobertura por FAIXAS de intensidade de sinal

sinal Otimo 28.4%

sinal Bom 24.8%

sinal Ruim 36.6%



Com Figura 31 tem-se a representação da cobertura por faixa de intensidade do sinal na simulação de dois *access points* em ambos os pisos do bloco C. É notável que  indo dois *access points* na simulação, a zona de sombreamente caiu de 49,0% na simulação com um *access point* para 10,2%, ou seja, um aproveitamento de 38,8% a mais da cobertura, adicionando apenas mais um equipamento.

A busca utilizando três *access points* também foi realizada. O resultado da propagação do sinal pode ser visto na Figura 32 utilizando equipamentos com potência de transmissão de -25 dBm.



Os valores da saída são os seguintes:

```
$ python PlacementAPs.py
```

Simulando ambiente com:

600 x 223 pixels

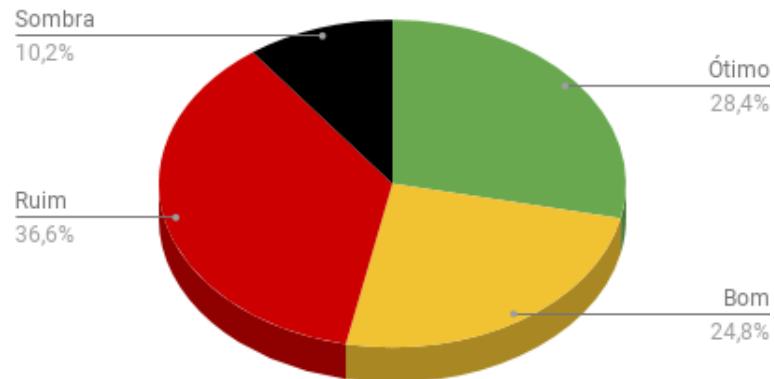
Escala de simulação:

1 px : 0.0850 metros

Quantidade de APs:

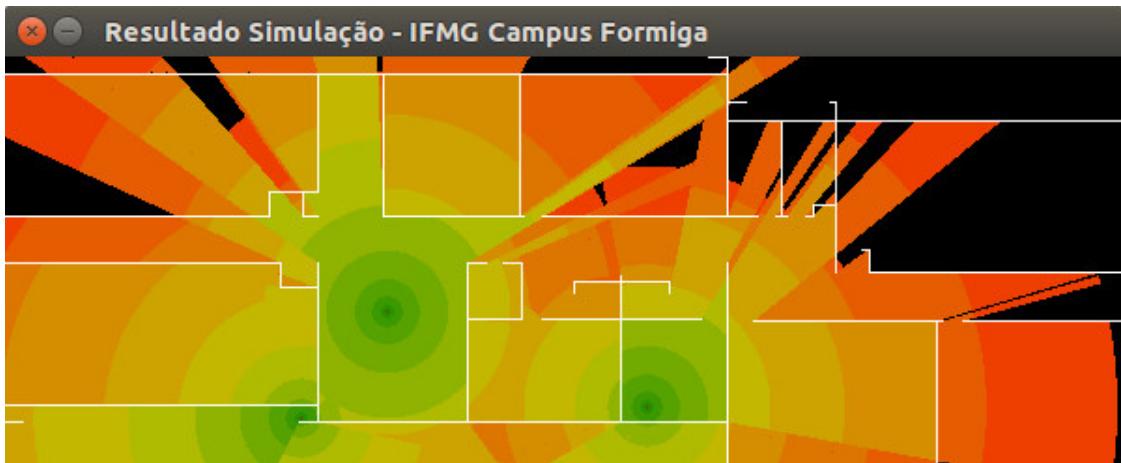
3

Figura 31 – Representação do resultado obtido para a otimização de dois accesspoints no bloco C.



Fonte: Elaboração do autor.

Figura 32 – Simulação da propagação de sinais de microondas no bloco C utilizando 3 APs com potência de -25 dB.



Fonte: Elaboração do autor.

Potencia de cada APs:

-25 \$dB\\$

 FC global best : 5.169 e+02

COBERTURA DE SINAL WI-FI:

81.69% com boa cobertura (sinal forte)

18.31% de zonas de sombra (abaixo da sensibilidade)

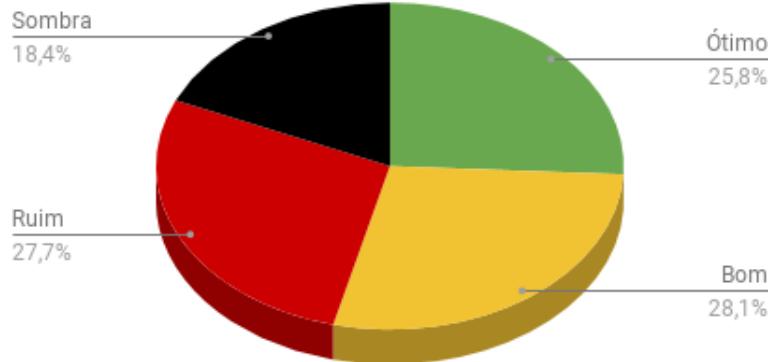
Cobertura por FAIXAS de intensidade de sinal

sinal Otimo	25.8%
sinal Bom	28.1%
sinal Ruim	27.7%



Com Figura 33 tem-se a representação da cobertura por faixa de intensidade do sinal na simulação de três *access points* em ambos os pisos do bloco C. Ao utilizar-se três *access points* na simulação, a zona de sombreamento aumentou, se comparada com a simulação de dois *access points*; a cobertura de sinal ótimo também foi menor, resultado que pode ser explicado pelo fato da solução da metaheurística, no momento da simulação, ter ficado presa em um ótimo local e não ter conseguido obter um resultado melhor. O resultado da simulação não foi o esperado, uma vez que esperava-se um aumento significativo da solução cujo número de *access points* era maior.

Figura 33 – Representação do resultado para a otimização de três *access points* no bloco C.



Fonte: Elaboração do autor.



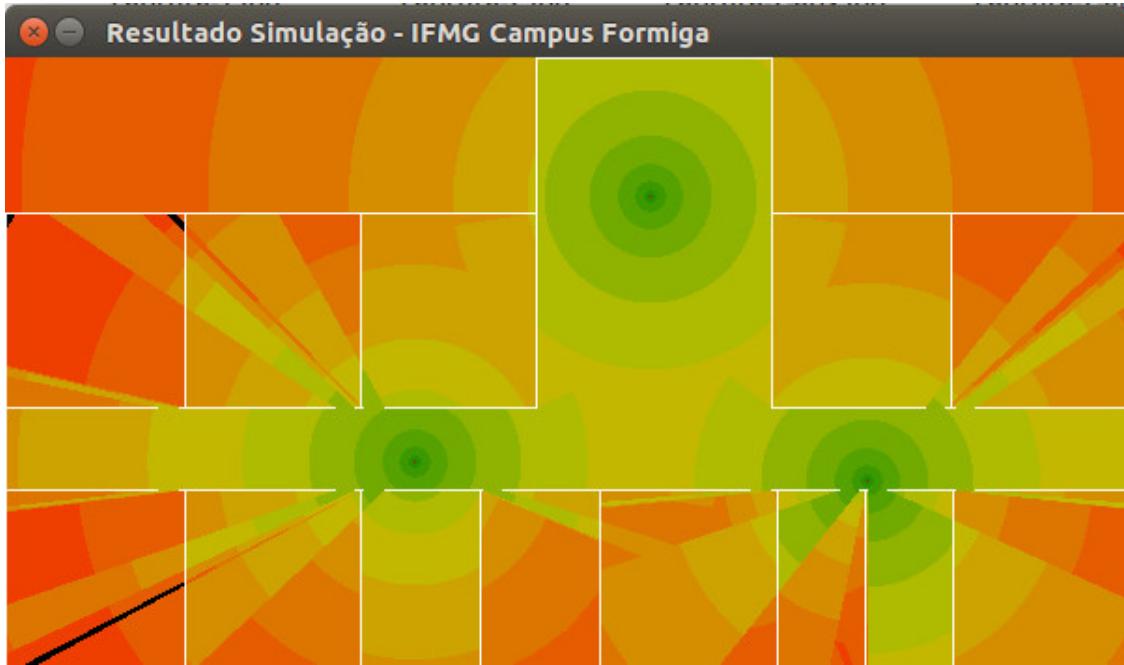
Na Figura 34 é possível ver a propagação do sinal utilizando 3 *access points* no bloco A com equipamentos transmitindo a -25 dB.

Os valores dados na saída para a simulação da Figura 34 com três *access points* no bloco A são os seguintes:

```
$ python PlacementAPs.py
```

Simulando ambiente com:	600 x 325 pixels
Escala de simulação:	1 px : 0.0800 metros
Quantidade de APs:	3
Potência de cada APs:	-25 dB

Figura 34 – Simulação da propagação de sinais de microondas no bloco A utilizando 3 APs com potência de -25 dB.



Fonte: Elaboração do autor.



FC global best : 6.974 e+02

COBERTURA DE SINAL WI-FI:

99.74% com boa cobertura (sinal forte)

0.26% de zonas de sombra (abaixo da sensibilidade)

Cobertura por FAIXAS de intensidade de sinal

sinal Ótimo 32.7%

sinal Bom 36.6%

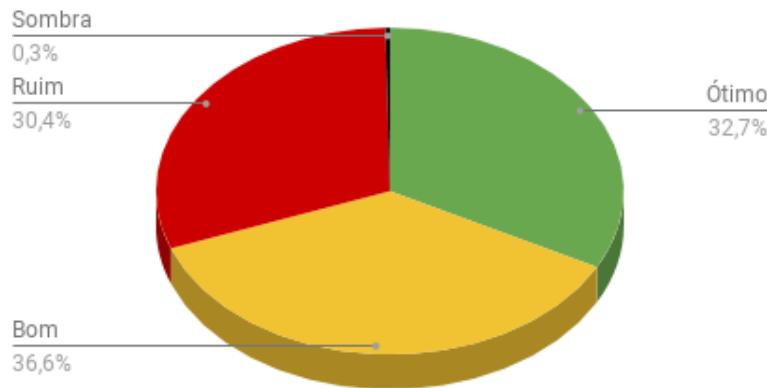
sinal Ruim 30.4%

Com Figura 35 tem-se a representação da cobertura por faixa de intensidade do sinal na simulação de três *access points* para o segundo piso do bloco A.

É claro que ao utilizar-se três *access points* na simulação, a zona de sombra teve-se a zero. Nenhum outro resultado das simulações para o bloco A mostrado anteriormente obteve um resultado em que a zona de sombra fosse tão menor. A fatia que representa o sinal ótimo não teve um grande aumento, mas se comparado aos resultados anteriores, foi significativo. No entanto, com o valor mínimo da zona de sombra, é possí-

vel garantir que toda a área do prédio seja coberta com o [redacted]. Ao ser aplicado em um projeto para a instituição, fica evidente que o acréscimo de equipamentos gerará um custo desnecessário.

Figura 35 – Representação do resultado para a otimização de três *accesspoints* no bloco A.



Fonte: Elaboração do autor.

Após as demonstrações feitas neste capítulo, fica claro que a ferramenta desenvolvida neste trabalho auxilia na escolha e na decisão da quantidade e da alocação de *access points* que podem ser utilizados para a cobertura das áreas desejadas. Todavia, a ferramenta desenvolvida neste trabalho pode passar por aprimoramentos que auxiliarão na [redacted]agem do ambiente, fazendo o uso, cada vez mais, da verossimilhança, aproximando [mas] a simulação mais da realidade. No entanto, se comparado aos *softwares* comerciais, o *software* desenvolvido neste trabalho executa uma das principais funções, sendo a mais relevante, o *Wireless AP Placement*, sem custo financeiro e de código fonte livre. No capítulo seguinte é feita uma alusão aos possíveis trabalhos futuros, que se executados poderão aproximar o *software* desenvolvido das versões pagas disponíveis no mercado.

6 TRABALHOS FUTUROS

Para trabalhos futuros, é necessária a realização de tratamento da interferência entre canais *Wi-Fi* no caso de utilização de mais de quatro *access points* em um mesmo ambiente, além de uma coleta de dados mais abrangente para obtenção de parâmetros mais precisos o modelo de regressão logístico (*NP-Log*) e tratamento de equipamentos de diferentes configurações e fabricantes. Também, sugere-se que seja utilizado mais de um modelo de propagação, complementando-se, visto que aparentemente um único modelo não realiza estimativas tão boas tanto para distâncias perto quanto longe. Portanto, sugere-se usar dois modelos de propagação, um para perto ($1 - 10m$) e outro para mais longe ($r_u = 100m$), numa abordagem híbrida. Uma outra abordagem poderia ser a aplicação de técnicas que envolvem simulação via *Ray-Tracing*, podendo ser incluída em trabalhos posteriores. Ainda, caso haja demanda, sugere-se a realização de simulação de propagação de sinais em ambiente 3D, com um maior aproveitamento dos recursos disponibilizados pela biblioteca CUDA, objetivando uma maior precisão da obtenção dos pontos cegos dentro de cômodos. Para se obter resultados que possibilitem uma melhor busca pelos pontos de acesso para dois ou mais *access points*, novas técnicas deverão ser utilizadas para o cálculo da função objetivo. Também é sugerido que outras metaheurísticas sejam implementadas e seus resultados comparados.

7 CONSIDERAÇÕES FINAIS

A ferramenta desenvolvida neste trabalho de conclusão de curso se torna útil e prática, uma vez que não havia disponível, no mercado, um *software* livre, gratuito e de código-fonte aberto para a realização de *Wireless AP Placement*. Nesse sentido, o *software* desenvolvido neste trabalho (disponibilizado gratuitamente) é capaz de (i) receber como entrada uma representação espacial 2D do ambiente (planta-baixa de um piso do edifício), podendo também ser informada a posição inicial dos *access points* como parâmetros adicionais para o algoritmo; (ii) realizar a simulação da propagação dos sinais de micro-ondas dos *access points Wi-Fi* pelas dependências dos blocos A e C do IFMG *campus* Formiga; (iii) aplicar a metaheurística computacional para explorar o espaço de soluções do posicionamento de *access points*; (iv) fornecer como saída a proposta de novo(s) posicionamento(s) dos *access points* visando ampliar a cobertura do sinal *Wi-Fi* para aquele ambiente.

Na versão final, o *software* é capaz de receber como entrada uma representação do ambiente, realizando a simulação de propagação dos sinais considerando a atenuação nas paredes e no espaço livre. A metaheurística buscou maximizar cada vez mais a cobertura do sinal *Wi-Fi* nas dependências do IFMG *campus* Formiga. Assim, também foi possível construir um modelo de propagação que se adequasse às dependências do IFMG *campus* Formiga, a partir da representação do ambiente fornecida pelo engenheiro da instituição. Além disso, é possível realizar a simulação da propagação do sinal *wireless* de *access points Wi-Fi* através das dependências do *campus*, informando ao *software* quantos *access points* se deseja alocar. Quando houver necessidade de ampliar, mesmo que temporariamente, a rede para a acomodação de novos pontos de acesso para telecomunicação, será fácil decidir os melhores locais a alocação dos equipamentos. Com a solução desenvolvida, se torna viável ter uma rede *wireless*, fazendo com que a cobertura de sinal possa abranger boa parte da área necessária, provendo uma qualidade de serviço satisfatória aos que fazem uso da rede.

Como mencionado anteriormente, a ferramenta desenvolvida foi disponibilizada de forma gratuita, utilizando Licença Pública Geral GNU — GPL no site [github.com¹](https://github.com/SamuelTerra22). Para reforçar sua relevância, ressalta-se que ele possibilita testar disposições de *access points* sem o custo operacional de fisicamente movê-los, de maneira a propor uma disposição espacial dos *access points*, que forneça uma maior cobertura e intensidade de sinal dentro do ambiente simulado.

¹ <<https://github.com/SamuelTerra22>>

Referências

- AARTS, E.; KORST, J. Simulated annealing and boltzmann machines. New York, NY; John Wiley and Sons Inc., 1988. Citado na página 68.
- ALMERS, P. et al. Survey of channel and radio propagation models for wireless mimo systems. *EURASIP Journal on Wireless Communications and Networking*, Hindawi Publishing Corp., v. 2007, n. 1, p. 56–56, 2007. Citado na página 53.
- BANERJI, S.; Singha Chowdhury, R. On ieee 802.11: Wireless lan technology. *International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol. 3, No.4, August 2013*, v. 3, p. 64, 2013. ISSN 18395678. Citado na página 30.
- BATISTA, V. de L. D. Software livre para wireless site survey. 2017. Citado 2 vezes nas páginas 77 e 85.
- BOF, E. Segurança em Redes Wireless. 2010. Citado na página 30.
- BUDGETS, L. Log-Normal Fading, Link Budgets. 2013. Citado na página 37.
- DALSSOTO, F. M.; SOUZA, R. H. D.; BECKER, L. B. Data analysis module for wirelesshart network planning. In: IEEE. *Computing and Automation for Offshore Shipbuilding (NAVCOMP), 2013 Symposium on*. 2013. p. 23–28. ISBN 978-0-7695-5123-4. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6686109>>. Citado na página 29.
- FARBER, R. *CUDA application design and development*. [S.l.]: Elsevier, 2011. Citado na página 49.
- FRANCISCATTI, V. Segurança em redes sem fio. *Trabalho de Conclusão de Curso-Curso de Especialização em Redes de Computadores e Comunicação de Dados, Universidade Estadual de Londrina, Londrina, Paraná*, 2005. Citado na página 30.
- KASE, K. et al. Volume cadcw-complexes based approach. *Computer-Aided Design*, Elsevier, v. 37, n. 14, p. 1509–1520, 2005. Citado na página 66.
- KIRK, D. et al. Nvidia cuda software and gpu parallel computing architecture. In: *ISMM*. [S.l.: s.n.], 2007. v. 7, p. 103–104. Citado na página 48.
- KIRKPATRICK, S. et al. Optimization by simulated annealing. *science*, Washington, v. 220, n. 4598, p. 671–680, 1983. Citado na página 67.
- LAARHOVEN, P. J. V.; AARTS, E. H. Simulated annealing. In: *Simulated annealing: Theory and applications*. [S.l.]: Springer, 1987. p. 7–15. Citado 2 vezes nas páginas 43 e 68.
- LAM, S. K.; PITROU, A.; SEIBERT, S. Numba: A llvm-based python jit compiler. In: ACM. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. [S.l.], 2015. p. 7. Citado na página 50.

- LEDESMA, S.; RUIZ, J.; GARCIA, G. Simulated annealing evolution. In: *Simulated Annealing-Advances, Applications and Hybridizations*. [S.l.]: InTech, 2012. Citado na página 44.
- LENTZ, C. 802.11 b wireless network visualization and radiowave propagation modeling. *Hanover, NH: Dartmouth College*, 2003. Citado na página 53.
- LUO, M. Indoor radio propagation modeling for system performance prediction. 2013. Disponível em: <<https://hal.archives-ouvertes.fr/tel-00937481/document>>. Citado 4 vezes nas páginas 32, 34, 35 e 36.
- MAGNUN. *A História do Python*. 2014. Disponível em: <<http://mindbending.org/pt/a-historia-do-python>>. Citado 2 vezes nas páginas 46 e 47.
- MARQUES, F. P. J. A. A regulação do acesso wireless à internet no brasil. Observatório de Economia e Comunicação (Obscom) da Universidade Federal de Sergipe (UFS), 2006. Citado na página 25.
- MARSCHALLINGER, R. A voxel visualization and analysis system based on AutoCAD. *Computers and Geosciences*, v. 22, n. 4, p. 379–386, 1996. ISSN 00983004. Citado na página 66.
- MATHURANATHAN. *Log Distance Path Loss or Log Normal Shadowing Model*. 2013. Disponível em: <<https://www.gaussianwaves.com/2013/09/log-distance-path-loss-or-log-normal-shadowing-model/>>. Citado na página 35.
- NAJNUDEL, M. Estudo de propagação em ambientes fechados para o planejamento de wlans. *Rio de Janeiro*, v. 136, 2004. Citado 2 vezes nas páginas 29 e 53.
- NAVIGATOR, A. *Anaconda The Most Popular Python Data Science Platform*. 2017. Disponível em: <<https://docs.anaconda.com/>>. Citado na página 51.
- NETO, B. B.; SCARMINIO, I. S.; BRUNS, R. E. *Planejamento e otimização de experimentos*. [S.l.]: Ed. da UNICAMP, 1996. Citado na página 69.
- NUMBA. *Compilation JIT functions*. 2017. Disponível em: <<http://numba.pydata.org/numba-doc/0.17.0/reference/compilation.html#numba.jit>>. Citado na página 50.
- NVIDIA. *CUDA para Medicina*. 2017. Disponível em: <http://www.nvidia.com.br/object/cuda_medical_br.html>. Citado na página 49.
- NVIDIA. *CUDA programação paralela facilitada*. 2017. Disponível em: <http://www.nvidia.com.br/object/cuda_home_new_br.html>. Citado na página 48.
- NVIDIA. *Develop, Optimize and Deploy GPU-accelerated Apps*. 2017. Disponível em: <<https://developer.nvidia.com/cuda-toolkit>>. Citado na página 49.
- R. Battiti M. Brunato et al. Optimal wireless access point placement for location-dependent services. *October*, n. October, p. 1–12, 2004. Citado na página 29.
- RAPPAPORT, T. S. Comunicações Sem fio Princípios e Práticas. v. 2, p. 412p, 102p, 2009. Citado 8 vezes nas páginas 31, 32, 33, 34, 35, 36, 62 e 64.

- RENSBURG, J. J. van; IRWIN, B. Wireless Network Visualization Using Radio Propagation Modelling. *Proceedings of Information Security South Africa Conference*, 2005. Citado na página 53.
- RIGO, A. Representation-based just-in-time specialization and the psyco prototype for python. In: ACM. *Proceedings of the 2004 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation*. [S.l.], 2004. p. 15–26. Citado na página 50.
- RIVERA, A. D. Redes de equipamentos sem fio de uso pessoal: comparação de tecnologias emergentes e análise de tendências. *Teses.Usp.Br*, 2010. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/3/3142/tde-10012011-084232/en.php>>. Citado na página 30.
- ROSSUM, G. van. *The History of Python*. 2009. Disponível em: <<http://python-history.blogspot.com.br/2009/01/introduction-and-overview.html>>. Citado 2 vezes nas páginas 46 e 47.
- RUBINSTEIN, M. G.; REZENDE, J. F. Qualidade de serviço em redes 802.11. *XX Simpósio Brasileiro de Redes de Computadores (SBRC2002)*, 2002. Citado na página 26.
- SANDEEP, A. et al. Wireless network visualization and indoor empirical propagation model for a campus wi-fi network. *World Academy of Science, Engineering and Technology*, v. 42, p. 730–734, 2008. Citado na página 53.
- SILVA, J. G. R. Introdução à linguagem python. *Universidade Federal de Juiz de Fora*, 2016. Citado na página 46.
- SULAIMAN, H. A. et al. Wireless Network Visualization in 3D Virtual Environment Framework. p. 170–174, 2012. Citado na página 53.
- THAKUR, A.; BANERJEE, A. G.; GUPTA, S. K. A survey of CAD model simplification techniques for physics-based simulation applications. *CAD Computer Aided Design*, Elsevier Ltd, v. 41, n. 2, p. 65–80, 2009. ISSN 00104485. Disponível em: <<http://dx.doi.org/10.1016/j.cad.2008.11.009>>. Citado na página 66.
- TORLAK, M. Path Loss. 2016. Disponível em: <<https://www.utdallas.edu/~torlak/courses/ee4367/lectures/lectureradio.p>>. Citado na página 31.
- VALENZUELA, R. A ray tracing approach to predicting indoor wireless transmission. *IEEE 43rd Vehicular Technology Conference*, p. 214–218, 1993. Citado 2 vezes nas páginas 37 e 38.
- XIE, M. *Indoor radio propagation modeling for system performance prediction*. Tese (Doutorado) — INSA de Lyon, 2013. Citado na página 36.
- YUN, Z.; ISKANDER, M. F. Ray tracing for radio propagation modeling: Principles and applications. *IEEE Access*, v. 3, p. 1089–1100, 2015. ISSN 21693536. Citado na página 53.