UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Matheus Dal Mago

OTIMIZAÇÃO DE PARÂMETROS DE UMA REDE LORA ATRAVÉS DE ALGORITMOS GENÉTICOS

Santa Maria, RS
2017

Matheus Dal Mago

OTIMIZAÇÃO DE PARÂMETROS DE UMA REDE LORA ATRAVÉS DE ALGORITMOS GENÉTICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Engenharia de Computação**.

ORIENTADOR: Prof. Carlos Henrique Barriquello

Santa Maria, RS
2017

Matheus Dal Mago

OTIMIZAÇÃO DE PARÂMETROS DE UMA REDE LORA ATRAVÉS DE ALGORITMOS GENÉTICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Engenharia de Computação**.

Aprovado em 13 de dezembro de 2017:

Carlos Henrique Barriquello, Dr. (UFSM)
(Presidente/Orientador)

Rodrigo da Silva Guerra, Dr. (UFSM)

Alexandre Campos, Dr. (UFSM)

Santa Maria, RS
2017

RESUMO

OTIMIZAÇÃO DE PARÂMETROS DE UMA REDE LORA ATRAVÉS DE ALGORITMOS GENÉTICOS

AUTOR: Matheus Dal Mago
ORIENTADOR: Carlos Henrique Barriquello

A tecnologia LoRa destaca-se por possibilitar a conexão de um grande número de dispositivos dispostos há diversos quilômetros da estação base, com um pequeno consumo de energia e baixo custo de manutenção. Para tal, cada dispositivo possui um conjunto de configurações que definem o funcionamento do mesmo, como frequência de operação, largura de banda, dentre outros. Uma correta escolha de tais configurações é essencial para um bom funcionamento da rede. Na tecnologia LoRa, a comunicação é comumente controlada pelo protocolo LoRaWAN, que define a forma como a rede troca mensagens. Neste protocolo, existem diretivas que alteram algumas configurações dos dispositivos conectados, de forma a otimizar o funcionamento do mesmo, seja aumentando a taxa de transmissão de dados ou diminuindo o consumo de energia. Porém, esta otimização leva em conta apenas o dispositivo em questão, e não o efeito que essa alteração vai ter no restante da rede. Este trabalho visa comparar o desempenho de uma rede configurada através do protocolo LoRaWAN com uma rede configurada através de um algoritmo genético, que tem a capacidade de avaliar a rede como um todo e de buscar soluções aproximadamente ótimas. O algoritmo genético foi implementado em Python e o desempenho da rede foi mensurado através de simulações baseadas em trabalhos já existentes. Foi possível observar que, para um pequeno número de dispositivos conectados na rede, o protocolo LoRaWAN possui um desempenho bastante aceitável. Porém, quando a complexidade da rede aumenta, soluções geradas por algoritmos genéticos são superiores às obtidas pelo protocolo.

Palavras-chave: LoRa. LoRaWAN. Configuração. Algoritmos Genéticos.

ABSTRACT

PARAMETERS OPTIMIZATION OF A LORA NETWORK THROUGH GENETIC ALGORITHMS

AUTHOR: Matheus Dal Mago
ADVISOR: Carlos Henrique Barriquello

The LoRa technology stands out for allowing the connection of a large number of devices allocated several kilometers away from the base station, with low energy consumption and maintenance costs. To do so, each device has a set of settings that define how it will work, like carrier frequency, bandwidth, and so on. Choosing the right configuration is essential for a well working network. In the LoRa technology, the communication is usually controlled by the LoRaWAN protocol, that defines how the network will exchange messages. In this protocol, there are instructions that change some configurations of the connected devices, optimizing its performance by either enlarging the transmission's data rate or lowering the energy consumption. However, this optimization considers just the device itself, and not how it will affect the rest of the network. This project compares the performance of a network that is configured by the LoRaWAN protocol with the performance of a network configured by a genetic algorithm, which is capable of evaluating the network as a whole and search solutions tending to the optimum. The genetic algorithm was implemented in Python and the network performance was measured using simulations based on existing works. It was possible to see that, for a small amount of devices connected on the network, the LoRaWAN protocol has a good enough performance. Still, when the network complexity gets larger, solutions generated by the genetic algorithm are superior than the ones generated by the protocol.

Keywords: LoRa. LoRaWAN. Configuration. Genetic Algorithms.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BW	*Bandwidth*
CF	*Carrier Frequency*
$Chirp$	*Compressed High Intensity Radar Pulse*
CR	*Coding Rate*
CSS	*Chirp Spread Spectrum*
CSV	*Comma-separated Values*
DER	*Data Extraction Rate*
DR	*Data Rate*
ID	*Identifier*
$IEEE$	*Institute of Electrical and Electronics Engineers*
FEC	*Forward Error Correction*
GA	*Genetic Algorithms*
IoT	*Internet of Things*
$LoRa$	*Long Range*
$LoRaWAN$	LoRa *Wide Area Network*
$LPWAN$	*Low Power Wide Area Network*
MAC	*Media Access Control*
$NBIoT$	*Narrow Band IoT*
$RSSI$	*Received Signal Strength Indication*
SF	*Spreading Factor*
SNR	*Signal-to-Noise Ration*
TP	*Transmission Power*

SUMÁRIO

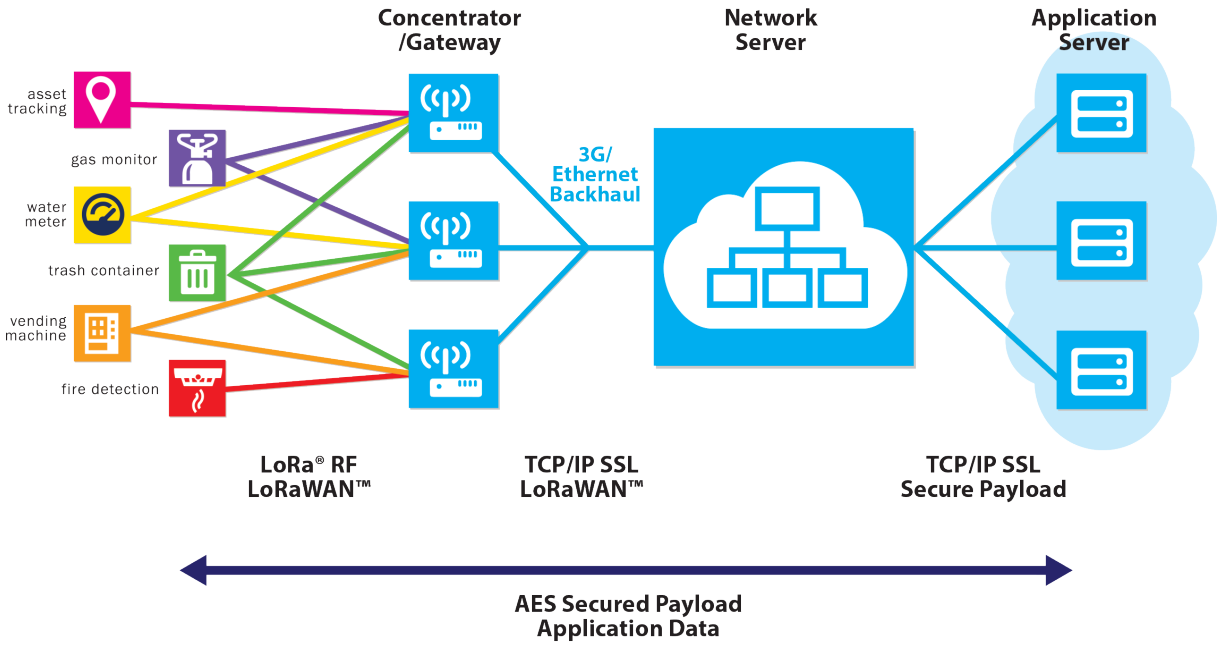
1 INTRODUÇÃO

Redes de comunicação que possuem um alto alcance e baixo consumo de energia estão ganhando grande espaço no mercado, principalmente devido ao crescimento do que é chamado de Internet das Coisas (IoT). Pesquisas apontam que até 2020 existirão cerca de 50 bilhões de dispositivos conectados à internet, sendo grande parte dispositivos remotos, como sensores de temperatura, pressão, luminosidade, dentre outros (EVANS, 2011). Apesar deste valor possuir contradições, como discutido por Nordrum (2016), uma quantidade de dispositivos em cerca de 30 bilhões continua a ser bastante expressivo. Para satisfazer as necessidades requisitadas por tais dispositivos, surgiram as redes de comunicação conhecidas como LPWAN. Estas permitem a conexão de centenas ou até milhares de dispositivos dispostos a diversos quilômetros da estação base, e ainda possuindo uma longa autonomia de bateria, o que é propício para o uso de sensores inteligentes. O Anexo A mostra uma análise de características de redes LPWAN comparadas com ZigBee e GSM. Tecnologias LPWAN como SIGFOX, NB-IoT e LoRa, apesar de possuírem algumas características diferentes, concorrem para ganhar mercado, que vem crescendo a um passo acelerado.

Uma das tecnologias LPWAN que mais se destaca em seu meio é o LoRa. Uma rede LoRa é composta pelos dispositivos finais, chamados de nós, as estações base, conhecidas como *gateways*, servidor de rede e servidor de aplicação. Os nós enviam seus dados diretamente para o *gateway*, se utilizando da rede LoRa, seguindo uma topologia em forma de estrela. Este, por sua vez, é responsável por repassar os dados recebidos para o servidor de rede através da internet (protocolo IP), garantindo a integridade e segurança dos dados entregues. O servidor de rede, por sua vez, elimina pacotes duplicados (caso mais de um *gateway* tenha recebido o mesmo pacote) e disponibiliza os dados para o servidor de aplicação. Neste último, os dados são decriptografados e utilizados para sua finalidade. Esta estrutura é mostrada na Figura 1.1.

Cada nó da rede possui um conjunto de configurações que deve ser definido, que afeta tanto a comunicação direta do dispositivo com o *gateway* assim como o comportamento do restante da rede, uma vez que existem determinadas configurações que podem fazer com que haja colisão de pacotes enviados ao mesmo tempo por diferentes nós, fazendo com que um dos pacotes, ou até mesmo ambos, não sejam recebidos pelo *gateway*. O controle sobre tais configurações é feito pelo dispositivo final, porém o protocolo de comunicação implementado na rede pode solicitar que o nó altere alguma configuração, se o mesmo estiver habilitado para tal. O protocolo mais utilizado em conjunto com uma rede LoRa é o LoRaWAN. Este pode alterar, através

Figura 1.1 – Estrutura de uma rede LoRa



Fonte: Semtech (2017).

de comandos enviados pelo *gateway*, a taxa de transmissão de dados (*data rate*) dos nós da rede, de forma a otimizar o funcionamento da mesma e reduzir o consumo de energia dos dispositivos finais. Porém, tal protocolo analisa cada dispositivo individualmente, sem levar em consideração o efeito que a mudança da configuração de um nó vai ter nos demais dispositivos da rede. Desta forma, uma tentativa de otimizar o desempenho de um dispositivo pode comprometer parcialmente a comunicação do restante da rede, levando a uma menor taxa de sucesso na entrega dos dados e um maior consumo de energia.

Este trabalho visa analisar, através de simulações e algoritmos evolutivos, uma determinada rede de dispositivos LoRa estáticos e propor um conjunto de configurações para os mesmos, de forma a otimizar o funcionamento desta rede. Ainda, é esperado que este conjunto de configurações obtenha um melhor desempenho com relação ao protocolo LoRaWAN, devido ao fato de a rede ser analisada como um todo, levando em consideração a influência que a configuração de um dispositivo vai ter sobre os demais, e não apenas sobre aquele dispositivo individualmente, como acontece com o LoRaWAN. Dentre as técnicas voltadas à busca de soluções e otimização de problemas, os algoritmos genéticos são bastante conhecidos pela sua capacidade de gerar soluções criativas para problemas, convergindo ao sub-ótimo sem a necessidade de possuir um grande conhecimento sobre o espaço de busca, uma vez que estes “não são limitados por suposições [...] relativas à continuidade, existência de

derivadas, etc." (CARVALHO, 2009). Os algoritmos genéticos são baseados na teoria da Seleção Natural (DARWIN, 1859), fazendo analogia a genes, cromossomos, indivíduos e população, e foram amplamente estudados por Holland (1992). O requisito para seu funcionamento é a existência de uma forma de avaliar o resultado gerado, conhecida como função *fitness*. A escolha de utilizar algoritmos genéticos em vez de outros algoritmos evolutivos de otimização se deu principalmente pelo contato prévio do autor com estes, facilitando a implementação e a escolha de parâmetros do algoritmo.

Em sequência, neste trabalho, encontra-se a Revisão de Literatura, que é organizada de forma a fornecer o conhecimento básico para o entendimento da importância da correta configuração de parâmetros para os nós de uma rede LoRa, assim como demonstrar a forma como tais parâmetros são escolhidos atualmente e técnicas capazes de aperfeiçoar esta escolha. Na etapa seguinte, encontram-se os métodos utilizados durante a execução deste trabalho, assim como os procedimentos adotados em busca da obtenção de melhores resultados. Em sequência, os resultados obtidos são apresentados e discutidos. Por fim constam as Considerações Finais e Referências.

2 REVISÃO DE LITERATURA

Nesta etapa serão abordados os conceitos básicos sobre o funcionamento e modulação de uma rede LoRa, assim como as configurações dos nós e a influência de cada no funcionamento da rede. Em complemento, serão mostradas as otimizações que o protocolo LoRaWAN já implementa, visando melhorar o funcionamento da rede LoRa, e o simulador LoRaSim, que é capaz de extrair índices de qualidade de funcionamento da mesma. Ainda, neste mesmo capítulo, serão apresentadas as principais características dos algoritmos genéticos e os benefícios que a escolha deste trazem para a solução do problema.

2.1 LORA

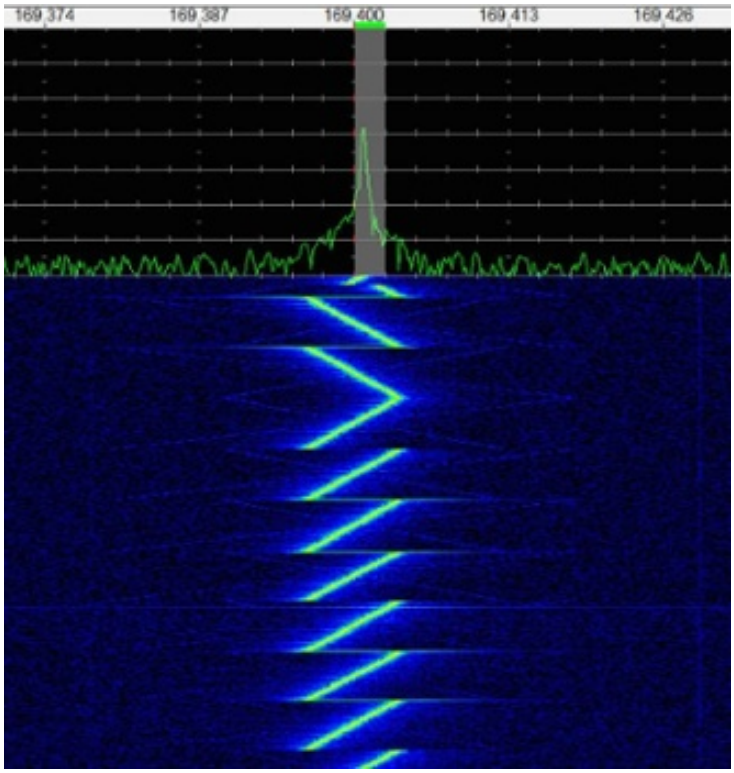
LoRa é uma tecnologia de comunicação sem fio baseada em radiofrequência, que opera na faixa sub-GHz, operando com uma técnica de modulação de espalhamento espectral proprietária da empresa Semtech, que deriva do *Chirp* (*Compressed High Intensity Radar Pulse*) *Spread Spectrum* (CSS). As propriedades principais de uma rede LoRa são: longo alcance, alta robustez, resistência ao efeito Doppler e baixo consumo de energia. Um receptor LoRa consegue decodificar uma transmissão até 19,5dB abaixo do nível de ruído (BOR et al., 2016).

Para codificar a mensagem enviada, uma transmissão LoRa envia, previamente, uma série de *up-chirps*, que são utilizados pelo receptor para detectar e sincronizar a transmissão (RAY, 2015). Cada *chirp* é um sinal transmitido que inicia em uma frequência menor e vai aumentando a frequência linearmente (*up-chirp*) ou vice-versa (*down-chirp*), explorando toda a largura de banda do sinal. O fim da mensagem prévia é sinalizado pelo *down-chirp*, como pode ser visto na Figura 2.1.

O módulo LoRa passa então a transmitir dados codificados através de uma mudança repentina na frequência dos *chirps*, como mostrado na Figura 2.2. O padrão distinto do comportamento dos *chirps* é o que faz com que a tecnologia LoRa possa decodificar sinais com potência muito baixa. Ainda, é possível decodificar sinais simultâneos que, mesmo operando na mesma frequência, utilizem um *chirp rate* diferente, isto é, a frequência de cada *chirp* varia de forma mais ou menos acentuada. Uma análise mais aprofundada pode ser encontrada em Semtech Corporation (2015).

Um transceptor LoRa possui cinco parâmetros que podem ser configurados e que influenciam no consumo de energia dos dispositivos, alcance total da rede, taxa de dados transmitidos com sucesso e no tempo de transmissão. Esses parâmetros

Figura 2.1 – Demonstração da mensagem prévia: *up-chirps* e *down-chirps*



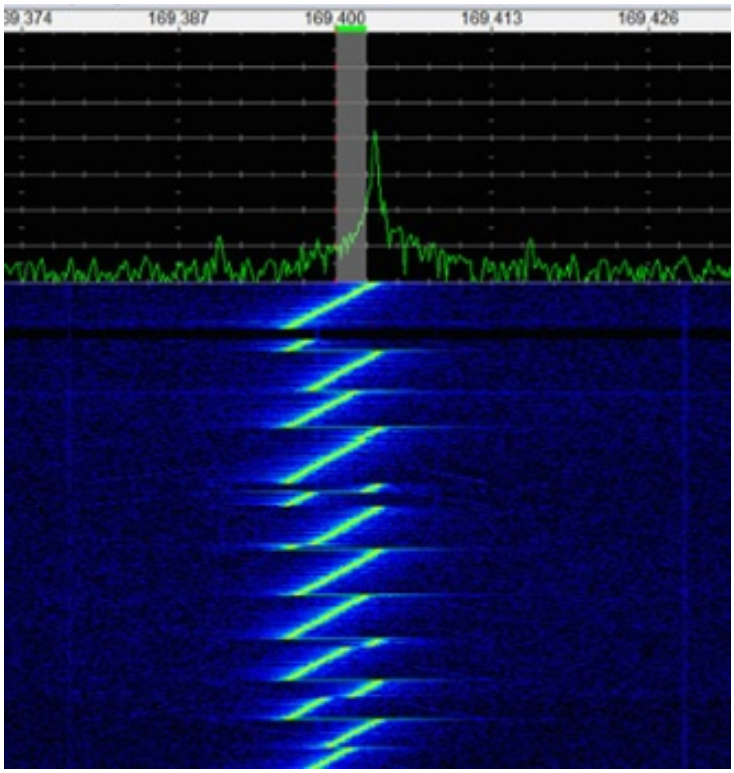
Fonte: Ray (2015).

são: potência de transmissão (TP), frequência (ou canal) de transmissão (CF), largura de banda (BW), fator de espalhamento (SF) e taxa de codificação (CR). Ainda, a união entre os parâmetros BW e SF determina a taxa de transmissão de dados (DR). Bor e Roedig (2017) mostram que existem mais de 6.720 combinações de configurações possíveis para um único dispositivo LoRa. Sendo assim, para uma rede com 100 dispositivos, tem-se um espaço de busca de 6.720^{100} possibilidades. Ainda, Bor e Roedig (2017) retratam que existem escolhas de configurações que podem variar o consumo de energia de um dispositivo em mais de 100 vezes, isto é, uma boa escolha nas configurações pode resultar em um dispositivo ter seu tempo de vida útil da bateria 100 vezes maior do que um que possui uma má escolha.

2.1.1 Potência de Transmissão (TP)

A potência, em dBm, com que o sinal é transmitido. Em um rádio LoRa típico, esse valor varia entre -4dBm e 20dBm. Uma potência de transmissão maior aumenta claramente o alcance do sinal, porém também aumenta o consumo de energia do dispositivo. A potência máxima em que um dispositivo pode transmitir é limitada pela regulamentação local.

Figura 2.2 – *Payload* codificado em meio aos *chirps*



Fonte: Ray (2015).

2.1.2 Frequência de Transmissão (CF)

A frequência de transmissão dos dispositivos, também conhecida como canal. As frequências centrais de transmissão podem adotar os valores de 433MHz, 868MHz ou 915MHz, dependendo da regulamentação regional de onde a rede será implantada. A regulamentação brasileira prevê o uso de uma rede LoRa na frequência central de 915MHz, possuindo canais distribuídos na faixa de 902MHz a 928MHz. Um *gateway* possui a capacidade de decodificar simultaneamente dados recebidos em 8 canais diferentes.

2.1.3 Largura de Banda (BW)

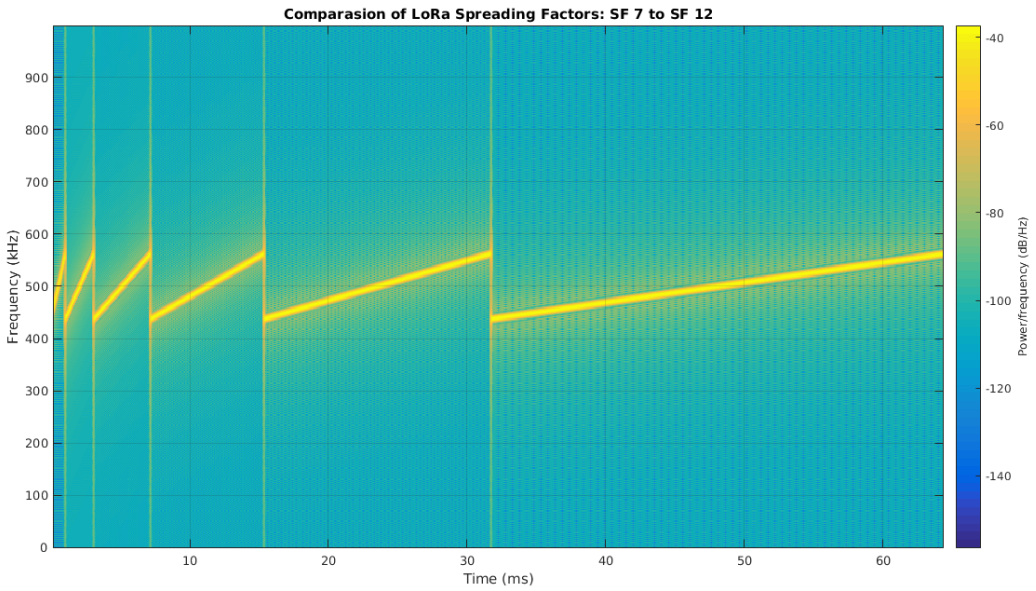
A largura de banda indica o intervalo de frequências que cada *chirp* vai variar. “No LoRa, a taxa de *chirps* [...] é igual à largura de banda (um *chirp* por segundo por Hertz da largura de banda).” (AUGUSTIN et al., 2016). Ao se aumentar a largura de banda, temos uma velocidade de transferência de dados maior, porém tornamos a rede mais suscetível a interferências. Os valores de largura de banda comumente utilizados em uma rede LoRa são de 125kHz, 250kHz e 500kHz.

2.1.4 Fator de Espalhamento (SF)

O fator de espalhamento mostra o quanto cada *chirp* é “espalhado” com relação ao tempo. “Valores maiores de SF espalham mais o sinal no tempo a fim de aplicar mais energia em um único *chirp*, permitindo recepção com sucesso a distâncias mais longas” (POP et al., 2017). A Figura 2.3 mostra uma comparação entre a variação da frequência de diferentes fatores de espalhamento com relação ao tempo. O SF é representado por valores de 7 a 12, demonstrando a razão entre símbolos e *chirps*. O aumento de uma unidade no Fator de Espalhamento faz com que o *chirp* duplique o tempo no ar. Sinais com valores diferentes de SF são considerados ortogonais, e portanto podem ser decodificados pelo *gateway* de forma independente. A taxa efetiva de transmissão de dados é dada pela Equação 2.1, sendo esta dependente dos valores da Largura de Banda e do Fator de Espalhamento.

$$Rb = SF.\frac{BW}{2^{SF}} \tag{2.1}$$

Figura 2.3 – Representação de Diferentes SFs



Fonte: Ghoslya (2017).

2.1.5 Taxa de Codificação (CR)

A tecnologia LoRa utiliza uma técnica de *Forward Error Correction* (FEC), a qual adiciona algumas informações redundantes no pacote, de forma a adicionar proteção contra interferências repentinas. A taxa de dados a ser adicionada pelo FEC é definida pela Taxa de Codificação, que pode adotar os valores de 4/5, 4/6, 4/7 ou 4/8. “Um valor maior de CR oferece mais proteção, porém aumenta o tempo no ar” (BOR et al., 2016).

2.1.6 Decodificação no *Gateway*

Um *gateway* LoRa possui a capacidade de receber com sucesso pacotes em 8 diferentes canais simultaneamente, sem que um interfira na recepção do outro. Ainda, pacotes de possuem SF diferente serão recebidos mesmo que ambos estejam no mesmo canal, pelo fato de cada SF ser ortogonal aos outros. Em complemento, ainda que dois sinais sejam recebidos simultaneamente no mesmo canal e com o mesmo SF, o sinal que foi recebido com maior potência normalmente ainda pode ser decodificado.

2.2 LORAWAN

LoRaWAN é um protocolo de acesso ao meio (MAC) que é comumente utilizado para operar sobre uma rede LoRa. De acordo com LoRa Alliance (2015), “LoRaWAN define o protocolo de comunicação e a arquitetura do sistema para a rede, enquanto o *layer* físico LoRa possibilita um *link* de comunicação de longo alcance.” A especificação e padronização do protocolo é mantida pela LoRa Alliance, uma associação sem fins lucrativos que visa impulsionar o mercado de IoT globalmente.

Em uma rede LoRaWAN, um dispositivo não está relacionado a um *gateway* em específico. Em vez disso, quando o mesmo deseja transmitir alguma informação, os dados são enviados sem endereçar nenhum destinatário. Se mais de um *gateway* receber o mesmo pacote, o servidor de rede se encarrega de identificar os dados duplicados e envia uma confirmação de recebimento (*acknowledge*) através do *gateway* que recebeu o pacote com maior RSSI. Ainda, é de responsabilidade do protocolo LoRaWAN garantir a segurança e confiabilidade dos dados transmitidos, assim como assegurar que a comunicação não infrinja nenhuma legislação regional, no que diz respeito à frequência de operação, intensidade do sinal transmitido e tempo de banda utilizado. Para tanto, o LoRaWAN define uma série de regiões, onde cada uma possui um conjunto de parâmetros e limites específicos de operação. As regiões mais utilizadas são a US915, que se refere aos Estados Unidos, e opera na faixa de 915MHz,

EU868, que opera na Europa na faixa de 868MHz, e AU915, na Austrália, também na faixa de 915MHz. Ainda não existe nenhuma definição oficial no protocolo LoRaWAN quanto sua operação no Brasil. A documentação completa sobre os parâmetros regionais pode ser encontrada em LoRa Alliance Technical Comittee (2017).

2.2.1 *Data Rate* (DR)

O protocolo LoRaWAN define a velocidade de comunicação utilizada através de um valor chamado de *Data Rate*. Este é definido pela combinação da Largura de Banda e do Fator de Espalhamento, indicando diretamente a taxa de dados transmitida por segundo. Um dispositivo deve sempre utilizar um valor de *Data Rate* definido para realizar sua comunicação. Para se aumentar o valor do *Data Rate*, é preciso aumentar a Largura de Banda ou diminuir o Fator de Espalhamento, o que fará com que os dados sejam transmitidos em um tempo menor, com a consequência de que o *gateway* necessitará de um sinal com maior potência para poder decodificar a mensagem. As definições de valores de cada *Data Rate* depende da regulamentação regional. O Quadro 2.1 relaciona valores de DR com suas respectivas combinações de BW e SF, indicando ainda a taxa de transmissão efetiva em bits por segundo para cada DR de acordo com os parâmetros regionais dos Estados Unidos, visto que ainda não existe uma definição de valores designados para o Brasil. Os valores de *Data Rate* de 5 a 7 e o valor 14 são reservados para uso futuro, e por isso não foram representados neste quadro.

Quadro 2.1 – Definição dos valores de *Data Rate* para o padrão regional US915.

Data Rate	SF	BW (kHz)	Taxa de Transmissão (b/s)
0	10	125	980
1	9	125	1760
2	8	125	3125
3	7	125	5470
4	8	500	12500
8	12	500	980
9	11	500	1760
10	10	500	3900
11	9	500	7000
12	8	500	12500
13	7	500	21900

Fonte: Adaptado de LoRa Alliance Technical Comittee (2017).

2.2.2 *Adaptative Data Rate* (ADR)

O protocolo LoRaWAN prevê uma forma de otimizar os parâmetros de configuração de um dispositivo conectado à rede através do que é chamado de *Adaptative Data Rate*. Este mecanismo, que pode ou não ser ativado pelo dispositivo, permite que o Fator de Espalhamento, Largura de Banda e Potência de Transmissão de um dispositivo sejam alterados durante a operação do mesmo, de forma a reduzir o consumo de energia e aumentar a taxa de transmissão de dados deste.

Quando um dispositivo possui o ADR ativado, o *gateway* então passa a monitorar os pacotes recebidos deste, armazenando uma lista de SNRs das últimas 20 mensagens recebidas. O valor máximo de SNR armazenado do dispositivo em questão é então subtraído por uma margem de segurança de 10dB (de forma a evitar que um dispositivo varie seu *Data Rate* para um valor muito próximo ao limite de recepção do *gateway*, correndo o risco de perder conexão) e é comparado com o valor de SNR mínimo requerido para cada *Data Rate*. Caso o valor da SNR resultante ainda seja maior do que o valor mínimo requerido pelo DR acima, o DR do dispositivo é então aumentado, fazendo com que as transmissões do mesmo sejam feitas em menos tempo, consequentemente diminuindo a energia total consumida pelo dispositivo. Quando o *Data Rate* atingir seu valor máximo, a Potência de Transmissão do dispositivo é então diminuída em 3dBm a cada passo. Caso a média de SNR de um dispositivo esteja abaixo do valor mínimo requerido para sua *Data Rate*, a Potência de Transmissão do dispositivo é aumentada, novamente em passos de 3dBm. Se o dispositivo parar de receber confirmações de recebimento de pacotes do *gateway*, ele automaticamente aumenta sua Potência de Transmissão e diminuiu seu *Data Rate*, até a conexão ser restabelecida. O Quadro 2.2 mostra quais os valores de SNR requeridos para cada DR. Só alguns valores de DR são representados aqui pois o Quadro 2.2 se refere a uma implementação do mecanismo de ADR para o padrão EU868, sendo que ainda não existe uma definição do *Adaptative Data Rate* para as demais regiões.

Quadro 2.2 – Valor de SNR mínimo requerido para cada *Data Rate* - EU868.

Data Rate	SNR (dB)
DR4	-10
DR3	-12,5
DR2	-15
DR1	-17,5
DR0	-20

Fonte: Adaptado de Telkamp (2017).

2.3 LORASIM

Para verificar o desempenho de uma rede LoRa composta por um grande número de dispositivos, o uso de simuladores se torna indispensável, devido ao custo e mão de obra que seria necessário para obter e instalar fisicamente tal quantidade de dispositivos. O LoRaSim é um simulador de rede LoRa proposto em Bor et al. (2016), implementado e disponibilizado de forma aberta por Voigt e Bor (2017), que é capaz de simular colisões de pacotes e analisar o comportamento da rede física. Pop et al. (2017) apresenta ainda uma extensão do LoRaSim, a qual foi chamada de LoRaWANSim, que inclui funcionalidades, simulando também o protocolo LoRaWAN. Este último, porém, não é disponibilizado para uso.

O simulador LoRaSim foi escrito utilizando a linguagem de programação Python e a biblioteca de simulação de eventos SimPy. Nele, é possível definir o número de dispositivos LoRa que se deseja simular, a quantidade de *gateways* da rede, com que periodicidade cada dispositivo vai realizar o envio de dados e o tempo máximo da simulação. Ainda, o simulador implementa 6 experimentos diferentes, numerados de 0 a 5. No experimento 0, todos os dispositivos são simulados no mesmo canal de comunicação e utilizando o menor *Data Rate*, ou seja, com o maior Fator de Espalhamento e a menor Largura de Banda. O experimento 1 estende o experimento 0 implementando uma escolha aleatória do canal de comunicação. No experimento 2, o simulador escolhe o maior *Data Rate* (menor Fator de Espalhamento e maior Largura de Banda). Já no experimento 3, as configurações são otimizadas para o maior Data Rate possível em que o dispositivo ainda consegue se comunicar com o *gateway*, baseado na distância entre os mesmos. O experimento 4 utiliza a configuração padrão inicial definida pelo protocolo LoRaWAN, e o experimento 5 estende o experimento 3, otimizando também a potência de transmissão. Com exceção do experimento 5, todos os dispositivos utilizam a mesma potência de transmissão.

Ao se definir os parâmetros de entrada, o LoRaSim posiciona os dispositivos em um ambiente bidimensional e simula o envio de pacotes. O Anexo B ilustra o posicionamento de 50, 100, 200 e 500 dispositivos, que ocorre de forma aleatória, mostrando o *gateway* ao centro, em cor verde, e os nós distribuídos pelo espaço, em cor azul. O momento de início da transmissão de cada dispositivo é obtido através de um valor aleatório, que segue uma distribuição exponencial, simulando o comportamento de dispositivos reais onde o momento de envio de dados dos dispositivos não é precisamente previsto. A biblioteca SimPy é utilizada para organizar o início da transmissão como eventos discretos, onde cada dispositivo fica em estado de espera até que sua transmissão inicie. O LoRaSim simula, então, a transmissão dos pacotes, através de modelos de propagação e perdas, e analisa se ocorreram colisões entre pacotes ou não, gerando um índice de dados transmitidos com sucesso, chamado de

DER.

2.3.1 Propagação

O cálculo de potência do sinal recebido pelo *gateway* é feito de forma simplificada, e é dado pela Equação 2.2, onde Prx é a potência recebida, em dB, Ptx é a potência transmitida pelo dispositivo, em dB, GL é um valor definido como constante, que representa uma combinação de ganhos e perdas como ganho de antenas, perdas em conexões e circuito RF, dentre outros, e Lpl representa a perda de propagação do sinal no caminho.

$$Prx = Ptx + GL - Lpl \tag{2.2}$$

A perda de propagação do sinal no caminho Lpl é calculada através do modelo logaritmo para perdas, dado pela Equação 2.3, sendo este um modelo empírico cujas variáveis são obtidas através de testes práticos. Nesta equação, Lpl(d0) representa a perda média do caminho em um ponto de referência, γ é o expoente de perda no caminho, d0 é a distância do ponto de referência e Xσ é uma variável aleatória Gaussiana, com média 0 e desvio padrão σ^2 .

$$Lpl(d) = Lpl(d0) + 10\gamma.log(\frac{d}{d0}) + X\sigma \tag{2.3}$$

Para o LoRaSim, estes valores foram obtidos experimentalmente. Como mostrado em Bor et al. (2016), para uma distância de referecia d0 em 40m, foi calculado o valor de Lpl(d0) em 127,41dB, $\gamma = 2,08$ e $\sigma = 3,57$. Porém, nas simulações, um valor de $\sigma = 0$ é utilizado, caso contrário algumas transmissões poderiam não ser recebidas pelo *gateway*, comprometendo os resultados das simulações. A sensitividade do receptor para cada Fator de Espalhamento e Largura de Banda também foi mensurada experimentalmente, e é demonstrada pelo Quadro 2.3.

2.3.2 Colisão

Para determinar se existiu colisão na transmissão de dois ou mais pacotes, o LoRaSim verifica se um conjunto de condições são satisfeitas. A primeira análise feita pelo simulador é se existe mais de um dispositivo transmitindo ao mesmo tempo.

Quadro 2.3 – Sensitividade (em dBm) de um receptor LoRa para cada SF e BW.

	Bandwidth (kHz)		
SF	125	250	500
7	-126,50	-124,25	-120,75
8	-127,25	-126,75	-124,00
9	-131,25	-128,25	-127,50
10	-132,75	-130,25	-128,75
11	-134,50	-132,75	-128,75
12	-133,25	-132,25	-132,25

Fonte: Adaptado de Bor et al. (2016).

Para tanto, quando cada dispositivo inicia sua transmissão, ele é adicionado a uma lista, quem mantém todos os dispositivos transmitindo naquele momento. Passado o *airtime* da transmissão, o dispositivo é removido da lista. Portanto, toda vez que essa lista tiver mais de um elemento, é possível que haja colisão na transmissão, e as outras condições devem ser analisadas.

A próxima verificação feita pelo LoRaSim é se existe colisão na Frequência de Transmissão. Para que não haja colisão, deve haver uma diferença absoluta mínima entre as duas frequências sendo analisadas, que depende da largura de banda sendo utilizada. Segundo Bor et al. (2016), para o módulo de comunicação Semtech SX1272, “A mínima diferença de frequência tolerada [. . .] é 60kHz para uma largura de banda de 125kHz, 120kHz para uma largura de banda de 250kHz e 240kHz para uma largura de banda de 500kHz”. O Anexo D mostra o código Python utilizado para fazer esta verificação.

Após as verificações anteriores, caso ainda exista a possibilidade de haver colisão na transmissão, o simulador verifica o valor do Fator de Espalhamento dos pacotes em questão. Dois pacotes enviados ao mesmo tempo e na mesma Frequência de Transmissão podem ser decodificados pelo *gateway* sem que haja perda de dados se utilizarem Fatores de Espalhamento diferentes. Isto é, existe colisão em transmissões apenas quando o SF destas forem iguais. Por fim, quando dois pacotes são transmitidos ao mesmo tempo e utilizando os mesmos valores de CF e SF, o que chegar no *gateway* com uma potência maior ainda pode ser decodificado, desde que a diferença absoluta entre as potências dos sinais recebidos seja maior do que um valor determinado. Para o LoRaSim, esse valor é de 6dBm.

2.3.3 Métricas

Para possibilitar uma avaliação da rede simulada, o LoRaSim extrai alguns índices de funcionamento da mesma, contendo a quantidade de pacotes que foram transmitidos pelo dispositivo final, quantos foram recebidos com sucesso pelo *gateway*, uma contagem do número de colisões que ocorreram durante toda a simulação e ainda a energia que foi consumida durante o período de simulação. Estes resultados são gravados em um arquivo, permitindo a plotagem dos dados para melhor observação. Ainda, através das métricas extraídas, é calculado o DER, que indica a porção dos pacotes que foram transmitidos com sucesso. Cabe notar que o LoRaSim não simula o reenvio de um pacote que foi perdido, descartando o mesmo, o que não ocorreria em uma aplicação real.

2.4 ALGORITMOS GENÉTICOS

As primeiras simulações computacionais envolvendo algoritmos evolutivos foram desenvolvidas por biólogos nos anos 50, baseando-se na ideia da teoria da evolução para otimizar soluções através de sistemas evolutivos. Na década de 1970, John Holland propôs os algoritmos genéticos, que são algoritmos evolutivos baseados em conceitos da seleção natural. Segundo Gabriel e Delbem (2008), os algoritmos genéticos são considerados robustos pois, em geral, os mesmos produzem soluções de qualidade independentemente da escolha dos parâmetros iniciais. Ainda, os algoritmos genéticos permitem o uso amplo de computação paralela, e conseguem encontrar soluções apenas com a avaliação de cada solução gerada, sem possuir um conhecimento amplo sobre o problema.

2.4.1 Funcionamento de um Algoritmo Genético

Em algoritmos genéticos, uma solução é, basicamente, uma cadeia de valores numéricos (binários ou inteiros). Cada valor é aqui chamado de gene, e a cadeia como um todo é conhecida como cromossomo. Um valor numérico do cromossomo (ou um conjunto destes, dependendo da implementação) representa um parâmetro na solução proposta pelo algoritmo. “Cada cromossomo pode ser imaginado como sendo um ponto no espaço de busca de soluções” (MITCHELL, 1998). A cada iteração (ou geração) do algoritmo, as piores soluções são eliminadas da população, enquanto as melhores são combinadas para gerar soluções novas. O algoritmo segue nesse laço até que seja atingido um critério de parada definido, que pode ser quando um

número máximo de gerações tenha passado ou quando uma solução ótima tenha sido atingida. Existem diversas formas de se implementar cada etapa de um algoritmo genético, e estudos elencando as melhores para cada tipo de aplicação. O fluxograma no Anexo C demonstra os passos comumente utilizados.

Neste trabalho, serão elencadas abordagens simples, de forma a melhor esclarecer os conceitos básicos do assunto. Em uma análise sintética, pode-se considerar que o funcionamento de um algoritmo genético se dá através da geração de uma população inicial, que é um conjunto de indivíduos aleatórios (cada indivíduo representa uma solução para o problema), que então passam por uma avaliação, onde os melhores são selecionados. Essas soluções definidas como melhores passam então pelo processo de *crossover*, que faz um cruzamento dos indivíduos (também de forma aleatória), gerando novas soluções. Ainda, uma pequena taxa de mutação é adicionada, permitindo que todo o espaço de busca seja explorado. O processo então se repete, sendo cada iteração chamada de geração.

2.4.1.1 População Inicial

O primeiro passo na utilização de algoritmos genéticos é a criação de um conjunto de soluções aleatórias para o problema. Cada uma dessas soluções é chama de indivíduo, e o conjunto de todas é conhecido como população. A escolha do tamanho da população inicial é de extrema importância, pois quanto mais indivíduos forem criados aleatoriamente, haverá uma melhor exploração do espaço de busca, havendo uma maior possibilidade de se encontrar mínimos globais. Por outro lado, o custo computacional necessário aumenta com o aumento da quantidade de indivíduos existentes.

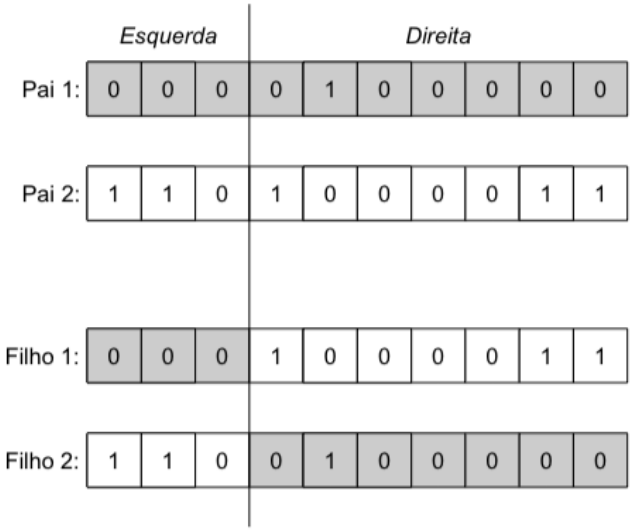
2.4.1.2 Pontuação e Seleção

Nesta etapa é realizada a seleção dos “indivíduos mais adaptados”. Em outras palavras, é realizada uma avaliação de toda a população, através do que é chamado de função *fitness*, onde uma pontuação é atribuída para cada indivíduo. É essencial, para um bom desempenho do algoritmo genético, que esta seja cuidadosamente projetada ilustrando o quão bem a solução em questão resolve o problema pretendido, uma vez que influenciará diretamente na escolha dos indivíduos que serão utilizados para dar continuidade ao processo. Os indivíduos com melhor pontuação são selecionados para continuar fazendo parte da população, enquanto os outros são descartados.

2.4.1.3 Crossover e mutação

Após os melhores indivíduos terem sido selecionados, eles são organizados aleatoriamente em pares. É feito então um *crossover*, também chamado de recombinação, dos cromossomos de cada par, onde a forma mais simples é através da divisão em 1 ponto, que é ilustrado na Figura 2.4. “Nessa operação, seleciona-se aleatoriamente um ponto de corte nos cromossomos, dividindo este em uma partição à direita e outra à esquerda do corte” (GABRIEL; DELBEM, 2008). Os cromossomos resultantes passam então por um processo de mutação, que consiste na edição arbitrária de uma quantidade de genes aleatória, o que “assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois [...] altera-se levemente a direção da busca” (CARVALHO, 2009). O Anexo E ilustra o processo de *crossover* e mutação. Os cromossomos resultantes são então utilizados para criar novos indivíduos, que são adicionados à população. Em sequência, o processo de avaliação, seleção, *crossover* e mutação se repete, até que o critério de parada seja alcançado.

Figura 2.4 – *Crossover* de 1 ponto



Fonte: Gabriel e Delbem (2008).

3 METODOLOGIA

Considerando-se que ao definindo os parâmetros de configuração de cada ponto de uma rede, pode-se obter, através de simulações, dados de desempenho da mesma, torna-se possível a aplicação de algoritmos que buscam um conjunto de parâmetros de forma a trazer um melhor funcionamento da rede. Para o desenvolvimento deste trabalho, o simulador LoRaSim (VOIGT; BOR, 2017) foi modificado, de forma a tornar as simulações mais acessíveis, tendo suas entradas e saídas se dando através de parâmetros. Este capítulo descreve as alterações feitas no simulador, assim como sua utilização, o desenvolvimento do algoritmo genético utilizado e a interação entre ambos. A versão final do código utilizado pode ser encontrado no repositório do GitHub, em https://github.com/dalmago/lorasim.

3.1 LORASIM

3.1.1 Python

A linguagem de programação Python possui algumas diferenças significativas entre sua versão 2 e 3 (mais recente), principalmente no que se trata da sintaxe da linguagem e desempenho. O simulador LoRaSim foi escrito para utilizar Python 2, versão esta que ainda é suportada oficialmente, porém está se tornando obsoleta uma vez que a maior parte das novas aplicações Python são escritas utilizando a versão 3. De tal modo, para uma melhor manutenibilidade do código, o mesmo foi portado para suportar Python 3.

3.1.2 Disposição dos Nós

Em uma simulação usual, o LoRaSim toma como entrada a quantidade de dispositivos que se deseja simular e os distribui aleatoriamente dentro de um raio de distância máxima da estação base. Porém esse comportamento é indesejável, pois o mesmo invalidaria a comparação entre duas simulações, já que o arranjo de dispositivos simulados seria sempre diferente. Para contornar este problema, arquivos CSV foram criados, contendo uma lista de dispositivos e suas coordenadas no plano bidimensional, e o LoRaSim foi editado para suportar tal arquivo como entrada. Cada ar-

quivo CSV possui uma quantidade especifica de dispositivos. Foram criados arquivos com 10, 15, 20, 25, 50, 100, 200, 500 e 1000 dispositivos. Desta forma, toda simulação executada utilizaria dispositivos dispostos na mesma posição, possibilitando a comparação entre simulações. O Apêndice A demonstra o arquivo CSV com uma lista de 10 dispositivos, onde a primeira coluna representa o ID do dispositivo, e a segunda e terceira coluna representam respectivamente as coordenadas nos eixos “x” e “y” do plano bidimensional.

3.1.3 Entrada de Parâmetros

Nas simulações, os parâmetros de configuração da rede são determinados pelo LoRaSim , de acordo com o experimento escolhido. Como deseja-se obter métricas de funcionamento de uma rede com configurações geradas por uma fonte externa (neste caso, um algoritmo genético), foi criado um novo experimento no LoRaSim que aceita uma lista de valores de Espalhamento Espectral e Largura de Banda como parâmetros opcionais. Deste modo, ainda é possível executar os experimentos normais do LoRaSim, mas agora com a possibilidade de fornecer para o mesmo os parâmetros que devem ser utilizados durante as simulações.

3.1.4 Extração de Dados

Os índices extraídos da simulação pelo LoRaSim são contadores de número de pacotes enviados e número de pacotes que sofreram colisão. A partir destes valores, é calculado o *Data Extraction Rate*, que expressa a fração de dados que foram transmitidos com sucesso. Ainda, o simulador calcula, através do tempo que cada pacote leva para ser transmitido e a potência do transmissor, a energia consumida por cada dispositivo e a energia total consumida pela rede. Porém, a saída destes resultados se dá através de arquivos CSV e *prints* na tela. Para evitar de buscar esses valores em arquivos salvos em disco, a rotina principal da simulação foi transformada em uma função que possui como valor de retorno o *Data Extraction Rate* e a energia consumida pela rede. Assim, algoritmos externos podem receber os índices das simulações sem a necessidade de buscar estes dados em arquivos salvos em disco, evitando uma carga desnecessária ao sistema.

3.2 ALGORITMOS GENÉTICOS

O algoritmo genético utilizado neste trabalho foi implementado na linguagem Python, devido à sua facilidade e capacidade de implementação em um curto período de tempo. O objetivo principal deste algoritmo genético é a geração de um conjunto de configurações para cada dispositivo da rede, onde que se busca uma melhor taxa de extração de dados e um menor consumo de energia. Para simplificar a análise, apenas os parâmetros SF (Fator de Espalhamento) e BW (Largura de Banda) foram utilizados, uma vez que são os que mais afetam a ocorrência de colisões. Ainda, como cada *gateway* possui capacidade de receber dados simultaneamente em 8 canais de comunicação diferentes, e um canal não causa interferência nos outros, a modelagem aqui utilizada para configuração da rede pode ser repetida para cada canal. Como critério de parada, definiu-se como sendo a obtenção de um indivíduo ótimo, que neste caso é uma rede com 100% da comunicação com sucesso, ou seja, DER igual a 1, ou caso o número máximo de gerações seja alcançado, que foi definido como sendo 100 gerações.

3.2.1 Indivíduos e População

Como se deseja gerar uma configuração para cada dispositivo e analisar a rede como um todo, cada indivíduo do algoritmo genético é composto por um conjunto de configurações que representa toda a rede. Desta forma, os dados providos por um único indivíduo são suficientes para simular toda a rede com a quantidade de dispositivos desejada, representando uma solução para a configuração da rede. Sendo assim, a população é um conjunto de possíveis soluções. Se deseja-se obter configurações para uma rede com 10 dispositivos, por exemplo, cada indivíduo deverá possuir 10 valores de SF e de BW. Ainda, se uma população de 10 indivíduos for gerada pelo algoritmo genético, ao total existirão 100 valores de SF e BW na população toda.

Em um momento inicial do trabalho, utilizou-se uma população de 100 indivíduos, porém, ao decorrer do mesmo, este valor foi aumentado para 300, em uma tentativa de explorar melhor o espaço de busca de soluções no início do algoritmo.

3.2.2 Genes e Cromossomos

Uma das formas mais comuns de se representar um cromossomo é através de uma cadeia de bits, onde determinados conjuntos de bits representam um gene em específico. Na implementação deste projeto na linguagem Python, utilizou-se de uma

lista de valores inteiros, porém sendo estes limitados a valores 0 ou 1, emulando uma cadeia de bits mesmo que estes valores ocupem mais espaço em memória. Cada dispositivo possui 2 genes: SF (Fator de Espalhamento) e BW (Largura de Banda). Foram empregados 6 “bits” (números inteiros) para representar ambos, onde os 4 mais significativos representam o gene de SF e os 2 menos significativos representam o de BW. A quantidade total de valores que compõem um cromossomo é de 6 “bits” multiplicado pela quantidade de dispositivos que se deseja simular. Por exemplo, ao criar uma rede que possui 10 dispositivos, será necessária uma lista com 60 valores “binários” para representar um cromossomo desta. Ainda, como o gene de SF pode assumir valores entre 7 e 12 apenas (limitação da tecnologia LoRa), cromossomos criados com genes fora dessa faixa não são considerados válidos. De forma semelhante, o gene de BW pode assumir 3 valores, sendo que cada um corresponde a um valor de Largura de Banda, conforme mostrado na Tabela 3.1.

Tabela 3.1 – Valor do Gene correspondente a cada BW.

Gene	BW (kHz)
0	125
1	250
2	500
3	-

Fonte: Autor.

3.2.3 *Fitness*

Como o LoRaSim considera um certo fator de aleatoriedade para indicar quando cada dispositivo vai iniciar a transmissão de dados, cada simulação obterá um valor de DER e energia diferente, mesmo quando simulando a mesma rede. Para não remover a aleatoriedade entre as simulações e correr o risco de obter uma solução que seja extrema (muito boa ou muito ruim), para cada conjunto de configurações sugerido pelo algoritmo genético, são realizadas 10 simulações. Os valores de DER e energia final são então obtidos através da média de todas as simulações. Para melhor aproveitar o poder computacional, foi utilizado a biblioteca *multiprocessing*, que permite separar tarefas em diferentes processos no sistema, paralelizando o processamento.

O valor do DER obtido das simulações afeta diretamente o consumo de energia da rede, pois quanto menos dados trafegarem com sucesso, mais tentativas de envio cada dispositivo fará, consequentemente o consumo de energia de toda a rede tende a aumentar. Sendo assim, para este trabalho, a função *fitness* tem se dado apenas pela avaliação do valor de DER das simulações. A metade da população com pior resultado

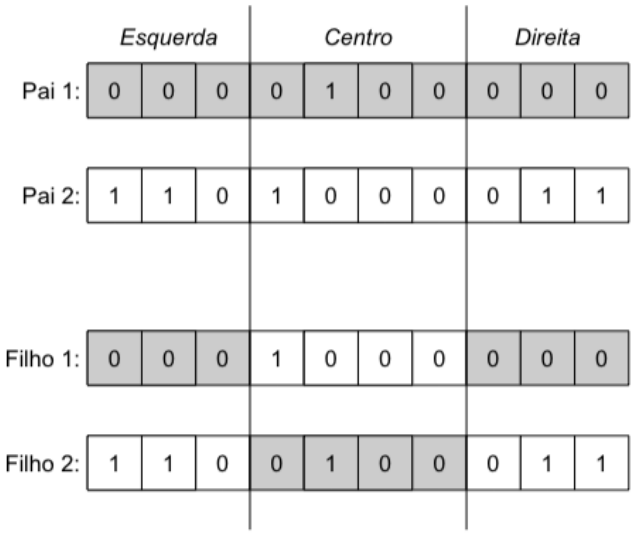
em sua *fitness* é eliminada, enquanto a outra metade é mantida e recombinada de forma a gerar um número de novos indivíduos igual à quantidade que foi removida, mantendo assim o tamanho da população.

3.2.4 *Crossover* e Mutação

Inicialmente, a técnica utilizada para cruzar dois elementos foi a de *crossover* de 1 ponto, já descrita anteriormente neste trabalho, sendo este ponto fixo na metade do cromossomo, ou seja, o indivíduo resultante do cruzamento de outros dois indivíduos pais receberá exatamente metade do cromossomo de cada um. Ainda, dois elementos pais são cruzados duas vezes, em ordens trocadas, assim dois indivíduos filhos são gerados. Para realizar mutação, aproximadamente 10% dos indivíduos gerados são selecionados aleatoriamente, e possuem N “bits” de seu cromossomo trocados, sendo N também aleatório e nunca maior que 10%. O Apêndice B demonstra o código fonte utilizado para realizar tal comportamento.

Em um segundo momento, ao decorrer do trabalho, a técnica de *crossover* utilizada foi modificada para a de 2 pontos, que é ilustrada na Figura 3.1, sendo estes fixos em 1/3 e 2/3 do cromossomo. Ainda, em um momento seguinte, a técnica foi alterada novamente para utilizar 2 pontos aleatórios do cromossomo.

Figura 3.1 – *Crossover* de 2 pontos



Fonte: Gabriel e Delbem (2008).

4 RESULTADOS

Para comparar a performance obtida pelos algoritmos genéticos, utilizou-se o experimento 3 do LoRaSim, que otimiza a configuração de cada dispositivo da rede de acordo com a distância do mesmo para a estação base, comportamento que se aproxima ao do protocolo LoRaWAN utilizando ADR. Ainda, cada nó da rede realiza o envio de cerca de 4 pacotes por minuto, e o tempo de cada simulação foi limitado em 1 minuto. Os algoritmos genéticos foram executados diversas vezes, e os resultados apresentados abaixo são uma compilação ddos melhores índices obtidos para cada situação. Podemos dividir os resultados em basicamente 3 etapas, onde mudanças significativas foram feitas no algoritmo genético com intuito de melhorar seu desempenho.

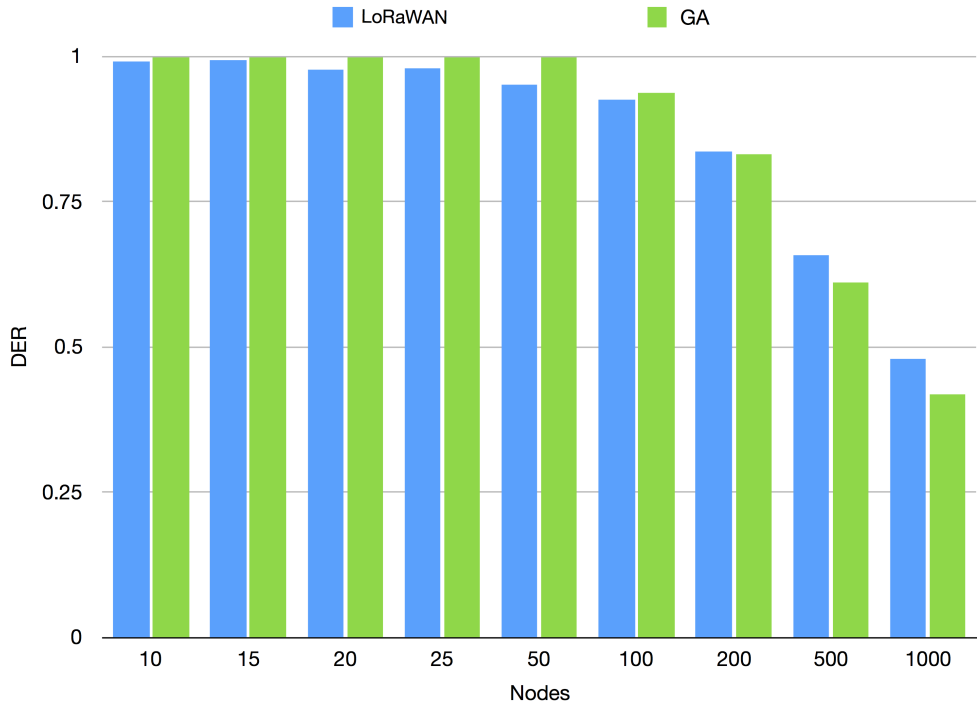
4.1 PRIMEIRA ETAPA

As primeiras execuções do algoritmo genético foram feitas com uma população de 100 indivíduos, um máximo de 100 gerações, e utilizando *crossover* de 1 ponto, sendo este fixo na metade do cromossomo. O resultado pode ser observado no Gráfico 4.1. Para até 50 dispositivos, o algoritmo genético conseguiu gerar uma solução que obtém um *Data Extraction Rate* de 1, ou seja, não houve nenhuma colisão de pacote durante o tempo de simulação da rede. Para 100 dispositivos simulados, a solução gerada pelo algoritmo genético ainda obteve desempenho superior ao do protocolo LoRaWAN, porém para um número maior de dispositivos, o resultado foi inferior.

4.2 SEGUNDA ETAPA

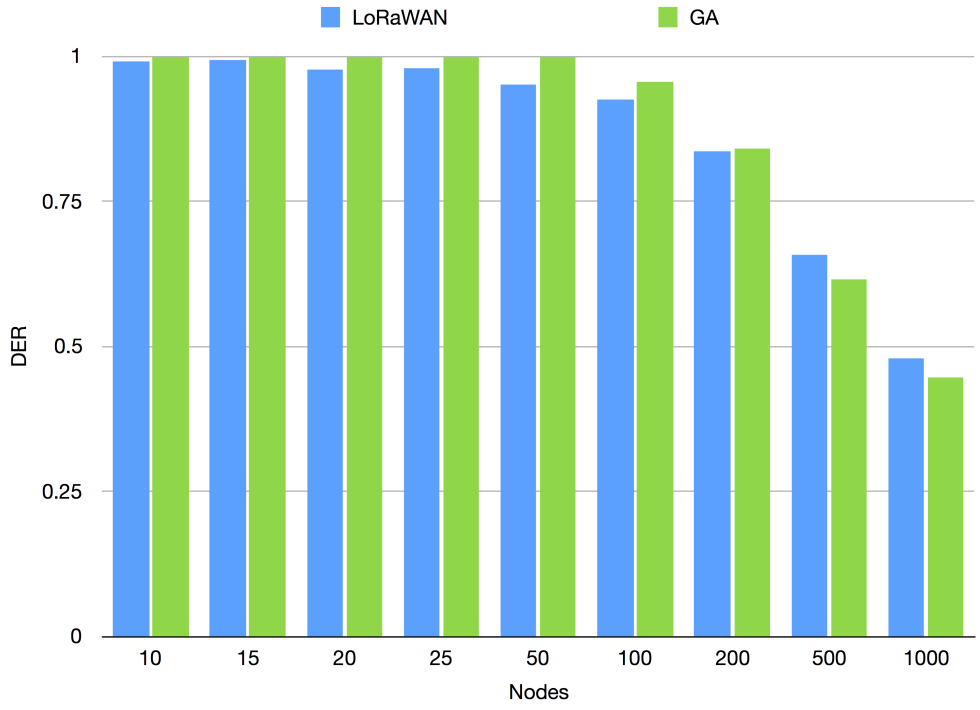
Como o desempenho do algoritmo genético não estava satisfatório, passou-se a utilizar a técnica de *crossover* de 2 pontos, sendo estes fixos. Além disso, a população foi aumentada para 300 indivíduos, o máximo de gerações do algoritmo foi aumentada para 400 e a taxa de mutação foi aumentada minimamente. O resultado é observado no Gráfico 4.2. Estas alterações trouxeram melhorias no desempenho do algoritmo genético, porém estas foram pouco significativas. Ainda, o fato de o protocolo LoRaWAN obter uma solução mais eficiente indica que o algoritmo genético não está explorando de maneira eficiente o espaço de busca, caso contrário o mesmo deveria encontrar soluções com no mínimo eficiência igual ao LoRaWAN.

Gráfico 4.1 – *Data Extraction Rate*: Primeira Etapa



Fonte: Autor.

Gráfico 4.2 – *Data Extraction Rate*: Segunda Etapa

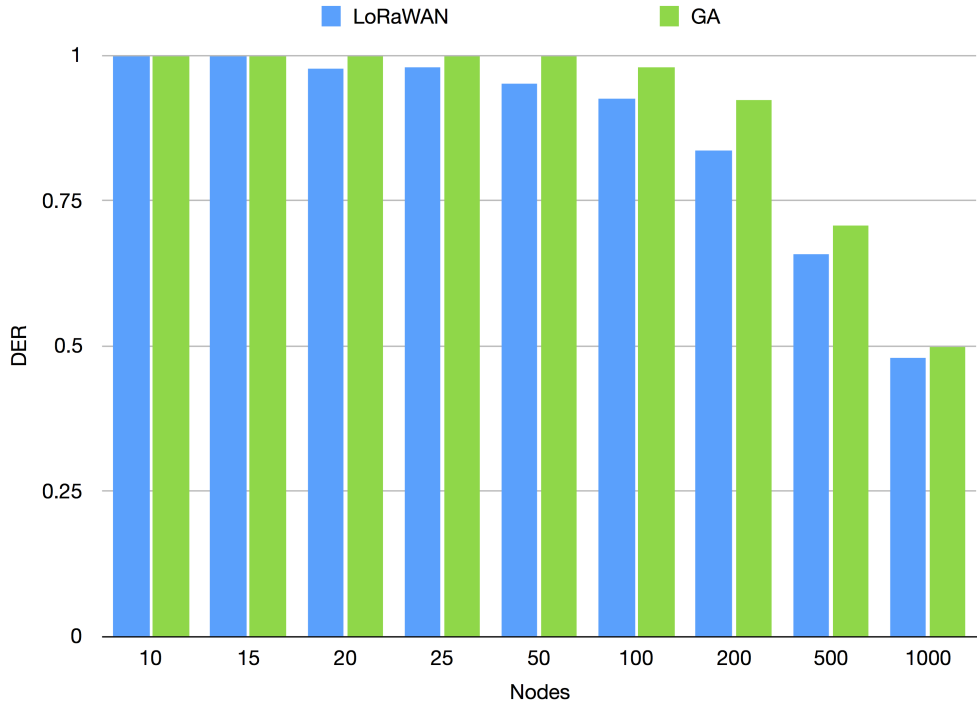


Fonte: Autor.

4.3 TERCEIRA ETAPA

Uma mudança que trouxe benefícios que puderam ser observados rapidamente foi a utilização da técnica de *crossover* de 2 pontos, porém com os pontos sendo variáveis, escolhidos aleatoriamente. O algoritmo utilizado para esta técnica é descrito no Apêndice C. Outra alteração realizada foi o aumento do número de gerações máximo para 600. No Gráfico 4.3, é possível notar um aumento considerável no valor de DER para um número maior de dispositivos. Ainda, com esta implementação do algoritmo genético, foi possível encontrar soluções mais eficientes que o protocolo LoRaWAN para todas as quantidades de dispositivos testadas. Outro ponto interessante de ser percebido é que o desempenho da rede com 100 dispositivos configurados através do algoritmo genético possui desempenho semelhante à rede com 20 dispositivos utilizando o LoRaWAN. O mesmo ocorre em outros pontos do gráfico.

Gráfico 4.3 – *Data Extraction Rate*: Terceira Etapa



Fonte: Autor.

5 CONSIDERAÇÕES FINAIS

Observou-se, através dos resultados encontrados, que para uma rede LoRa com dispositivos estáticos e utilizando uma única estação base, o protocolo LoRaWAN consegue obter um bom desempenho com um baixo número de dispositivos conectados. Porém, conforme a quantidade de nós da rede aumenta, seu desempenho vai diminuindo. Em tais situações, é possível otimizar o funcionamento da rede através da configuração de cada dispositivo utilizando um algoritmo de inteligência artificial.

Apesar da proposta de configurar todos os nós da rede com parâmetros otimizados ser cativante, em uma rede LoRa real tais configurações devem ser definidas em cada dispositivo, o que torna o conceito deste projeto inviável para um grande número de pontos. Um trabalho interessante a ser desenvolvido seria uma modificação do mecanismo de ADR, de forma que o protocolo LoRaWAN possa repassar para os dispositivos configurações geradas no servidor de rede. Deste modo, não seria necessário configurar os dispositivos individualmente, pois o conjunto de configurações poderia ser gerado por um algoritmo externo e repassado ao *gateway* LoRa, que através do protocolo LoRaWAN repassaria para todos os dispositivos. Ainda, uma aplicação no servidor de rede seria capaz monitorar a qualidade de sinal de cada pacote transmitido. Este histórico, juntamente com a contagem de tentativas de transmissão de cada dispositivo, que pode ser transmitida juntamente com o *payload* de cada mensagem, poderia servir como entrada dos algoritmos de otimização, de forma a complementar os resultados obtidos pelas simulações. Assim, um conjunto otimizado de configurações poderia ser gerado periodicamente, levando em consideração dados reais obtidos da rede.

Outro trabalho com implementação relevante seria a obtenção do desempenho obtido pelo algoritmo genético utilizando um simulador que representa uma rede LoRa com protocolo LoRaWAN mais fielmente, como por exemplo o LoRaWANSim, descrito em Pop et al. (2017). Além disso, outra contribuição expressiva seria a realização de uma avaliação mais profunda de cada solução produzida pelo algoritmo genético, como a análise da variância e desvio padrão do DER e consumo de energia de cada dispositivo da rede. Ainda, a implementação e comparação com outros algoritmos de inteligência artificial trariam um complemento importante para este trabalho, como por exemplo otimização por enxame de partículas ou colônia de abelhas.

REFERÊNCIAS BIBLIOGRÁFICAS

AUGUSTIN, A. et al. A study of lora: Long range & low power networks for the internet of things. **Sensors**, v. 16, p. 1466, 2016.

BOR, M. C.; ROEDIG, U. Lora transmission parameter selection. **13th International Conference on Distributed Computing in Sensor Systems (DCOSS)**, p. 27 – 34, 2017. IEEE.

BOR, M. C. et al. Do lora low-power wide-area networks scale? In: **Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems**. New York, NY, USA: ACM, 2016. (MSWiM '16), p. 59–67.

BRANDAO, A. **Genetic Algorithms in PHP Code**. 2015. Acesso em 10 nov. 2017. Disponível em: <http://www.abrandao.com/2015/01/simple-php-genetic-algorithm/>.

CARVALHO, A. P. de. **Algoritmos Genéticos**. Department of Computer Science - USP, 2009. Acesso em 16 nov. 2017. Disponível em: <http://conteudo.icmc.usp.br/pessoas/andre/research/genetic/>.

DARWIN, C. **A Origem das Espécies**. Londres: John Murray, 1859. 502 p.

EGLI, P. R. **Low Power Wide Area Network**: Overview of emerging technologies for low power wide area networks in internet of things and m2m scenarios. 2015. Acesso em 16 nov. 2017. Disponível em: <https://www.slideshare.net/PeterREgli/lpwan>.

EVANS, D. **The Internet of Things**: How the next evolution of the internet is changing everything. Cisco Internet Business Solutions Group, 2011. Acesso em 16 ago. 2017. Disponível em: <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf>.

GABRIEL, P.; DELBEM, A. **Fundamentos de algoritmos evolutivos**. [S.l.]: ICMC-USP, 2008.

GHOSLYA, S. **All About LoRa and LoRaWAN**: Lora: Symbol generation. 2017. Acesso em 15 set. 2017. Disponível em: <http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html>.

GóMEZ, F.; QUESADA, A. **Genetic algorithms for feature selection in Data Analytics**. Neural Designer, 200–? Acesso em 16 nov. 2017. Disponível em: <https://www.neuraldesigner.com/blog/genetic_algorithms_for_feature_selection>.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence**. Cambridge, MA, USA: MIT Press, 1992.

LoRa Alliance. **LoRaWAN What is it?**: A technical overview of lora and lorawan. Technical Marketing Workgroup 1.0, 2015. Acesso em 14 set. 2017. Disponível em: <https://docs.wixstatic.com/ugd/eccc1a_ed71ea1cd969417493c74e4a13c55685.pdf>.

LoRa Alliance Technical Comittee. **LoRaWAN 1.1 Regional Parameters**. Oregon, 2017. A, 56 p. Acesso em 16 nov. 2017. Disponível em: <https://www.lora-alliance.org/lorawan-for-developers>.

MITCHELL, M. **An Introduction to Genetic Algorithms**. Cambridge, MA, USA: MIT Press, 1998.

NORDRUM, A. **Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated**: Telecom/internet. IEEE Spectrum, 2016. Acesso em 16 ago. 2017. Disponível em: <https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated>.

POP, A. I. et al. Does bidirectional traffic do more harm than good in lorawan based lpwa networks? **Telecommunications Research Laboratory**, Toshiba Research Europe Limited, Bristol, UK, 2017.

RAY, B. **What is LoRa?**: The lora iot protocol. Link Labs, 2015. Acesso em 22 set. 2017. Disponível em: <https://www.link-labs.com/blog/what-is-lora>.

Semtech. **What is LoRa?** 2017. Acesso em 14 set. 2017. Disponível em: <http://www.semtech.com/wireless-rf/internet-of-things/what-is-lora/>.

SEMTECH CORPORATION. **AN1200.22**: Lora modulation basics. California, 2015. 26 p. Acesso em 19 out. 2017. Disponível em: <http://www.semtech.com/images/datasheet/an1200.22.pdf>.

TELKAMP, T. **Adaptative Data Rate**. The Things Network, 2017. Acesso em 17 ago. 2017. Disponível em: <https://www.thethingsnetwork.org/wiki/LoRaWAN/ADR>.

VOIGT, T.; BOR, M. **LoRaSim**. 2017. Acesso em 16 ago. 2017. Disponível em: <http://www.lancaster.ac.uk/scc/sites/lora/lorasim.html>.

APÊNDICE A – ARQUIVO CSV: LISTA DE DISPOSITIVOS

```
Id , x , y
0,60.0263579782,151.858095079
1,88.9619299645,50.4153306671
2,105.285388699,203.074777928
3,134.531408603,85.3667015942
4,103.063186142,92.0832913087
5,196.422801171,93.5733262465
6,140.818682291,96.1058772809
7,110.439616473,103.026357327
8,182.23387255,137.32939536
9,72.0026155182,183.232091973
```

Fonte: Autor.

APÊNDICE B – MUTAÇÃO DE UM CROMOSSOMO

```
if random.random() < 0.1:
    mutation_len = round(length*random.uniform(0, 0.3))

    for i in range(mutation_len):
        index = random.randrange(0, length)
        new_chromossome[index] ^= 1
```

Fonte: Autor.

APÊNDICE C – *CROSSOVER* DE 2 PONTOS VARIÁVEIS

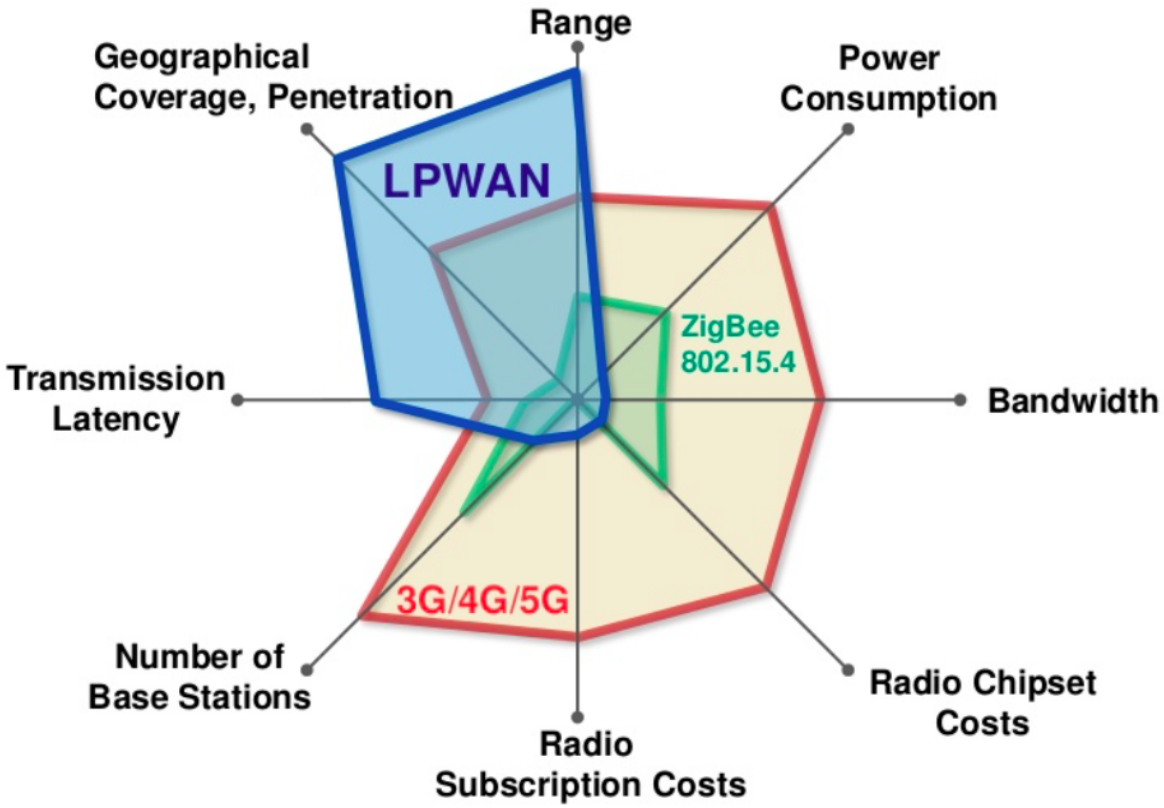
```
length = nodes*self.length

crossover_point_1 = random.randrange(0, length)
crossover_point_2 = random.randrange(crossover_point_1, length)

new_chromossome = self.chromosome[:crossover_point_1]
new_chromossome.extend(other.chromosome[crossover_point_1:crossover_point_2])
new_chromossome.extend(self.chromosome[crossover_point_2:])
```

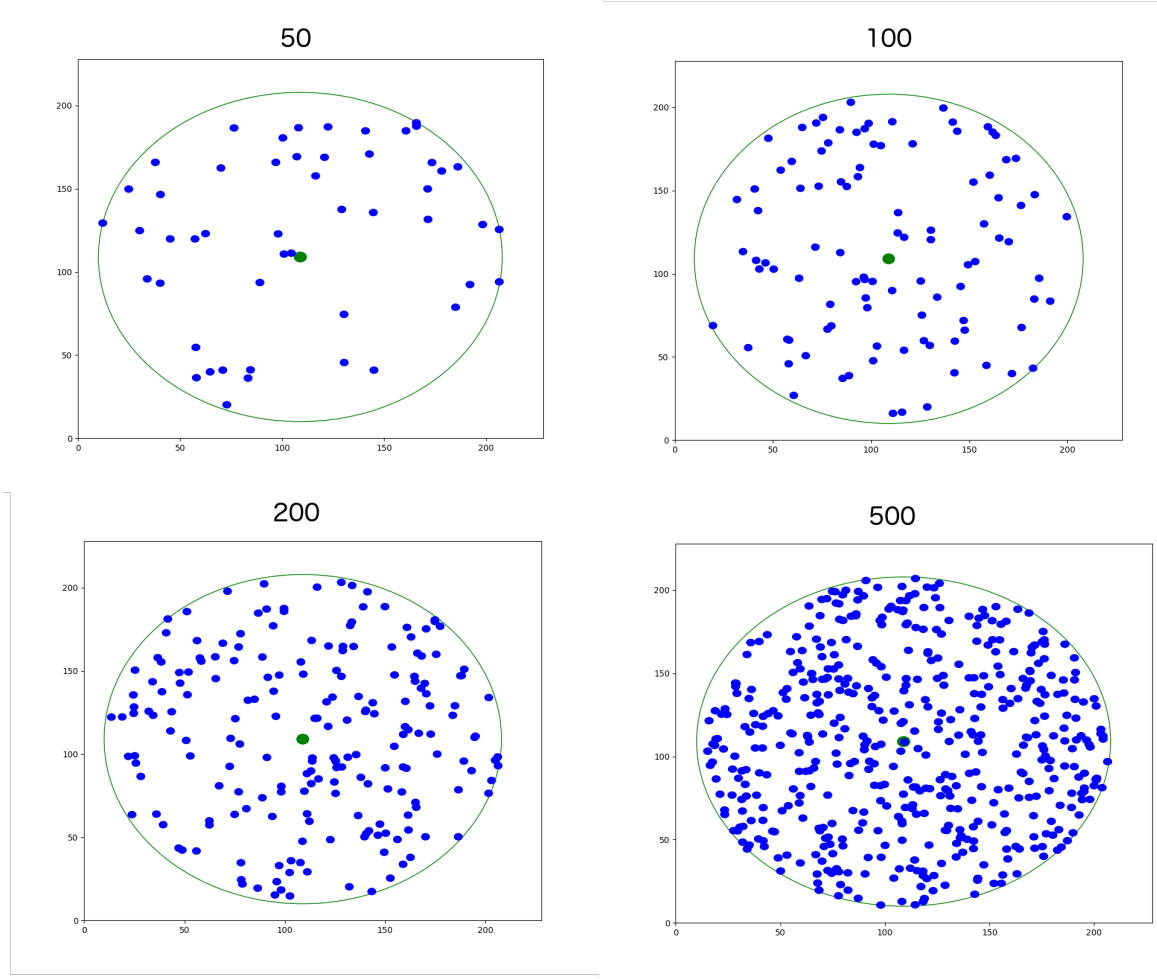
Fonte: Autor.

ANEXO A – COMPARAÇÃO DE LPWAN COM ZIGBEE E GSM



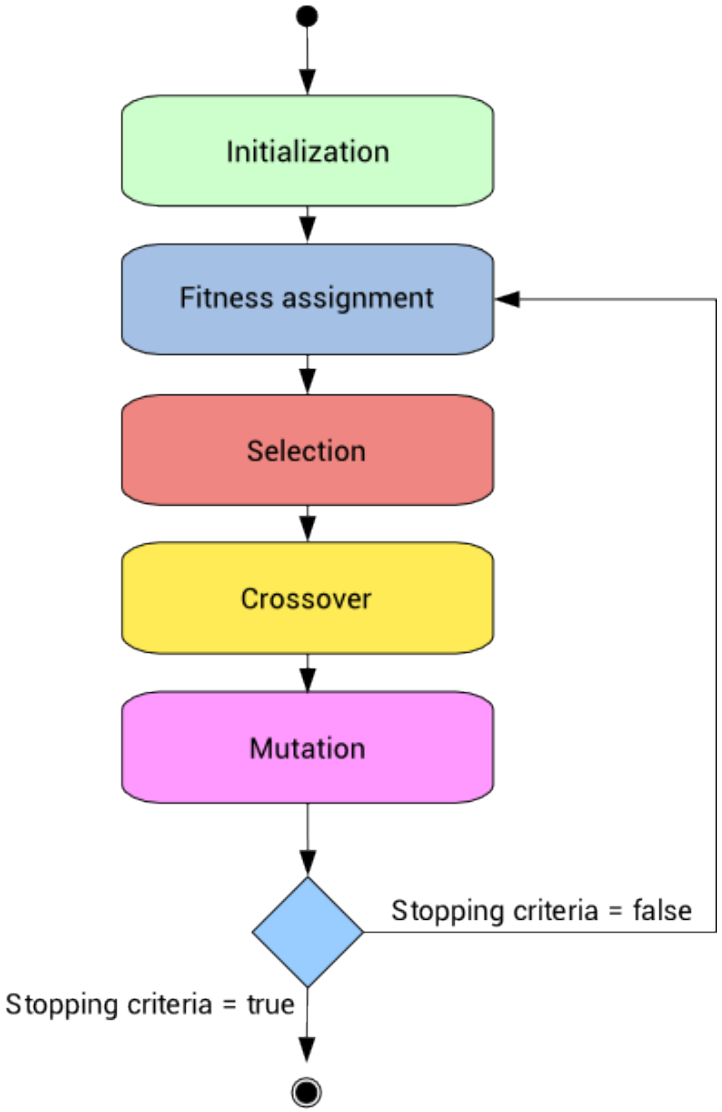
Fonte: Adaptado de Egli (2015).

ANEXO B – DISPOSITIVOS POSICIONADOS PELO LORASIM



Fonte: Adaptado de Voigt e Bor (2017).

ANEXO C – FLUXOGRAMA DE UM ALGORITMO GENÉTICO



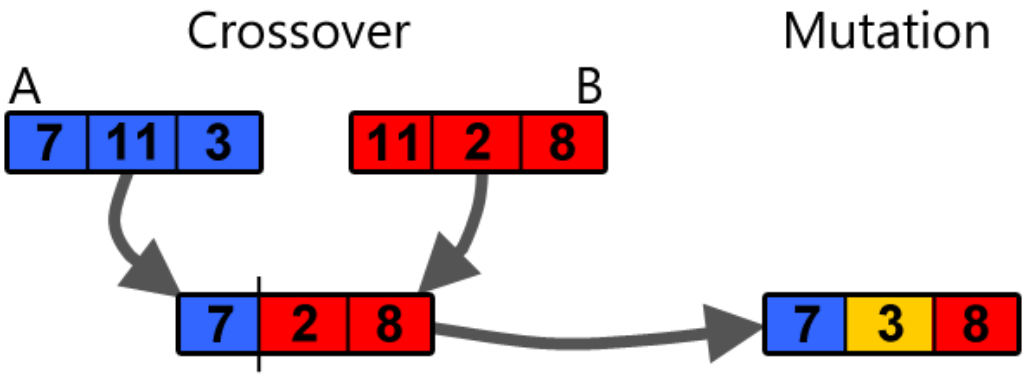
Fonte: Gómez e Quesada (200-?).

ANEXO D – VERIFICAÇÃO DE COLISÃO POR FREQUÊNCIA

```
# frequencyCollision, conditions
#
#         |f1-f2| <= 120 kHz if f1 or f2 has bw 500
#         |f1-f2| <= 60 kHz if f1 or f2 has bw 250
#         |f1-f2| <= 30 kHz if f1 or f2 has bw 125
def frequencyCollision(p1,p2):
    if (abs(p1.freq-p2.freq)<=120 and (p1.bw==500 or p2.bw==500)):
        print ("frequency collision 500")
        return True
    elif (abs(p1.freq-p2.freq)<=60 and (p1.bw==250 or p2.bw==250)):
        print ("frequency collision 250")
        return True
    # bw == 125
    else:
        if (abs(p1.freq-p2.freq)<=30):
            print ("frequency collision 125")
            return True
        else:
    print ("no frequency coll")
    return False
```

Fonte: Adaptado de Voigt e Bor (2017).

ANEXO E – *CROSSOVER* E MUTAÇÃO EM UM CROMOSSOMO



Fonte: BRANDAO, A. (2015).