Proposta de Controle de Espalhamento Espectral Utilizando Aprendizado por Reforço Profundo para Otimização do Desempenho de Redes LoRa/LoRaWAN

Carlos D. Bezerra[1], **Antonio Oliveira-Jr**[1,2], **Flávio H. T. Vieira**[1]

[1]Instituto de Informática (INF) – Universidade Federal de Goiás (UFG)

[2]Fraunhofer Portugal AICOS

carlosbezerra@inf.ufg.br, antoniojr@ufg.br, flavio_vieira@ufg.br

Abstract. *The number of connected Internet of Things (IoT) devices is growing more and more and will tend to increase in the coming years mainly with the arrival of 5G networks. This will lead to a large amount of data traffic on the communication network. The purpose of this article is to propose an intelligent method based on Deep Q Networks networks (DQN), where the agent is trained to learn a policy of parameters involving involving the LoRa modulation, so that a multi-serial connection is optimized, improving the transmission rate and minimizing the loss rate of transmitted packets. The development methodology is through computer simulations. The results point to a promising optimization and control technique.*

Resumo. *O número de dispositivos de Internet das Coisas (IoT) conectados cresce cada vez mais e tende a aumentar nos próximos anos, principalmente com a chegada das redes 5G. Isso resultará em um intenso tráfego de dados no sistema de comunicação, podendo prejudicar a qualidade de transmissão devido aos congestionamentos e perdas de pacote por colisão. O objetivo desse artigo é propor um método inteligente baseado em redes Deep Q Networks (DQN), onde o agente é treinado para aprender uma política de ações envolvendo parâmetros de modulação do protocolo LoRa, de forma que a conexão multiusuário seja otimizada. A metodologia de desenvolvimento desse artigo é por meio de simulações computacionais. Os resultados apontam para uma técnica de otimização e controle promissora.*

1. Introdução

A Internet das Coisas (*Internet of Things - IoT*) é realidade na sociedade moderna e refere-se a conexão digitalizada de sensores e objetos do nosso cotidiano. Esta rede de sensores é capaz de reunir e transmitir dados com a Internet, permitindo o seu monitoramento remoto. Veículos autônomos, cidades inteligentes e processos industriais são exemplos da aplicabilidade de tecnologias embarcadas que transmitem dados e podem ser controlados ou manipulados [Al-Fuqaha et al. 2015]. Hoje a Internet deixa de ser somente uma rede de computadores e passa a ser uma rede de pessoas, comunidades e dispositivos.

Tem-se como requisito básico da IoT a eficiência energética. Uma rede de sensores e dispositivos IoT é movida e operada normalmente por baterias com baixa

potência disponível. Dessa forma, a vida útil dos equipamentos é uma prioridade em seu processo de fabricação e uso, levando a restrições que limitam a aplicabilidade de determinados protocolos de redes de comunicações.

Grande parte dos dispositivos IoT utiliza um único transceptor *half-duplex* pela necessidade de eficiência energética dos módulos, impossibilitando o uso de múltiplos canais simultaneamente durante o processo de transmissão de dados [Hasegawa et al. 2020]. A conexão massiva multiusuário, bem como a coexistência de redes sem fio de diferentes protocolos, levam a problemas relacionados a conexão e perdas de informação durante a transmissão. A tendência da coexistência e conexão massiva de dispositivos é aumentar cada vez mais, pois com a chegada do sistema 5G, uma tecnologia habilitadora, surgem problemas relacionados ao tráfego de dados na rede de acesso.

O protocolo LoRa (*Long Range*) é um dos mais utilizados em IoT, principalmente em redes de área ampla (*Long Power Wide Area Networks - LPWAN*) [Park et al. 2020]. Este protocolo inclui duas camadas do modelo OSI/ISO: física (LoRa RF) e camada de enlace (LoRaWAN). Sua principal característica é a capacidade de realizar comunicações de longa distância e baixa potência, atendendo requisitos para IoT que redes sem fio convencionais não costumam atender. Uma rede LoRa possui parâmetros de comunicação adaptáveis para manter boas conexões de *uplink* e *downlink* [Raza et al. 2017].

Uma comunicação eficiente entre dispositivos LoRa ocorre quando se obtém parâmetros apropriados para a potência mínima de transmissão com satisfatória taxa de transferência de dados, apresentando também baixa perda por colisões de pacotes. Entretanto, como já citado, este é um desafio em sistemas multiusuário [Tesfay et al. 2020].

O aprendizado por reforço (*Reinforcement Learning - RL*) é uma técnica de aprendizado de máquina e portanto um subcampo da inteligência artificial. Problemas de RL são matematicamente modelados como cadeias de *Markov* e compostos basicamente por: um agente que realiza ações em ambiente, a recompensa obtida e os estados do agente. O ambiente pode ser real ou simulado e produz informações que descrevem os estados do sistema. Utilizando estas informações de estados, o agente realiza uma determinada ação resultando em uma recompensa ou penalização. O objetivo da técnica é maximizar as recompensas obtidas, de forma que o agente aprenda por experiências e execute ações ótimas, denominadas como políticas. A tecnologia de RL é usada atualmente para resolver uma variedade de problemas. No campo de redes, ela pode ser utilizada para controle e alocação de recursos e balanceamento de carga [Park et al. 2020].

Atualmente o protocolo LoRa especifica o algoritmo Taxa de Dados Adaptativo (*Adaptative Data Rate - ADR*). Este algoritmo é capaz de otimizar parâmetros da camada física, como a taxa de dados e o "Tempo no Ar" (*Time on Air - ToA)* e tem por objetivo minimizar os efeitos causados pelo aumento de usuários na rede. Basicamente o ADR usa a relação sinal ruído *(Signal to Noise Ratio - SNR)* para determinar bons parâmetros LoRa RF. Portanto é uma técnica iterativa, baseada em leituras de estados coletados e controle em tempo real.

Segundo [Park et al. 2020] as pesquisas atuais visam modificar o ADR existente, para melhorar o seu rendimento ou adicionar tecnologias, como aprendizado de máquina para uso confiável dos recursos de rede. Assim, formula-se o seguinte problema de

pesquisa: se é possível aplicar em ambiente de simulação, algoritmos de aprendizado por reforço para maximizar o desempenho de sistemas multiusuário LoRa.

Neste contexto, o artigo apresenta uma proposta de controle de espalhamento espectral, um parâmetro de modulação LoRa, utilizando técnicas inteligentes, sem o uso do ADR tradicional. Levantamentos bibliográficos realizados indicam que a literatura voltada para este assunto ainda é incipiente em métodos baseados em aprendizagem por reforço. Assim, este trabalho contribui no atual estado da arte nos seguintes pontos:

- novos algoritmos visando a melhoria de sistemas multiusuário;
- técnicas alternativas ao ADR existente;
- sistema de controle robusto/inteligente de parâmetros da camada física LoRa RF.

Este artigo está organizando da seguinte forma. Para além dessa Introdução, a Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve detalhadamente a proposta do artigo, enquanto que a Seção 4 apresenta a validação da proposta em um ambiente de simulação. A Seção 5 discute os resultados da avaliação de desempenho, e por fim a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Esta seção apresenta os principais trabalhos relacionados aos pontos de melhorias na transmissão multiusuário e eficiência energética em redes LoRa, essencialmente trabalhos que envolvem estudos sobre as camadas físicas e de enlace de dados do protocolo.

[Augustin et al. 2016] desenvolvem estudo sobre o protocolo LoRa, apontando soluções para melhoria de desempenho. Os autores explicam que de acordo com a especificação atual, os dispositivos e *gateways* podem transmitir em qualquer tempo. Não existe um mecanismo para ”ouvir depois de falar” como o protocolo CSMA (*Carrier Sense Multiple Access*). Os resultados de suas pesquisas mostram que o LoRa é extremamente sensível ao carregamento do canal e as soluções implementadas em protocolos de rede, como *IEEE 802.11* podem ajudar a mitigar este problema. A fim de garantir escalabilidade, torna-se interessante estudar a viabilidade de implementação de um mecanismo CSMA ou semelhante. Ainda sobre o trabalho de [Augustin et al. 2016], os autores afirmam que os parâmetros de modulação do protocolo influenciam na efetiva taxa de dados, resistência à interferência do ruído e decodificação de pacotes recebidos.

Atualmente o protocolo LoRa especifica o ADR, algoritmo capaz de otimizar parâmetros da modulação digital empregada, adequando o tempo no ar (ToA). Melhorias ou alternativas ao ADR podem ser propostas com o uso de técnicas de inteligência artificial. Pesquisadores como [Park et al. 2020] propuseram técnicas inteligentes para compará-las ao ADR e conseguiram melhoria de aproximadamente 15% em relação a transmissão de dados e eficiência energética.

O pesquisador [Weyn 2016] da Universidade da Antuérpia, bem como [De Poorter et al. 2017] de seu grupo de pesquisa, investigam o desempenho de redes LPWA para IoT. Desta forma [Weyn 2016] propôs o simulador LoRaWAN e *Sigfox* para estudo e observação das colisões de pacotes. O simulador encontra-se reconhecido em demais trabalhos, como o de [Marais et al. 2019]. A partir dos resultados de simulação observa-se sensibilidade do protocolo LoRa em relação a conexão massiva de dispositivos. Evidentemente é necessário o controle constante de parâmetros da camada

física e de enlace de dados. Os resultados mostram, ainda, que a alteração dos parâmetros da modulação utilizados no protocolo modificam a eficiência da transmissão, entretanto, não foi proposta nenhuma solução para a seleção otimizada deles.

Os autores [Hasegawa et al. 2020] implementaram técnica de aprendizado por reforço para otimizar a conexão multiusários em dispositivos IoT coexistentes. A proposta de aprendizagem por reforço é baseada no paradigma *Multi-Armed Bandit - (MAB)*. A metodologia de implementação é por experimentação prática, aplicando o método em redes IoT coexistentes (LoRa, *SigFox* e *Wi-Fi Sun*). Os resultados revelam eficiência da técnica proposta, reduzindo perda de pacotes por colisão. Observa-se que os autores utilizam a técnica CSMA em conjunto com o sistema inteligente, entretanto essa implementação não é detalhada no artigo. Sabe-se que o CSMA não é um protocolo nativo do LoRa. Um ponto de melhoria identificado no artigo é a possibilidade de utilização de técnicas modernas em comparação ao MAB, como redes neurais profundas (*Deep Q Networks - DQN*). Ainda em relação ao trabalho, o método experimental proposto não leva em consideração parâmetros da modulação LoRa, que quando são controlados de maneira ótima, podem melhorar a eficiência do processo comunicação.

3. Proposta de Controle de Espalhamento Espectral Inteligente

A rede LoRa utiliza a modulação digital denominada *Chirp Spread Spectrum - (CSS)*, um tipo de tecnologia proprietária desenvolvida pela Semtech [Semtech 2019]. Nesta tecnologia, o sinal digital é modulado por meio de pulsos de *chirp* lineares (*Compressed High Intensity Radar Pulse*) que possuem amplitude constante e varrem toda a largura de banda. A eficiência em termos de taxa de dados e potência desta modulação depende principalmente de três parâmetros: i) largura de banda *(Bandwidth - BW)*, fator de espalhamento espectral (*Spreading Factor - SF)*, taxa de codificação *(Code Rate - CR*. Um sinal digital modulado pela tecnologia CSS é composto pelo preâmbulo e carga útil de informação (*payload*). Os pulsos *chirp* (uma espécie de rampa) são portadores de dados espaçados no domínio da frequência.

O SF representa a forma em que um *bit* (ou símbolo) é espaçado ou modulado, podendo assumir seis valores distintos. A taxa de dados (*bit rate*) R_b *bps* do LoRa é dada por:

$$R_b = SF \times \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} \tag{1}$$

Um sistema de aprendizado por reforço visa fornecer conhecimento a um determinado agente por meio interações em um ambiente. As qualidades das ações deste agente são medidas por meio da função de recompensa. A função de valor estado-ação $Q(s, a)$ quantifica as recompensas obtidas por este agente em relação aos seus estados e ações, tanto em curto quanto em longo prazo. Desta forma, o agente aprende políticas de ações π que maximizam a função de valor. A função $Q(s, a)$ é uma variação do algoritmo de aprendizagem temporal proposto por Bellman e é dada por:

$$Q^{\pi}(s, a) = r + \gamma \ \max Q^{\pi}(s', a') \tag{2}$$

onde r é a recompensa imediata obtida e γ um fator de desconto para recompensas futuras, s, s', a e a' os estados e ações presentes e futuros respectivamente.

O algoritmo DQN, um tipo de RL, incorpora em seu modelo os parâmetros de uma rede neural, com pesos sinápticos (θ) e *bias*. Normalmente a rede utilizada é profunda (mais de três camadas) e tem como objetivo estimar a função $Q(s, a)$ ótima, como representando na Equação 3. O DQN não necessita de um sistema tabular para mapear os estados e ações do agente, como o tradicional Q-Learning. Em um ambiente com muitos estados e ações, por exemplo uma rede LoRaWAN, sistemas tabulares são inviáveis. O mapeamento estado-ação do DQN é fornecido pela própria rede neural. Além disso, diferente do aprendizado supervisionado, no DQN os dados são dinâmicos e atualizados a cada passo de iteração do algoritmo, o que aumenta seu poder de generalização.

$$Q(s, a) \approx \hat{Q}(s, a, \theta) \tag{3}$$

Contextualizando a rede LoRa com o aprendizado por reforço, pode-se representar o ambiente como um simulador de rede sem fio de longo alcance. O agente é incorporado ao *gateway* e experimenta ações na camada física (modulação) para se comunicar com dispositivos finais (*end-nodes*). Os estados representam variáveis mensuráveis e observáveis neste processo, resultantes da experimentação das ações do agente no ambiente. Se um dispositivo final, por exemplo um sensor, está muito distante do *gateway* LoRa, torna-se necessário aumentar o fator SF na modulação CSS, mantendo a informação transmitida por mais tempo no ar (*ToA*). Entretanto este aumento faz com que a taxa de transferência de dados decaia e a potência de transmissão aumente, comprometendo a eficiência energética dos módulos.

Quando a transmissão de um pacote de dados ocorre de maneira simultânea, no mesmo canal, existe a probabilidade destes pacotes se colidirem, corrompendo a informação. Após uma colisão, normalmente é necessário retransmitir um pacote, diminuindo assim a capacidade do canal de comunicação e a eficiência da rede [Xu et al. 2020]. Para evitar as colisões, a rede LoRa adota o protocolo de acesso randômico ALOHA. A escolha do SF altera o tempo médio para transmissão do (T_{rf}), que é diretamente proporcional ao *ToA* daquela determinada SF. Ou seja, a escolha do SF está relacionado ao desempenho do protocolo ALOHA na contenção das colisões entre pacotes.

Diante destas possibilidades, observa-se a expectativa dos parâmetros SF serem otimizados durante um processo de comunicação entre dispositivos LoRa. A Figura 1 ilustra a relação entre o *ToA* e a taxa de dados para diferentes fatores de espalhamento espectral. Quanto menor o *ToA* na transmissão de dados entre um dispositivo final e o *gateway*, menor é a chance da ocorrência de colisões de pacotes. Todavia, quanto maior a distância entre o *gateway* e o dispositivo final, necessita-se disponibilizar a informação por mais tempo no espaço, aumentando assim o *ToA* e consequentemente o SF.

O problema de pesquisa a ser abordado neste trabalho é: como um sistema inteligente pode ser treinado para selecionar de forma automatizada e ótima, parâmetros de modulação na camada física do protocolo LoRa. Assim, este artigo propõe um agente inteligente baseado em aprendizagem por reforço para atuar na seleção de parâmetros da camada física (modulação CSS) de forma a maximizar a eficiência de uma rede multiusuário LoRa/LoRaWAN.

Para o desenvolvimento do agente inteligente capaz de realizar de forma otimizada a seleção dos parâmetros de modulação LoRa, é necessário considerar três pontos

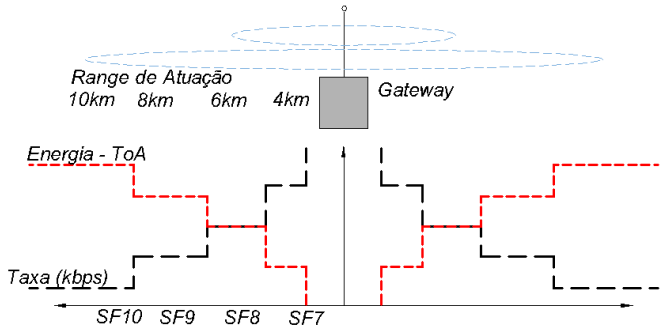


Figura 1. Impactos funcionais nas alterações do SF - adaptado de [Semtech 2019].

essenciais: **i) colisões de pacotes, ii) atenuação do sinal e iii) taxa de transferência.** O efeito da distância entre módulos e *gateway*, consequentemente as perdas de propagação no espaço livre e atenuação do sinal, não são considerados no modelo proposto por [Weyn 2016].

O agente proposto considera o cenário apresentado na Figura 2, onde tem-se um *gateway* conectado a *n*-dispositivos que representam sensores IoT, ou receptores LoRa denominados (*end-nodes*). Estes *end-nodes* são distribuídos sob a mesma distância, representando uma simulação homogênea. A distância é alterada durante a simulação, mas a homogeneidade, ou seja, a distância de todos os dispositivos ao *gateway* é mesma. Os dispositivos são dispostos igualmente no espaço.

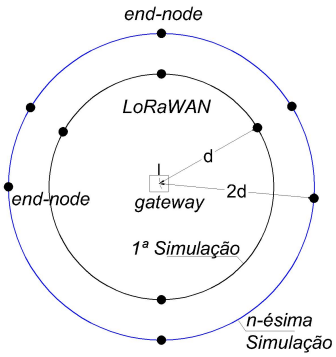


Figura 2. Cenário considerado para desenvolvimento do agente inteligente.

Nesse artigo os parâmetros do *frame* que influenciam no cálculo do *ToA*, tais como o *Code Rate* (*CR*) e o comprimento do preâmbulo são fixos e não variam durante a simulação. A Tabela 1 abaixo apresenta os valores de *ToA* estimados com o auxílio da calculadora LoraTools (`https://www.loratools.nl/#/airtime`) para transmissão de um pacote de 25 bytes, tamanho este padronizado para as análises deste artigo.

Tabela 1. *ToA* estimado para cada SF.

ToA(ms)	1646	921	460	230	127	70
SF	12	11	10	9	8	7

A Figura 3 dispõe o diagrama de blocos que contextualiza o agente inteligente que atua no processo de aprendizagem por reforço para otimização da performance da rede LoRa, contendo as **ações**, os **estados observáveis** e a **recompensa** imediata obtida.

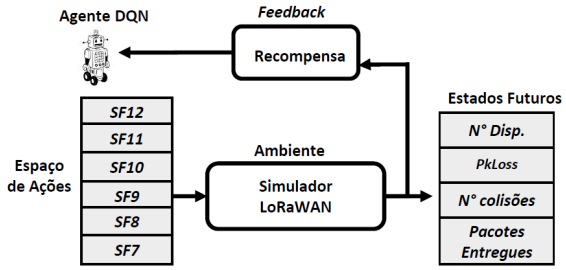


Figura 3. Ações, Estados e Recompensas.

O espaço de estados do sistema de aprendizado por reforço é definido por meio de quatro variáveis observáveis no processo: i) número de dispositivos finais conectados ao *gateway*, ii) perdas de propagação no espaço livre, iii) n°. de colisões e iv) pacotes entregues. O agente deve aprender a definir uma boa SF para cada situação, realizando seis ações possíveis: i) SF12, ii) SF11, iii) SF10, iv) SF9, v) SF8 e vi) SF7.

Para avaliar os efeitos das distâncias entre o *end-node* e o *gateway*, considera-se a razão entre as perdas de propagação no espaço livre (*Packet Loss*) Pk_{Loss} e o *ToA* (ms) de cada **SF**. O Pk_{Loss} é usado para estimar a atenuação (dB) do sinal em função da distância. Esta relação compõe parte do cálculo da recompensa imediata obtida, de forma que quanto maior o Pk_{Loss}, consequentemente a distância, menor é o valor da recompensa. Como o *ToA* compensa altos valores de Pk_{Loss}, o agente tende a selecionar uma SF com maior *ToA*.

A recompensa imediata (R) elaborada para desempenhar *feedback* de cada ação no simulador é dada pela função:

$$R(P, Pk_{Loss}) = \sum_{i=0}^{N} P \times \left(\frac{Pk_{Loss}}{ToA} \right)^{-1} \tag{4}$$

onde P são os pacotes entregues com sucesso com N dispositivos, Pk_{Loss} (dB) as perdas de propagação no espaço livre e ToA o tempo de propagação do sinal (*time on air*) respectivamente. Ao longo do processo de aprendizagem esta função deve ser maximizada.

O cálculo de Pk_{Loss} (dB) é dado por:

$$Pk_{Loss} = 32,45 + 20log(D) + 20log(f) \tag{5}$$

onde D(km) é a distância entre um nó e um *gateway*, f(MHz) é a frequência de modulação.

A Figura 4 apresenta o modelo da rede neural proposta para o desenvolvimento do algoritmo DQN. O diagrama de blocos é gerado pela biblioteca Keras do *Python*, onde a rede neural é composta por cinco camadas ocultas *fully connected* (densas). As três camadas ocultas possuem, respetivamente 32, 32, 24 e neurônios, função de ativação

ReLU. A camada de saída possui 6 neurônios, e função de ativação linear, pois o problema de regressão da função Q necessita de uma saída $[0, \infty]$. O número de neurônios da camada de saída deve coincidir com o tamanho do espaço de ações, onde a função Q indexa o valor da ação a ser tomada. A métrica de avaliação (*loss function*) é o Erro Médio Quadrático (*Mean Square Error* - MSE) entre o Q_{alvo} calculado (vide Equação 1) e o predito pela rede neural $\hat{Q}$.

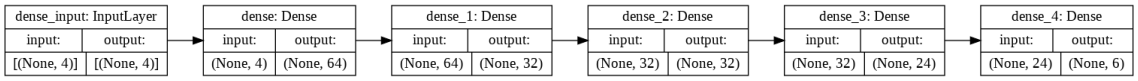


Figura 4. Configuração da rede DQN.

As entradas da rede neural são normalizadas evitando problemas denominados explosão de gradientes, que são falhas no processo de aprendizagem. A normalização melhora a estabilidade da rede neural. Por isso todos os estados são normalizados entre valores de $0 - 1$. A normalização dos estados é dada por:

$$y = \frac{(x - min)}{(max - min)} \tag{6}$$

onde min representa o valor mínimo de um determinado estado e max o valor máximo. O valor atual de um estado é dado por x e y o estado normalizado.

O procedimento de aprendizagem das redes DQN ocorre com o uso da memória de aprendizagem (*Memory Replay*). Este procedimento consiste em adquirir amostras aleatórias do tamanho do *mini-batch*, ou seja, tamanho do lote de dados utilizados para treinar cada época da rede neural. Estas amostras correspondem as experiências obtidas pelo agente durante o processo exploratório. A memória de aprendizagem é computacionalmente representada por um *deque* (ou fila) contendo os estados, ações e recompensas que são atualizados a cada nova experiência. Amostrar um pequeno subconjunto de memórias no largo conjunto de experiências torna o processo de aprendizagem mais eficiente. Por esta razão o *deque* utilizado é normalmente grande.

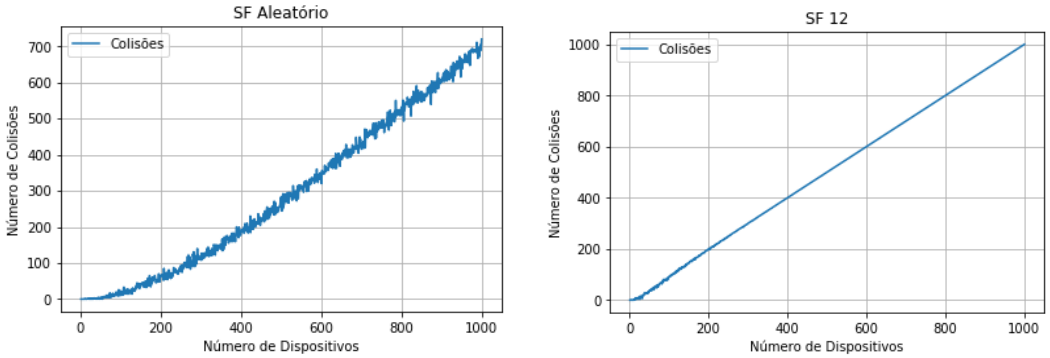
Com a proposta e sequência de desenvolvimento do algoritmo definidas, parte-se para a implementação e validação. A próxima seção descreve a metodologia experimental da proposta por meio de simulações computacionais. Desta forma, desenvolve-se a rede de comunicação LoRa. Logo a seguir parametriza-se o algoritmo inteligente para ser implementado no *gateway*.

4. Validação e Ambiente de Simulação

São realizadas simulações multiusuário, envolvendo um *gateway* e múltiplos nós (usuários). O simulador é implementado em *Python* com base no trabalho e experimentos desenvolvidos por [Weyn 2016]. Após a implementação do simulador e observação dos resultados oriundos desse processo, realiza-se o desenvolvimento do algoritmo DQN para maximizar o desempenho da rede LoRa.

A Figura 5(a) ilustra o resultado de implementação do simulador LoRaWAN proposto por [Weyn 2016] com as adaptações necessárias. Avalia-se na simulação a obtenção dos resultados de colisões entre pacotes com utilização de até 1000 (mil)

dispositivos que transmitem ao mesmo tempo, contidos pelo protocolo ALOHA. A seleção dos parâmetros de modulação SF foi inicialmente aleatória, sem controle específico, onde cada mensagem possui o tamanho de 25 bytes.



(a) Simulação SF Aleatório. (b) Simulação SF 12.

Figura 5. Validação do Simulador LoraWAN em Python.

A Figura 5(b) apresenta o resultado da simulação com seleção do SF fixo (SF12). É possível observar o aumento do número de colisões de pacotes transmitidos com o aumento respectivo do número de dispositivos finais. Na simulação com SF aleatório, em 1000 dispositivos obteve-se $\approx$ 70% de colisões. Já fixando SF em 12 e 1000 dispositivos, nenhum pacote foi entregue. Observa-se, portanto, que a escolha do SF impacta diretamente nas colisões de pacotes.

Como já explicado na seção 3, o simulador proposto por [Weyn 2016] não é suficiente para promover a seleção otimizada do parâmetro SF. A distância entre o dispositivo final e *gateway* também é um critério importante, por esta razão é adicionado no simulador o procedimento de cálculo do *ToA* e $Pkloss$. Assim o futuro agente inteligente a ser treinado pode selecionar uma ação levando em consideração tanto as colisões de pacotes quantos as distâncias entre os *end-nodes* e *gateway* (relação $\frac{ToA}{Pk_{loss}}$) em seu sistema de recompensas.

Em relação as configurações do algoritmo DQN, o tamanho do *deque* de memória é um hiperparâmetro do algoritmo de aprendizagem por reforço. Como exemplo, ao utilizar um *deque* de memória com 1000 elementos têm-se a memória mais larga do que 500 elementos (memória mais curta). Em observações empíricas do experimento, verificou-se que o algoritmo utilizado estabiliza melhor com memórias mais largas e *mini batch* menor. Desta forma mantém-se o tamanho do *memory* em 2000 elementos. O *mini-batch* possui tamanho 16. Por meio da Figura 6 ilustra-se o diagrama de blocos do processo de treinamento da rede DQN, contendo os componentes de *memory replay*, *mini-batch*, entradas e saídas da rede.

Visando o treinamento do agente, desenvolveu-se dois testes no simulador, onde os parâmetros foram configurados da seguinte forma: i) distância e número de dispositivos fixos e ii) distância e número de dispositivos variáveis. O objetivo do teste com os parâmetros fixos foi verificar a convergência do algoritmo, ou seja, se ocorre a maximização das recompensas, visando menos oscilações possíveis. Após a confirmação da convergência do algoritmo, realizou-se os testes com parâmetros variáveis, visando aumentar a gama de experiências do agente, generalizando outras situações. No teste com

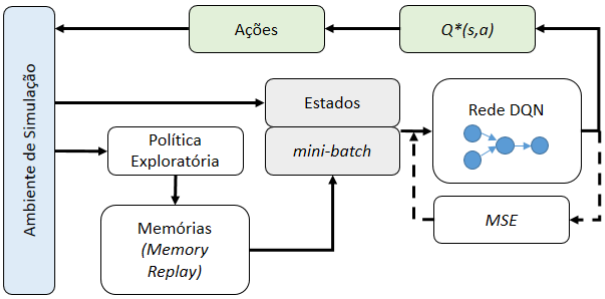


Figura 6. Representação do algoritmo DQN.

parâmetros variáveis foi criada uma lista com distâncias entre $0,5km$ e $7km$ espaçadas de maneira uniforme. Desta forma, em cada passo de simulação a distância entre *end nodes* e *gateway* foram indexados a uma determinada distância. Os números de *end node* também são variados da mesma forma, em uma faixa de 300 a 800 dispositivos.

A seleção dos hiperparâmetros da rede neural foi obtida por experimentação, observando sempre a estabilização do processo e a maximização das recompensas. Os melhores parâmetros para treinamento do modelo estão resumidos na Tabela 2, contendo a configuração da rede, o número de episódios de treinamento, o tamanho do *mini-batch* e memória de experiência, a taxa de aprendizagem e o decaimento do parâmetro de exploração (*exploration*) ϵ - *greedy*.

Tabela 2. Parâmetros da rede DQN.

N° Camadas Ocultas	Episódios de Treinamento	*mini-batch*	Memória de experiência (elementos)	Taxa de Aprend. (α)	ϵ (Decaimento)
5 (3 ocultas)	1000	16	2000	0,0001	0,998
5 (3 ocultas)	500	16	2000	0,0001	0,998

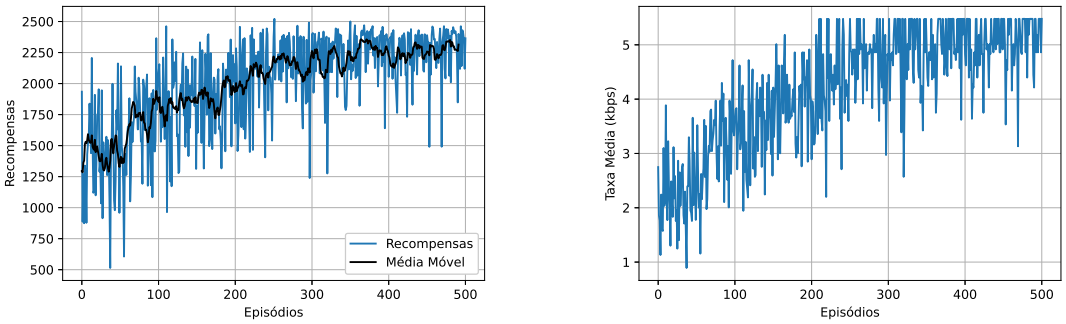
5. Avaliação de Desempenho e Resultados

O método de avaliação do desempenho da rede LoRaWAN consiste em utilizar métricas usuais de redes de computadores, tais como o número de colisões de pacotes, o PER (*Packet Error Rate*) e a vazão geral do sistema. Estas métricas são aplicadas no agente treinado por parâmetros fixos e variáveis, bem como no agente aleatório, que seleciona suas ações de forma randômica. A partir destas métricas é possível constatar se houve melhoria no desempenho da rede LoRa inteligente. Esta seção está organizada da seguinte forma: inicialmente são apresentados os resultados de treinamento da rede DQN por parâmetros fixos e variáveis. Logo em seguida compara-se através das métricas supracitadas, o agente inteligente com um agente randômico.

5.1. Treinamento com parâmetros fixos

Os testes com parâmetros fixos objetivam, inicialmente, validar o algoritmo DQN. A Figura 7(a) apresenta o resultado do treinamento com 500 dispositivos e distância entre *end node* e *gateway* de 200m. Foi possível observar a maximização das recompensas ao longo da evolução de aprendizagem por episódios de treinamento. A arquitetura

neural foi treinada por 500 episódios. Como a recompensa obtida apresenta uma alta taxa de variação, apresenta-se a mesma por meio de uma média móvel simples ($N = 10$), objetivando estabilizar a representação do resultado. A Figura 7(b) mostra a evolução da taxa de transferência média (kbps) durante o processo de treinamento. Observa-se que a taxa média apresentou comportamento semelhante às recompensas obtidas, estabilizando próximo à $5,4kbps$, *bit rate* típico da SF7. A partir do episódio 300 o agente compreende que SF7 é a melhor ação a ser utilizada nesta situação.



(a) Rede DQN - 500 Episódios.

(b) Taxa de transferência (kbps).

Figura 7. Simulação Rede DQN com 500 Episódios.

5.2. Treinamento com parâmetros variáveis

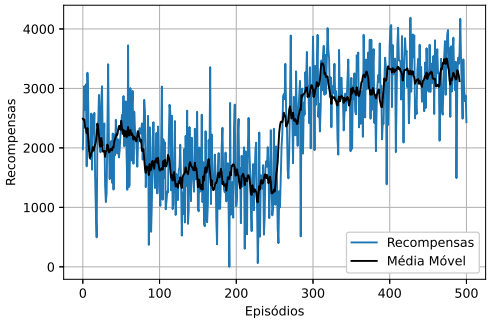
Como citado na seção 4, o treinamento com parâmetros variáveis consiste em alterar o número de dispositivos e as distâncias de forma homogênea. Neste experimento, o agente foi treinado por 500 e 1000 episódios de treinamento. A Figura 8(a) dispõe o resultado do **primeiro teste**: simulação do algoritmo DQN com 500 episódios de treinamento. Para ilustrar a influência dos hiperparâmetros, a Figura 8(b) apresenta um teste com taxa de aprendizagem distinta, parametrizada em $\alpha = 0,001$. É possível notar instabilidade na maximização das recompensas apenas alterando um hiperparâmetro da DQN. A Figura 8(c) dispõe o resultado do **segundo teste**: simulação do algoritmo DQN em 1000 episódios de treinamento.

As variações observadas no resultados são consequências das aleatoriedades das distâncias D e números de dispositivos finais parametrizados. Apesar destas variações, é possível verificar a tendência de maximização das recompensas obtidas durante o processo de aprendizagem. Com o aumento dos elementos de sequência da média móvel ($N = 100$) na Figura 8(d), é possível enfatizar essa tendência. A Tabela 3 apresenta um resumo dos resultados obtidos no treinamento em questão. O fator de desconto

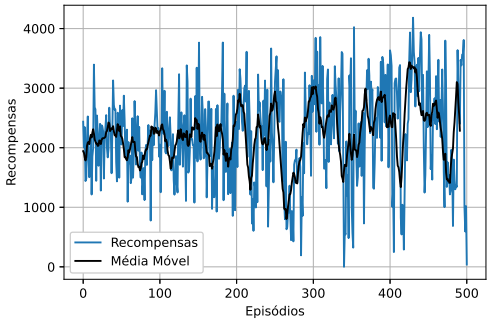
Tabela 3. Resultados do Treinamento.

Episódios	Taxa α	Taxa γ	Recompensa Média	SF (>Freq.)
500	0,0001	0,99	≈ 3100	7
500	0,001	0,99	≈ 3000	7
1000	0,001	0,99	≈ 3900	7

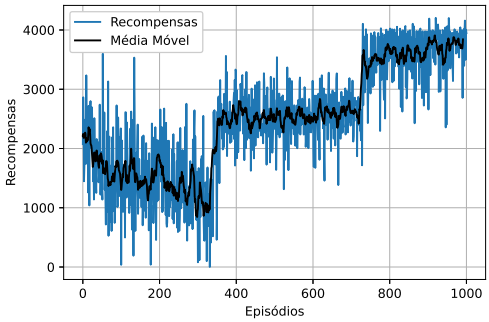
temporal γ foi mantido o mesmo para todos os testes $(0, 90)$. O agente com maior número



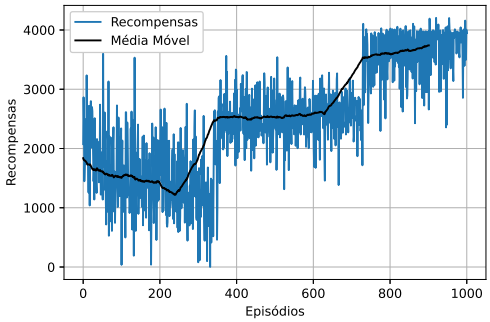
(a) Simulação da Rede DQN com 500 Episódios.



(b) 500 Episódios com alteração em α.



(c) Simulação da Rede *DQN* com 1000 Episódios.



(d) Simulação Rede *DQN* - 1000 Episódios (N=100).

Figura 8. Treinamento da Rede DQN com parâmetros variáveis.

de episódios (1000) obteve a maior recompensa média. A SF com maior frequência selecionada como ação foi SF7, isso em todos os experimentos (distância fixa e variável). Observa-se o viés do agente para a escolha deste fator de espalhamento espectral. Assim, os pacotes entregues no sistema foram otimizados, entretanto as distâncias envolvidas não foram satisfatoriamente consideradas no processo.

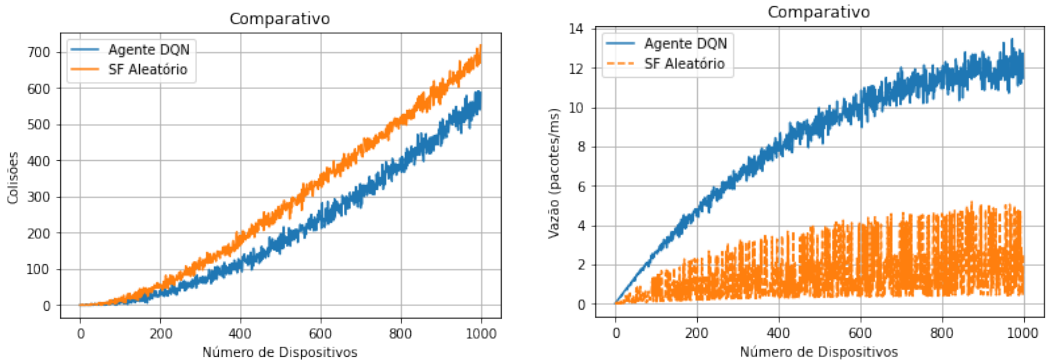
5.3. Resultados comparativos: Agente DQN x Aleatório (não inteligente)

Com a finalidade de testar o agente treinado, foi desenvolvido o comparativo entre um *gateway* aleatório de SF com o agente treinado por DQN. Neste teste foi avaliado apenas as colisões de pacotes com 1000 dispositivos. Com o objetivo de verificar a tendência de alteração de SF por parte do agente treinado, realizou-se alterações das distâncias entre *gateway* e *end node* utilizando uma distribuição gaussiana ($\mu = 1km, \sigma = 1km$).

Na Figura 9(a), é possível verificar que o agente inteligente conseguiu reduzir o número de colisões, bem como o PER (*Packet Error Rate*). O PER para o agente treinado é de $55,77\%$ com 1000 dispositivos transmitindo ao mesmo tempo. Já o PER para o agente aleatório é de $\approx 71\%$ com 1000 dispositivos transmitindo ao mesmo tempo.

A Figura 9(b) apresenta a vazão geral do sistema (pacotes/ms). Observa-se que o agente treinado por DQN obteve a melhor vazão: $\approx 13,5$ (pacotes/ms), com 1000 dispositivos transmitindo ao mesmo tempo. Sua vazão possui maior estabilidade em relação ao agente aleatório.

Apesar dos resultados apresentados acima levarem a compreensão da competência



(a) Comparativo Agente DQN e Aleatório.

(b) Comparativo da vazão geral do sistema.

Figura 9. Resultados comparativos.

do agente DQN, resultados não satisfatórios foram detectados. Ao verificar a política de ações tomadas, conclui-se que o agente tende a escolher a ação SF7, fator de espalhamento espectral este que possui maior taxa de transmissão e menor ToA. Esperava-se a comutação do SF com o aumento da distância e isso não ocorreu de forma satisfatória.

Ao investigar a função custo, nota-se um possível processo multiobjetivo. Por exemplo, o aumento do ToA levaria a uma redução da segunda parcela da Equação 4, entretanto as SF com alto ToA apresentam baixa capacidade de entrega de pacotes, reduzindo automaticamente a primeira parcela da equação. Ou seja, possivelmente existem dois objetivos independentes a serem tratados, otimização da distância e pacotes.

6. Conclusões

Ao observar os impactos que a modulação CSS influencia na eficiência de uma rede LoRa, foi proposta a aplicação do sistema inteligente capaz de intermediar e otimizar o processo. Com a implementação da metodologia de treinamento desenvolvida neste artigo, concluí-se que o agente treinado por aprendizado por reforço tende a selecionar os melhores fatores de espalhamento espectral (SF) em um processo de comunicação. Foi possível observar este resultado no treinamento com parâmetros fixos. Desta forma, a questão de pesquisa levantada neste artigo pode ser respondida de maneira positiva, ou seja, é possível implementar um sistema de aprendizagem por reforço em uma Rede LoRa para maximizar seu rendimento.

Em contra partida, após a obtenção dos resultados com parâmetros variáveis, verificou-se um possível problema multiobjetivo, o que levou a irrealização completa dos objetivos propostos. Atualmente a investigação deste estudo ocorre em responder se o problema em questão pode ser tratado de maneira mono objetiva, com a utilização de outra função custo a ser elaborada, envolvendo também outros parâmetros LoRA. Caso contrário, o algoritmo deverá ser tratado como multiobjetivo.

Apesar da existência de técnicas dedicadas ao controle do processo modulatório, como o ADR, a implementação do sistema baseado em inteligência artificial torna-se um método alternativo para controle dos parâmetros críticos. Com os resultados alcançados, foi possível concluir que o sistema treinado por redes DQN apresentou melhoria no controle das colisões de pacotes e taxa média de transferência de dados, quando comparado a um sistema seletor aleatório dos parâmetros SF.

Como trabalhos futuros, pretende-se reescrever a função objetivo do problema (função recompensa) através de outros parâmetros escalonáveis, de forma que a mesma se torne mono objetiva. Desta forma é necessário estudar e compreender outras variáveis do LoRa RF e LoRaWAN que podem contribuir para este fim. Planeja-se testar a aplicação de outros simuladores LoRaWAN que permitam desenvolver e representar a rede de comunicação de maneira mais fidedigna às situações práticas. Além disso, pretende-se, também, testar variantes do algoritmo DQN, como o *Double* DQN (DDQN) que podem melhorar o rendimento da resolução do problema em questão.

Referências

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376.

Augustin, A., Clausen, T., J.Yi, and Townsley, W. (2016). A study of lora: Long range low power networks for the internet of things. *MDPI - Sensors*, 4(2):16.

De Poorter, E., Hoebeke, J., Strobbe, M., Moerman, I., Latré, S., Weyn, M., Lannoo, B., and Famaey, J. (2017). Sub-ghz lpwan network coexistence, management and virtualization : an overview and open research challenges. *WIRELESS PERSONAL COMMUNICATIONS*, 95.

Hasegawa, S., Kim, S. J., Y.Shoji, and Hasegawa, M. (2020). Performance evaluation of machine learning based channel selection algorithm implemented on iot sensor devices in coexisting iot networks. *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC).*, 4(2):201–213.

Marais, J. M., Abu-Mahfouz, A. M., and Hancke, G. P. (2019). A review of lorawan simulators: Design requirements and limitations. In *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–6.

Park, G., Lee, W., and I.Joe (2020). Network resource optimization with reinforcement learning for low power wide area networks. *EURASIP Journal on Wireless Communications and Networking*, (176).

Raza, U., Kulkarni, P., and Sooriyabandara, M. (2017). Low power wide area networks: An overview. *IEEE Communications Surveys Tutorials*, 19(2):855–873.

Semtech (2019). An1200.22 lora modulation basics. Technical Report 2, Semtech - LoRa Alliance, Semtech Corporation.

Tesfay, A. A., Simon, E. P., Nevat, I., and Clavier, L. (2020). Multiuser detection for downlink communication in lora- like networks. *IEEE Access*, 8:199001–199015.

Weyn, M. (2016). Lpwan simulation. `https://github.com/maartenweyn/lpwansimulation`.

Xu, Z., Luo, J., Yin, Z., He, T., and Dong, F. (2020). S-mac: Achieving high scalability via adaptive scheduling in lpwan. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 506–515.