Aplicação do Método de Entropia Cruzada em Aprendizagem por Reforço para Controle de Fator de Espalhamento Espectral em Sistemas Internet das Coisas

Vittor Gomes de LIMA
Escola de Engenharia Elétrica e de Computação
Universidade Federal de Goiás
Goiânia, Brazil
https://orcid.org/0000-0001-8767-3002

Carlos Daniel de Sousa Bezerra
Instituto de Informática
Universidade Federal de Goiás
Goiânia, Brazil
https://orcid.org/0000-0002-0628-5276

Flávio Henrique Teles Vieira
Escola de Engenharia Elétrica e de Computação
Universidade Federal de Goiás
Goiânia, Brazil
https://orcid.org/0000-0003-3572-4036

Resumo— ***O número de dispositivos de Internet das Coisas (IoT) conectados cresce cada vez mais. Dependendo do cenário na prática, pode ocorrer tráfego intenso de dados no sistema de comunicação, dificultando que o seu desempenho se mantenha adequado devido aos congestionamentos e perdas de pacote por colisão. Este trabalho propõe um algoritmo inteligente baseado em aprendizado por reforço para atuar no espalhamento espectral da tecnologia LoRa de forma que a vazão da rede seja aumentada e o número de colisões de pacotes seja reduzido. É proposto utilizar o método de Entropia Cruzada associado à aprendizagem por reforço profundo para essa aplicação. A metodologia de pesquisa é por meio de simulações computacionais, implementação e validação do algoritmo proposto. Os resultados apontam para uma técnica de otimização e controle promissora comparada a outros algoritmos de aprendizagem por reforço profundo, principalmente para um número elevado de dispositivos.***

Palavras Chave — LoRaWAN, Otimização, Fator de Espalhamento Espectral, Internet das Coisas.

I. INTRODUÇÃO

A Internet das Coisas (*Internet of Things* - IoT) refere-se à conexão, por meio de enlaces de comunicação digital, 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. Sistemas baseados em Internet das Coisas podem ser aplicados por exemplo no monitoramento e controle de veículos autônomos, cidades inteligentes e processos industriais [1].

O protocolo LoRa (Long Range) é um dos mais utilizados em IoT, principalmente em redes de área ampla (*Long Power Wide Area Networks* - LPWAN) [7]. 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* [8].

O uso de Inteligência Artificial (IA) vem aumentando em diferentes ramos da ciência uma vez que os problemas vêm se tornando cada vez mais complexos e demandando soluções menos restritivas que se adaptem à natureza não trivial das dificuldades modernas [7] [6]. Nesses cenários, o emprego de Inteligência Artificial é ainda mais desejável visto que ela possibilita que sistemas sejam capazes de aprender e tomar decisões onde não há soluções adequadas claras.

O aprendizado por reforço (*Reinforcement Learning* - RL) é uma técnica de aprendizado de máquina e, portanto, um subcampo da inteligência artificial [11]. Problemas de RL são compostos basicamente por: um agente que realiza ações em um ambiente, uma função de 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 experiência e execute boas ações.

Um dos desafios na implementação de sistemas de comunicações é alocar eficientemente recursos. Seria necessária uma alta capacidade de processamento em uma estação rádio base celular para encontrar rapidamente a melhor solução considerando várias combinações. Algoritmos de aprendizagem de máquina podem ser empregados para encontrar a melhor solução de alocação de recurso de modo *offline* e depois prover rapidamente uma solução sob demanda.

O objetivo principal do presente trabalho é aplicar algoritmos de aprendizagem de máquina, especificamente aprendizagem por reforço, para controlar parâmetros em redes baseadas no protocolo LoRa de forma a otimizar seu desempenho. Propõe-se para esta aplicação em redes LoRa, utilizar o Método de Entropia Cruzada associado ao aprendizado por reforço profundo.

II. 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.

Pesquisadores da Universidade da Antuérpia, em [12], bem como os autores em [5] investigam o desempenho de redes LPWA para IoT. Em [12] é apresentado um simulador LoRaWAN para estudo e observação das colisões de pacotes. A partir dos resultados de simulação, esses autores avaliam a sensibilidade do protocolo LoRa em relação à conexão massiva de dispositivos. Os resultados mostram ainda, que a alteração dos parâmetros da modulação utilizados no protocolo LoRa modificam a eficiência da transmissão.

No trabalho de [4] os pesquisadores propuseram o algoritmo denominado EXPLoRa-AT. Este algoritmo tem

como objetivo igualar o tempo no ar da transmissão dos pacotes dos dispositivos LoRa envolvidos, focando em reduzir as colisões. Os autores concluíram que é possível atribuir valores para os fatores de espalhamento espectral (*Spreading Factors* - SFs) da tecnologia LoRa de modo a diminuir o número de colisões de pacotes e aumentar a probabilidade de sucesso das transmissões.

Em [2] foi proposto algoritmo de aprendizado por reforço para otimizar a conexão multiusuário em dispositivos IoT. Estes autores propõem efetuar controle de espalhamento espectral utilizando o algoritmo *Deep Q Networks* - (DQN), um algoritmo de *Reinforcement Learning* - (RL) que incorpora uma rede neural em seu modelo. Neste mesmo trabalho, a distância de um dispositivo ao *gateway* foi considerada. Além disso, para avaliação do algoritmo proposto, estes pesquisadores adaptaram para a linguagem *Python* um simulador de rede LoRa, disponível em https://github.com/maartenweyn. Este mesmo simulador de rede LoRa é adotado neste trabalho. Os resultados revelam eficiência da técnica proposta, reduzindo perda de pacotes por colisão e aumento da vazão.

De forma similar ao desenvolvido em [2], o presente trabalho propõe utilizar um método de Entropia-Cruzada associado a um algoritmo de aprendizagem por reforço no controle de espalhamento espectral, aplicação também realizada pelos autores em [2], mas utilizando uma rede DQN.

III. Proposta de Controle de Espalhamneto Espectral

A rede LoRa utiliza a modulação digital denominada *Chirp Spread Spectrum* - (CSS), um tipo de tecnologia proprietária desenvolvida pela Semtech [10]. 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), ii) fator de espalhamento espectral (*Spreading Factor* - SF), iii) 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 [10]. O SF representa a forma em que um bit (ou símbolo) é espaçado ou modulado, podendo assumir seis valores distintos. A taxa de dados R_b (bps) do LoRa é dada por:

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

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 (*Time on Air* - ToA). Entretanto, este aumento do SF faz com que a taxa de transferência de dados decaia e a potência de transmissão aumente, comprometendo a eficiência energética dos módulos.

Diante destas características da tecnologia LoRa, vislumbra-se a possibilidade do parâmetro SF ser otimizado durante as comunicações entre dispositivos LoRa.

3.1) Entropia Cruzada e Aprendizado por Reforço

Um sistema de aprendizado por reforço visa fornecer conhecimento a um determinado agente por meio de interações com um ambiente. As qualidades das ações deste agente são medidas por meio da função de recompensa.

O método Entropia Cruzada (*Cross-Entropy*) é um algoritmo que pode ser utilizado para treinar agentes RL descrito em [3]. O método de Entropia Cruzada descrito em [3] é considerado um Algoritmo Evolutivo: alguns indivíduos são amostrados de uma população, e apenas os da "elite" governam as características das gerações futuras.

Essencialmente, o método *Cross-Entropy* seleciona várias entradas de algoritmo, analisa suas saídas e escolhe aquelas que levam às melhores saídas. Em seguida, ajusta o agente até que os resultados desejados sejam obtidos. O algoritmo cria uma distribuição Gaussiana $N(\mu, \sigma)$ que descreve os pesos θ da rede neural. Em uma segunda instância, ele amostra lotes de tamanho N com θ usando uma distribuição Gaussiana; todas essas amostras são avaliadas usando a função de custo da rede neural. Em seguida, seleciona as melhores amostras θ e calcula os novos μ e σ para parametrizar a nova distribuição gaussiana. O algoritmo se repete até a convergência.

Em um algoritmo RL, uma política $\pi(a|s)$ determina a ação que o agente deve tomar para cada estado observado. No método *Cross-Entropy*, a política que é inicializada aleatoriamente é melhorada episodicamente ajustando os parâmetros da rede neural. Repetir o processo gradualmente melhora a política para uma determinada tarefa.

3.2. Simulação do Agente Inteligente

Pode-se representar o ambiente de comunicação por meio de um simulador. No simulador, considera-se que 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.

O simulador de comunicação LoRa é implementado em *Python* com base no trabalho e experimentos desenvolvidos por [12]. Após a implementação do simulador e observação dos resultados de simulação do ambiente de comunicação LoRa, desenvolvemos os algoritmos de RL a serem aplicados neste ambiente para melhorar o desempenho deste tipo de rede.

A distância entre o dispositivo final e *gateway* também é um critério importante, por esta razão é adicionado na função de recompensa do algoritmo de aprendizagem por reforço, um termo referente à relação *distâncias-SF* (Ver Equação 2). Assim, o agente inteligente a ser treinado pode selecionar uma ação levando em consideração tanto as colisões de pacotes quanto as distâncias entre os *end-nodes* e *gateway*.

Para o desenvolvimento do agente inteligente capaz de selecionar de forma eficiente os parâmetros de modulação LoRa, propõe-se considerar três pontos essenciais: i) colisões de pacotes, ii) distância entre módulos e *gateway* e iii) dispositivos conectados em cada SF. O agente proposto considera o cenário onde tem-se um *gateway* conectado a n-dispositivos que representam sensores IoT, ou receptores

LoRa denominados (*end-nodes*). Estes *end-nodes* são distribuídos de com distâncias aleatórias e heterogêneas.

Neste trabalho, os parâmetros do *frame* que influenciam no cálculo do ToA (*Time on Air*), 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 trabalho.

TABELA 1. TOA ESTIMADO PARA CADA SF.

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

A Figura 1 dispõe o diagrama de blocos que contextualiza o agente inteligente que atua através de aprendizagem por reforço para otimização do desempenho da rede LoRa, contendo as ações, os estados observáveis e a recompensa imediata obtida.

O espaço de estados do sistema de aprendizado por reforço é definido por meio de oito variáveis observáveis no processo: i) distância do atual dispositivo conectado ao *gateway*, ii, iii, iv, v, vi, vii) Número de dispositivos utilizando o SF mostrado, viii) pacotes entregues. O agente deve aprender a definir um valor adequado de 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*, parte da função de recompensa imediata R_i obtida pelo agente é dada por:

$$R_i = \frac{1}{|D_{norm} - SF_{norm}|}, \quad (2)$$

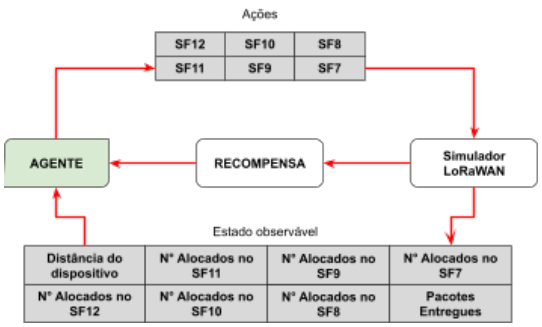

Figura 1. Ações, Estados e Recompensas

onde a distância *Dnorm* o número do SF normalizado *SFnorm* são normalizados usando a fórmula de normalização Min-Max:

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3)$$

Assim, segundo a Equação 2, se a distância entre os dispositivos da rede for próxima de 7 metros, o valor de recompensa será maior se o agente escolher um SF maior. Utiliza-se a ideia do algoritmo EXPLoRa-AT que é de igualar o tempo no ar dos SFs. Um dispositivo com SF12 possui o ToA de 1646ms, conforme mostrado na Tabela 1, ou seja, se um dispositivo for alocado no SF12, outros 23 poderão ser alocados no SF7 por exemplo, já que o ToA do SF7 é de 70ms. Considerando o máximo de 1000 dispositivos na rede, é calculado também o máximo de dispositivos que podem ser alocados em cada SF, de modo que o tempo no ar se iguale. Caso o agente escolha um SF, de modo que ultrapasse o número máximo de dispositivos dado na Tabela 2, o agente é penalizado.

TABELA 2. DISPOSITIVOS QUE PODEM SER ALOCADOS EM CADA SF.

	SF12	SF11	SF10	SF9	SF8	SF7
Máximo Número de Dispositivos	20	36	72	143	259	470

A recompensa imediata (R) proposta para monitorar o desempenho de cada ação no simulador é dada pela função:

$$R = \frac{1}{|D_{norm} - SF_{norm}|} \times Pac_{norm} \quad (4)$$

O agente em situações ideais pode receber uma recompensa de até 650 de acordo com a Equação 4. Caso o agente escolha um SF de modo que ultrapasse o número máximo de dispositivos a ser alocado naquele SF, propomos utilizar uma função de recompensa que penalize o agente em -2000, de acordo com a seguinte Equação:

$$R = \left(\frac{1}{|D_{norm} - SF_{norm}|} \times Pac_{norm}\right) - 2000 \quad (5)$$

onde *Dnorm* é a distância normalizada conectado ao gateway, *SFnorm* é o número de SF escolhido normalizado e *Pacnorm* é o número de pacotes entregues também normalizado. Ao longo do processo de aprendizagem esta função (Equação 5) deve ser maximizada.

3.2) Configuração da Rede Neural

O modelo da rede neural proposta para o desenvolvimento do algoritmo de RL é composto por 4 camadas totalmente conectadas. As duas camadas ocultas possuem 64 neurônios cada e utilizam funções de ativação ReLU. A camada de saída da rede neural possui 6 neurônios e função de ativação linear. Note que o número de neurônios da camada de saída deve coincidir com o tamanho do espaço de ações. A métrica de avaliação do desempenho da rede neural é o Erro Médio Quadrático (*Mean Square Error* - MSE).

As entradas da rede neural são normalizadas para evitar problemas denominados de explosão de gradientes, que são falhas no processo de aprendizagem [9]. Em outras palavras, a normalização das entradas melhora a estabilidade da rede neural.

3.3) Aprendizado do Agente

Uma vez que consideramos uma rede neural como a principal componente deste Agente, é preciso encontrar uma maneira de selecionar dados de treinamento, que inclui dados de entrada e seus respectivos rótulos.

O treinamento do agente pode ser visto como um problema de aprendizagem supervisionado onde os estados observados são considerados os recursos (dados de entrada) e as ações constituem os rótulos.

Durante a vida do agente, sua experiência é apresentada como episódios. Cada episódio é uma sequência de observações de estados que o agente tem do Ambiente, ações que têm emitido e recompensas por essas ações. A essência do método *Cross-Entropy* é descartar as ações ruins tomadas e treinar o agente com as melhores. Para cada episódio, são considerados 1000 dispositivos com distâncias diferentes. São realizadas três simulações por episódio para o agente. Em cada simulação, são adicionados os dispositivos um a um.

A Tabela 3 apresenta um exemplo de como será a escolha da melhor ação em cada passo de um episódio. O mesmo estado (distância no caso) de cada passo do episódio, é passado para cada uma das três simulações. No início, o agente não possui nenhum conhecimento sobre o ambiente então toma majoritariamente ações randômicas. Como nas três simulações são adicionados os mesmos dispositivos, o que varia então são as ações do agente. Devido à aleatoriedade inicial, as ações em cada simulação tendem a ser diferentes, mesmo se tratando do mesmo estado. Ações diferentes geram recompensas diferentes, sendo assim, a melhor ação tomada para cada passo do episódio, aquela que gera a maior recompensa, é guardada na memória do agente.

TABELA 3. EXEMPLO DE ESCOLHA DA MELHOR AÇÃO.

Dispositivo	Distância do Dispositivo	Simulação 1- Recompensa obtida	Simulação 2- Recompensa obtida	Simulação 3- Recompensa obtida
1	0.765	10	20	**30**
2	3.245	**30**	20	10
3	6.234	20	**30**	10
. . .	. . .	. . .	. . .	. . .
1000	1.222	20	**30**	10

Como pode ser observado na Tabela 3, é decidido guardar a ação que gera a melhor recompensa, ou seja, é guardado um terço das experiências do agente. Essa pequena parte é denominada “elite” e servirá para o treinamento das redes neurais. Com isso, a cada episódio o agente terá maior conhecimento para tomada de decisão e aos poucos deixará de tomar ações aleatórias (*Exploration and Exploitation*).

O Agente então tenta acumular o máximo de recompensa total possível interagindo com o Ambiente. Como resultado, a rede neural aprende a repetir ações que levam aos melhores resultados à medida que usamos novos lotes (*batch*) de episódios de elite. O Agente deve ser treinado até que uma certa recompensa média para o limite de episódios seja alcançada.

IV. RESULTADOS E DISCUSSÕES

A Figura 2 ilustra o resultado da implementação do simulador LoRaWAN proposto por [12] com as adaptações necessárias. Avalia-se nas simulações da Figura 2 a obtenção dos resultados de colisões entre pacotes com utilização de até 1000 (mil) dispositivos que transmitem ao mesmo tempo, cujo acesso ao meio é determinado pelo protocolo ALOHA [8]. A seleção dos valores de SF foi inicialmente aleatória, sem controle específico, onde cada mensagem possui o tamanho de 25 bytes. A Figura 3 apresenta o resultado da simulação com a seleção do SF fixo (SF12). É possível observar o aumento do número de colisões de pacotes transmitidos com o aumento do número de dispositivos finais.

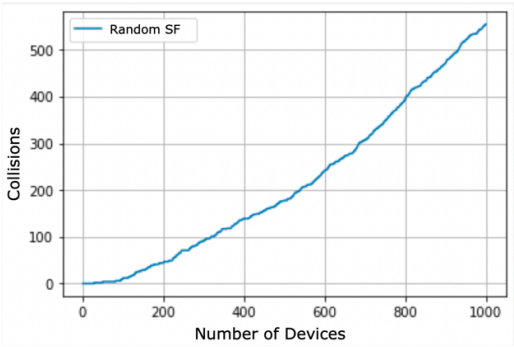

Figura 2. Simulação Randômica

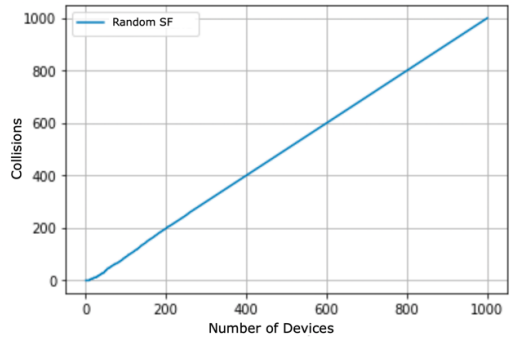

Figura 3. Simulação com SF12

Na simulação com SF aleatório, com 1000 dispositivos obteve-se ≈ 57\% de colisões. Já fixando SF em 12 e para 1000 dispositivos, nenhum pacote foi entregue. Observa-se, portanto, que a escolha do SF impacta diretamente nas colisões de pacotes.

4.1) Treinamento do Agente Inteligente

O agente inteligente é colocado para atuar no cenário de simulação. Após um total de 250 episódios, é possível perceber na Figura 4 que as recompensas acumuladas por episódio cresceram consideravelmente.

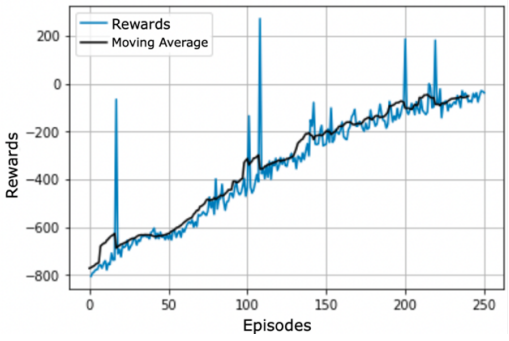

Figura 4. Curva de Aprendizado do Agente para 250 Episódios

Conclui-se, portanto, que o agente está treinado e preparado para um teste de validação, onde o objetivo é comparar seus resultados com os de outros métodos. Esta comparação é realizada na próxima seção.

4.2) Validação do Agente Inteligente

Com a finalidade de avaliar o agente treinado com o algoritmo proposto neste trabalho (Entropia-Cruzada), comparou-se seu desempenho com o de um *gateway* alocando SF de forma aleatória aos dispositivos e um agente DQN (*Deep Q Networks*) baseado no trabalho descrito em [2]. Nestes testes foi avaliado o desempenho dos algoritmos em uma rede com até 1000 dispositivos.

Na Figura 5 é possível verificar que o agente proposto conseguiu reduzir o número de colisões, bem como o PER (*Packet Error Rate*). O PER para o agente treinado com o método de Entropia Cruzada é de aproximadamente 31,8% com 1000 dispositivos transmitindo ao mesmo tempo. O PER para o agente DQN é de aproximadamente 46,2% com 1000 dispositivos transmitindo ao mesmo tempo. Já o PER para o agente aleatório é de ≈ 54,6% com 1000 dispositivos transmitindo ao mesmo tempo.

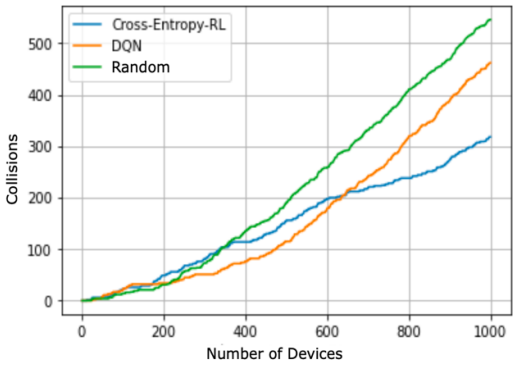

Figura 5. Comparativo entre Colisões

A Figura 6 apresenta a vazão geral do sistema (*pacotes/ms*). Observa-se que o agente treinado pelo algoritmo proposto obteve a melhor vazão: ≈ 18,9 (*pacotes/ms*), com 1000 dispositivos transmitindo ao mesmo tempo. De fato, a partir de aproximadamente 625 dispositivos o método de Entropia-Cruzada provê maior vazão para a rede do que os outros algoritmos considerados.

Com o objetivo de verificar a tendência de alteração de SF por parte do agente treinado em relação à distância entre o *gateway* e o *end node*, é gerada uma tabela que apresenta o estado do ambiente e a ação tomada pelo agente.

Observa-se pela Figura 7 que para os dispositivos com maiores distâncias, o agente atribui maiores SFs e para os dispositivos mais próximos do *gateway*, o agente atribui SFs menores. Além disso, para garantir baixa perda por colisão de pacotes, se muitos dispositivos são adicionados na rede, o agente proposto passa a atribuir o menor SF (7) independentemente da distância.

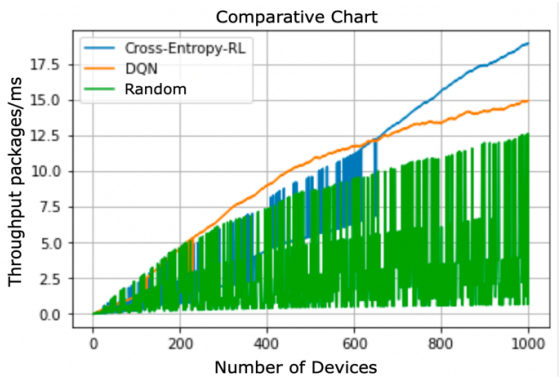

Figura 6. Comparativo da Vazão Geral do Sistema

É notado que o agente não atribui o SF12 para nenhum dispositivo. Em outras palavras, pode-se entender que possivelmente o agente adotou em sua política, que a escolha do SF12 tende a gerar colisões no longo prazo.

Considerando os resultados obtidos, fica claro que o uso de métodos de aprendizado por reforço pode ajudar no desenvolvimento de sistemas de comunicação.

	device distance	SF12	SF11	SF10	SF9	SF8	SF7	packages delivered	chosen SF
0	3.075	0	0	0	1	0	0	1000	9
1	2.425	0	0	0	1	1	0	1000	8
2	5.038	0	1	0	1	1	0	1000	11
3	3.809	0	1	1	1	1	0	1000	10
4	5.479	0	2	1	1	1	0	1000	11
5	1.874	0	2	1	1	2	0	1000	8
6	4.133	0	2	2	1	2	0	1000	10
7	5.039	0	3	2	1	2	0	1000	11
8	6.559	0	4	2	1	2	0	1000	11
9	5.262	0	5	2	1	2	0	1000	11
10	6.415	0	6	2	1	2	0	1000	11
11	1.630	0	6	2	1	3	0	1000	8
12	5.677	0	7	2	1	3	0	998	11
13	3.983	0	7	3	1	3	0	998	10
14	6.606	0	8	3	1	3	0	998	11

Figura 7. Dataframe de Ações do Agente

CONCLUSÕES

Ao observar os impactos da modulação CSS (*Chirp Spread Spectrum*) na eficiência de uma rede LoRa, foi proposto um agente inteligente baseado no Método de Entropia-Cruzada associado ao aprendizado por reforço capaz de intermediar e melhorar o desempenho desta rede. Com a implementação da metodologia de treinamento desenvolvida neste trabalho, conclui-se que o agente treinado por aprendizado por reforço tende a selecionar os melhores fatores de espalhamento espectral (SF) em uma rede de comunicação LoRa. Além disso, conclui-se que o algoritmo proposto de Entropia-Cruzada provê maior vazão e menor número de colisões para rede LoRa do que outros algoritmos de aprendizado por reforço profundo quando se tem um número de dispositivos maior do que um certo valor. O código completo utilizado no artigo está disponível em https://github.com/VittorLima/RL-agent-LoRa.git.

Referências Bibliográficas

[1] 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.

[2] Bezerra, C., Oliveira-Jr, A., and Vieira, F. (2021). Proposta de controle de espalhamento espectral utilizando aprendizado por reforço profundo para otimizacão do desempenho de redes LoRa/LoRaWAN. In Anais da IX Escola Regional de Informática de Goiás, pages 54–67, Porto Alegre, RS, Brasil. SBC.

[3] Boer, P.-T., Kroese, D., Mannor, S., and Rubinstein, R. (2005). A tutorial on the cross-entropy method. Annals of Operations Research, 134(1):19–67.

[4] Cuomo, F., Campo, M., Caponi, A., Bianchi, G., Rossini, G., and Pisani, P. (2017). EXPLoRa: extending the performance of LoRa by suitable spreading factor allocations. 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pages 1–8.

[5] 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(1):187–213.

[6] Jang, S., Yoon, H., Park, N., Yun, J., and Son, Y. (2019). Research trends on deep reinforcement learning. Electronics and Telecommunications Trends, 34(4):1–14.

[7] Park, G., Lee, W., and Joe, I. (2020). Network resource optimization with reinforcement learning for low power wide area networks. EURASIP Journal on Wireless Communications and Networking, 2020(1):1–20.

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

[9] Salimans, T. and Kingma, D. (2016). Weight normalization: a simple reparameterization to accelerate training of deep neural networks. Advances in Neural Information Processing Systems, 29.

[10] Semtech (2019). AN1200.22 LoRa modulation basics. Technical Report 2, Semtech-LoRa Alliance, Semtech Corporation.

[11] Sutton, R.-S. and Barto, A.-G. (1998). Reinforcement learning: an introduction. MIT Press. Also translated into Japanese and Russian.

[12] Weyn, M. (2016). LPWAN simulation. https://github.com/maartenweyn/lpwansimulation.