Avaliação de Desempenho de Algoritmos de *Machine Learning* para Otimização de Simulações de Redes de Computadores

Joyce Quintino, Silas Santiago, Reinaldo Braga, Mauro Oliveira, Carina Oliveira

[1]Laboratório de Redes de Computadores e Sistemas (LAR)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Campus Aracati
Aracati – CE – Brasil

joycequintino11@gmail.com,(silas,reinaldo,mauro,carina)@lar.ifce.edu.br

Abstract. *Network simulators are computational tools capable of assisting in the analysis and improvement of communication protocols in different scenarios before implementation in a real environment. However, the execution time of the simulations and the allocated computational resources tend to increase as the requirements of complexity and precision increase. Besides, many simulation processes have traces without logical interpretation as their only output. In this sense, this work evaluates the performance of Machine Learning algorithms in the optimization of computer network simulations. The achieved results show that the algorithms are capable of making assumptions close to the simulations performed, reaching precision in the results above 90%.*

Resumo. *Os simuladores de redes são ferramentas computacionais capazes de auxiliar na análise e aperfeiçoamento de protocolos de comunicação em diferentes cenários antes da implantação em um ambiente real. No entanto, o tempo de execução das simulações e os recursos computacionais alocados tendem a crescer à medida que os requisitos de complexidade e precisão aumentam. Além disso, muitos processos de simulação possuem como única saída traces sem interpretação lógica. Neste sentido, este trabalho avalia o desempenho de algoritmos de Machine Learning na otimização de simulações de redes de computadores. Os resultados alcançados mostram que os algoritmos são capazes de fazer suposições próximas das simulações realizadas, alcançando precisões acima de 90%.*

1. Introdução

Os simuladores são ferramentas computacionais utilizadas em vários campos da ciência e da engenharia [Lee et al. 2017]. Os recursos disponibilizados por eles permitem a experimentação de determinada solução antes de colocá-la em produção. Tal aspecto reduz os riscos associados à criação de uma nova solução, assim como reduz os riscos de uma mudança em uma solução já existente. Consequentemente, os simuladores possibilitam a redução significativa dos custos e do tempo de uma implantação [Abreu et al. 2017].

Os simuladores são utilizados nos mais diversos tipos de sistemas e aplicações. Na área de redes de computadores, os simuladores são capazes de reproduzir o comportamento de uma rede de comunicação real. Alguns exemplos mais populares de simuladores de redes são o Cisco *Packet Tracer*, OMNet++, *Optimized Network Engineering Tool*

(OPNET), QualNet, *Network Simulator* 2 (NS-2) e 3 (NS-3). Esses e outros inúmeros simuladores de redes fornecem ambientes e ferramentas para os mais diversos fins, partindo desde o auxílio no entendimento de conceitos de redes, muitas vezes disponibilizando interfaces gráficas animadas, até o desenvolvimento, implantação e análise de desempenho de complexos protocolos e arquiteturas. No mais, também permitem simular diferentes tipos de aplicações, modelar o tipo de tráfego e a interação entre diferentes entidades de uma rede (ex: roteadores, comutadores, pontos de acesso, etc).

Apesar dos inúmeros benefícios do uso de simuladores na área de redes, eles também apresentam limitações relativas ao seu uso. O tempo de execução das simulações e, consequentemente, os recursos computacionais alocados para execução das simulações crescem à medida que os requisitos de complexidade e precisão da rede aumentam. Nesse contexto, destaca-se que as Redes do Futuro (*Networks of the Future* - NoF) tendem a ser cada vez mais heterogêneas, densas e complexas. Como exemplos de ambientes NoF, existem as *smart cities*, Internet das Coisas (IoT), redes *Machine-to-Machine* (M2M), Sistemas de Transporte Inteligente (STI), *green networking*, etc [Santos et al. 2016]. Assim, realizar experimentos que envolvam a simulação desses ambientes muitas vezes exige recursos computacionais de processamento e armazenamento que nem sempre estão disponíveis. Quando disponíveis, podem levar muito tempo para simular (horas, dias, meses etc). Também há o custo de energia alocado para a realização das simulações, indo na contramão da tendência mundial de *green computing* [Saha 2018].

Como contribuição nesse sentido, este trabalho avalia o desempenho de algoritmos de *Machine Learning* (ML) na otimização de simulações de redes de computadores. Em particular, é avaliado se os algoritmos conseguem predizer satisfatoriamente o mesmo comportamento alcançado pelas simulações a partir de um conjunto de discriminantes estatísticos obtidos para cenários de rede simulados. Quatro algoritmos de ML (i.e., Rede *MultiLayer Perceptron* - MLP, *Decision Tree* - DT, *Support Vector Machine* - SVM e *Random Forest* - RF) são treinados a partir de um *dataset* de resultados de simulações realizadas no *Network Simulator* 3 (NS-3) [Riley and Henderson 2010]. A rede utilizada como estudo de caso é uma Rede em Malha Sem Fio (*Wireless Mesh Network* - WMN) [Akyildiz et al. 2005]. A proposta tem como foco predizer a taxa de entrega de diferentes cenários de WMN, uma métrica importante no gerenciamento da rede. Os resultados alcançados mostram que os algoritmos fazem suposições próximas das simulações reais, tendo alguns modelos de ML com taxa de acerto acima de 90% nas predições da taxa de entrega.

A sequência do trabalho está organizada do seguinte modo: a Seção 2 exibe os trabalhos relacionados à simulação de redes e ML; a Seção 3 apresenta o detalhamento da proposta e os algoritmos de *Machine Learning* aplicados na predição da taxa de entrega; a Seção 4 mostra os resultados obtidos; e, por fim, a Seção 5 apresenta as conclusões e os direcionamentos para trabalhos futuros.

2. Trabalhos Relacionados

Hwang *et al.* [Hwang et al. 2017] propõem uma solução para a problemática da perda de pacotes em redes com e sem fio (usando o TCP *NewReno* na camada de Transporte) com base no algoritmo de ML *Single Layer Perceptron*. Essa solução é aplicada no algoritmo de controle de congestionamento quando a causa da perda de pacotes é prevista

como um congestionamento de rede. Quando previsto como um erro aleatório, o pacote perdido é retransmitido. A solução proposta resultou em um algoritmo chamado de controle de congestionamento da perda de pacotes com e sem fio usando ML (ML-TCP) com o intuito de auxiliar no desempenho do TCP. Porém, os autores abordam o fato do TCP ser adequado apenas para uma rede com fio e com baixa taxa de erro de bit. Diante disso, os autores ressaltam a importância de melhorias no desempenho do protocolo. Os resultados da simulação mostraram que o ML-TCP possui maior precisão para prever a causa da perda de pacotes em diferentes ambientes.

Kajita *et al.* [Kajita et al. 2015] projetaram estimadores de atraso e vazão para canais 2.4GHz com interferência. Os autores concentram esforços em projetar duas funções de estimação que possam ser utilizadas por cada ponto de acesso para escolha de um canal baseado na observação do tráfego de outros pontos de acesso. Os dois estimadores visam calcular o atraso e a vazão a partir de um conjunto de observações sobre o ponto de acesso. Desta forma, a pesquisa teve o intuito de permitir ao ponto de acesso um comportamento autônomo ao estimar corretamente o *status* de outro ponto de acesso sem a necessidade de alternar o canal. Para a construção do modelo preditivo, os autores utilizaram cerca de 10.000 cenários de simulação e uma abordagem supervisionada SVM e análise de regressão. Os autores realizaram o treinamento da técnica SVM para obter um classificador binário a partir de dados rotulados como saturado ou não saturado. A validação do modelo preditivo foi realizada a partir de 2.000 experimentos de simulação com cenários distintos. Os resultados obtidos com validação cruzada demonstraram que a função de estimação foi capaz de estimar a vazão com um erro médio inferior à 10%.

Saggioro *et al.* [Saggioro et al. 2012] propõem a ferramenta *TraceMetrics* como alternativa para a obtenção de medidas de interesse em simulações no NS-3. A *TraceMetrics* analisa cada linha do *trace* gerado pelo simulador e obtém, a partir dele, medidas de interesse. Os autores destacam que a principal vantagem está na flexibilidade, uma vez que se tem o registro de tudo o que aconteceu na simulação, pacote a pacote, podendo-se calcular qualquer medida desejada. A ferramenta também oferece suporte a algumas medidas de acordo com os tipos definidos: medidas calculadas por nó na simulação e calculadas por fluxo. Como exemplos de medidas suportadas, tem-se: quantidade e tamanho médio dos pacotes enviados, recebidos e descartados; quantidade de dados enviados, recebidos e descartados; *throughput* e *goodput*; atraso fim-a-fim por pacote, média e variância. Contudo, os autores ressaltam que a desvantagem está no desempenho, pois são necessários acessos à disco e armazenamento de informações em memória.

Diferentemente dos trabalhos apresentados, o presente trabalho propõe otimizar simulações usando algoritmos de *Machine Learning* na predição de resultados de métricas de desempenho de cenários em redes, como a taxa de entrega. Essa otimização é capaz de evitar que simulações desnecessárias sejam executadas, diminuindo o tempo de execução das simulações.

3. Proposta

Nesta Seção, é detalhado todo o processo metodológico de implementação da solução de otimização de simulações de redes de computadores através de algoritmos de *Machine Learning* (ML). A Figura 1 ilustra as principais etapas de tal processo, que são apresentadas e discutidas nas seções a seguir.

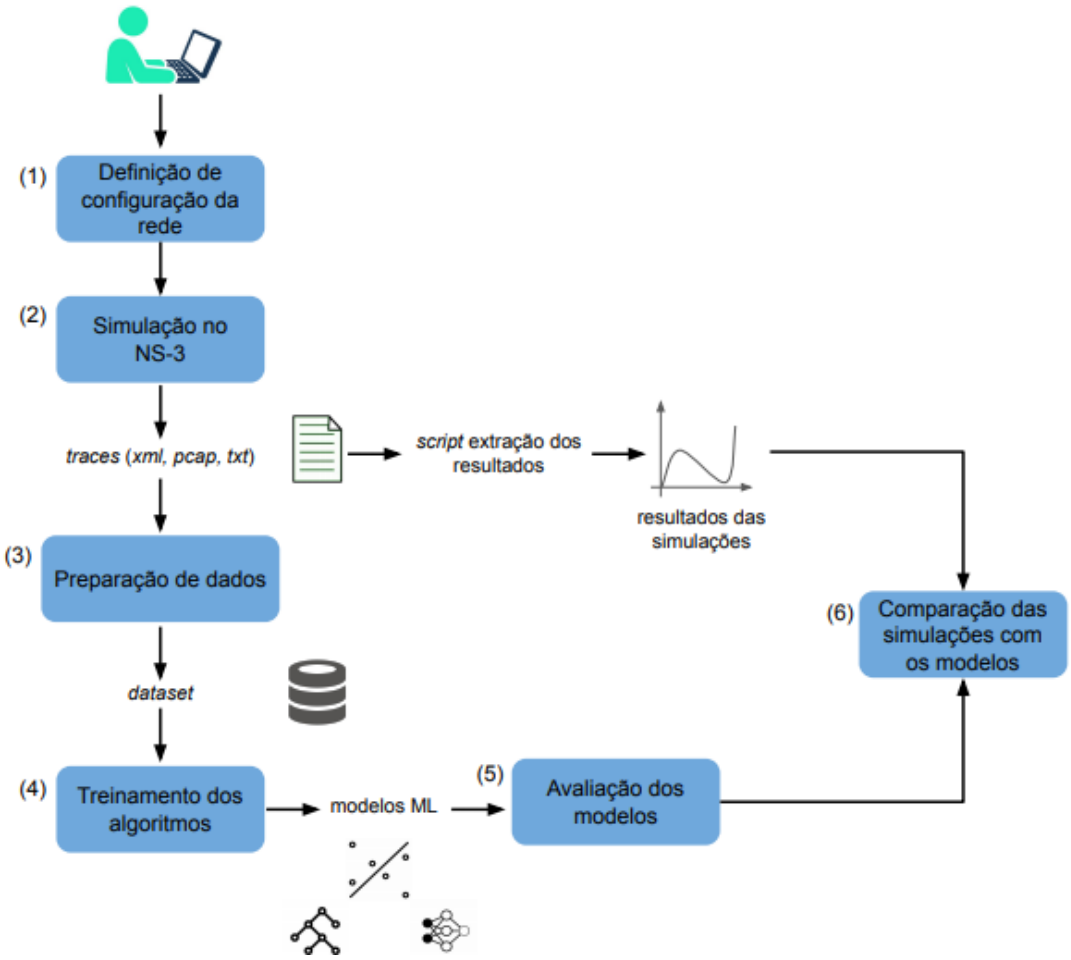


Figura 1. Visão geral da proposta.

Etapa 1: Definição de configuração da rede

O presente trabalho usa como estudo de caso um conjunto de cenários de Redes em Malha Sem Fio (*Wireless Mesh Network* - WMN) [Akyildiz et al. 2005]. As WMNs são consideradas um tipo de Rede do Futuro e têm atraído cada vez mais atenção por causa de seu baixo custo, facilidade de implantação, robustez e cobertura de serviço confiável. Estas redes são capazes de oferecer uma cobertura sem fio em banda larga para grandes áreas (como *smart cities*), sem exigências em termos de infraestrutura, ao mesmo tempo em que são capazes de garantir a conectividade em ambientes com dinamicidade das condições do meio sem fio e com usuários móveis [Silva et al. 2018]. As WMNs são padronizadas pelo IEEE como uma solução de rede em malha para difusão (*broadcast*) e entrega de pacotes *unicast* ao longo de uma topologia de múltiplos saltos auto-configurável. O padrão é chamado de IEEE 802.11s *Mesh Networking* [IEEE 802.11s 2011] e propõe, entre outros serviços de malha, a seleção de caminhos e o encaminhamento com competências de roteamento na camada MAC, interoperabilidade com redes externas e soluções de segurança.

O cenário base utilizado nos experimentos de simulação segue uma topologia de rede em malha na qual um fluxo de dados UDP é gerado entre dois nós, sendo um deles o gerador de tráfego e o outro o receptor (*gateway*), conforme ilustra a Figura 2. A camada física utiliza na sua configuração o padrão IEEE 802.11n a uma frequência de 5 GHz.

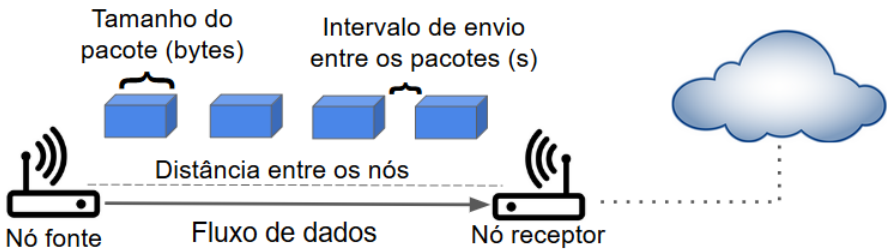


Figura 2. Topologia *Mesh* configurada para o estudo de caso.

A taxa de entrega da rede é a métrica em questão no estudo, ou seja, é ela que os algoritmos de ML fazem suposições e tentam predizer. Logo, baseado na topologia da Figura 2, este trabalho tem como foco a mensuração e análise de três parâmetros da rede que impactam diretamente na taxa de entrega: a distância entre nós, o tamanho dos pacotes e intervalo de envio entre os pacotes. A Tabela 1 detalha os valores assumidos por cada parâmetro na criação de diferentes cenários *mesh*.

Tabela 1. Valores dos parâmetros variados nas simulações.

Parâmetro	**Valores**
Distância entre os nós	50 a 110 metros
Tamanho dos pacotes	256, 512, 1024 bytes
Intervalo de envio entre os pacotes	0.1, 0.01 segundos

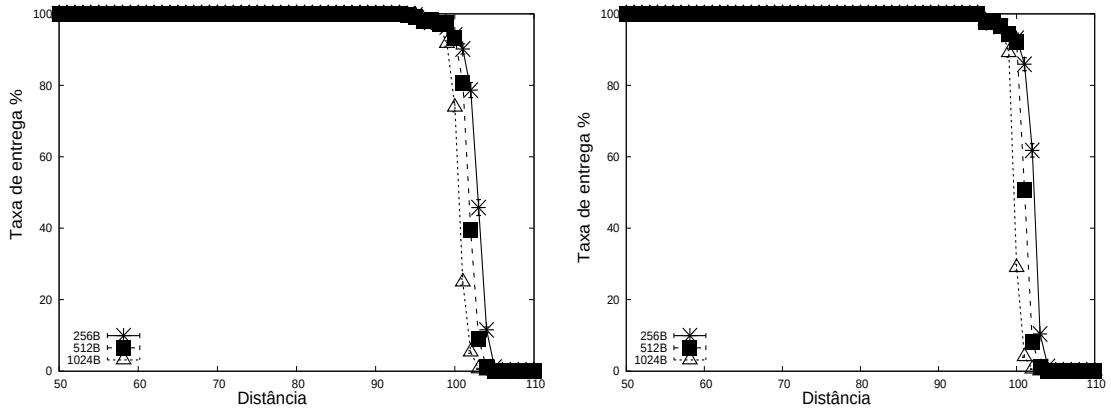
Etapa 2: Simulação no NS-3

Como mostra o fluxo da Figura 1, definidas as configurações da rede para o estudo, os cenários *mesh* são simulados no *Network Simulator* 3 (NS-3). É importante destacar que, apesar do simulador disponibilizar a implementação do padrão de WMN IEEE 802.11s [Andreev et al. 2010], foi necessária a implementação de algumas extensões com o objetivo de capturar as informações desejadas para o trabalho.

No total, foram realizadas 16.200 execuções de simulações para obtenção dos resultados. A Equação 1 justifica esse total, sendo v_{Dist} os 60 valores de distância utilizados, $v_{TamPacote}$ os 3 valores de tamanho de pacote e $v_{IntPacotes}$ os 3 valores de intervalo de envio entre os pacotes, conforme apresentado anteriormente na Tabela 1. Além disso, cada combinação de parâmetro é simulada 30 vezes para considerar as variações presentes nas amostras [Jain 1991]. Esse processo de execução das simulações é o maior responsável pelo longo tempo de resposta do simulador, o grande consumo de processamento e de energia.

$$Total_{\text{Execuções}} = v_{Dist} \cdot v_{TamPacote} \cdot v_{IntPacotes} \cdot 30 \quad (1)$$

As Figuras 3(a) e 3(b) apresentam os resultados da taxa de entrega em função da distância para os três diferentes tamanhos de pacotes e para os intervalos de envio entre os pacotes de 0.1s e 0.01s, respectivamente. A extração dos valores da taxa de entrega dos *traces* e o cálculo do intervalo de confiança são feitos com o auxílio de um *script* em *Python*. Também é usado o programa *gnuplot* para plotar os gráficos, já que o NS-3 não disponibiliza esse tipo de funcionalidade para representação dos fluxos.



(a) Intervalo de envio entre os pacotes de 0.1s (b) Intervalo de envio entre os pacotes de 0.01s

Figura 3. Resultados da Simulações no NS-3: Taxa de entrega em função da Distância entre os nós.

Em todos os casos, a taxa de entrega é de 100% para curtas distâncias e tende a diminuir até alcançar 0%. Nota-se que para pacotes de maior tamanho (como 1024B) e menores intervalos de envio entre os pacotes, a queda da taxa de entrega ocorre mais rapidamente. É importante lembrar que quanto maior a distância entre dois nós, menor é a relação sinal-ruído (*Signal-to-Noise Ratio* - SNR) e, portanto, maior é a taxa de erro de bit (*Bit Error Rate* - BER).

Etapa 3: Preparação de dados

Após a execução das simulações, os *traces* gerados passam pela etapa de limpeza e transformação de dados. Assim, para facilitar a leitura dos resultados das simulações, as informações contidas nos *traces* de interesse são transferidas para arquivos no formato `txt`, criados ao final de cada simulação. Nesses arquivos estão contidas as informações para avaliação dos cenários de redes (taxa de entrega, vazão, atraso, *jitter*, etc). Em outras palavras, as informações que caracterizam o fluxo de transmissão dos pacotes na rede.

Após o processo de extração de dados, as informações não estruturadas são dispostas como vetores de atributos que compõem o *dataset* de treinamento. O *dataset* inicial é então composto por 60 fluxos de tráfego UDP. Cada fluxo UDP possui 24 atributos, sendo 23 deles atributos que implicam na avaliação de desempenho em redes e o último que representa o atributo alvo a ser estimado pelos algoritmos de aprendizado. Esse *dataset* inicial é submetido a uma seleção manual de atributos, na qual os atributos não importantes apontados por um especialista no assunto são retirados no intuito de diminuir a dimensionalidade dos dados. O *dataset* utilizado para treinamento e avaliação dos modelos preditivos é então composto de um conjunto de instâncias $X = \{dist, tamPacote, intPacote, taxaEntrega\}$. Cada exemplo é disposto como um vetor com 4 atributos, sendo 3 parâmetros de entrada (distância entre os nós $dist$, intervalo entre os pacotes $intPacote$ e tamanho do pacote $tamPacote$) e 1 parâmetro de saída (taxa de entrega $taxaEntrega$), representando o atributo de interesse para os algoritmos.

Um ponto importante na Etapa 3 é quando os nós da rede estão distantes um do outro ao ponto de não conseguirem se comunicar mais, sendo considerado um fluxo não iniciado. Neste caso, os pacotes não chegam ao destino. Esse fator causa o não

preenchimento das informações de fluxo, gerando *traces* vazios e, consequentemente, arquivos `txt` vazios. O tratamento de dados para os fluxos não iniciados é realizado da seguinte forma: uma vez identificado um fluxo não iniciado, um segundo arquivo `txt` é carregado com todas as informações dos resultados da simulação preenchidas com 0, de maneira a caracterizar este padrão em específico.

O escalonamento de dados é um passo importante por colocar todos os valores presentes no *dataset* em um mesmo padrão, diminuindo as chances de suposições errôneas dos algoritmos sobre os dados. Na padronização dos atributos preditores do *dataset* de simulação, usa-se a classe *StandardScaler* disponível na biblioteca *Scikit-learn* [Pedregosa et al. 2011]. Esse método subtrai cada valor de observação da média geral calculada de acordo com o *dataset* e divide pelo desvio padrão geral. O cálculo realizado segue a equação $z = (x - u)/s$, sendo x o valor da observação ou exemplo, u a média geral e s o desvio padrão, todos do *dataset* em estudo.

No intuito de contribuir para a interpretação dos resultados da simulação, a análise exploratória de dados pode ser útil por prover técnicas estatísticas e gráficas para explorar as informações geradas pelo simulador e auxiliar na visualização de padrões e comportamentos relevantes que possam estar presentes nos dados. Neste trabalho, é utilizada a técnica de Análise de Componentes Principais (*Principal Component Analysis* - PCA), um método que reduz a dimensionalidade dos dados de tal maneira que novas variáveis são ortogonais entre si, ou seja, são independentes ou não estão correlacionadas. Cada componente principal representa uma combinação linear das variáveis originais [Jolliffe 2011]. Neste trabalho, o PCA é aplicado para permitir a visualização da relação entre a taxa de entrega com os demais atributos. O *dataset* gerado, que possui 3 atributos independentes (distância entre os nós, intervalo entre pacotes e tamanho do pacote) e 1 atributo dependente (taxa de entrega) após a aplicação de PCA, recebe uma nova configuração nos valores das variáveis. Os 3 atributos independentes são transformados em um único componente principal (PC), representando o eixo x nos gráficos apresentados na Seção 4.

Etapas 4 e 5: Treinamento dos algoritmos e Avaliação dos modelos

Dando continuidade ao processo metodológico da Figura 1, a etapa de treinamento dos algoritmos de ML consiste em submeter o *dataset* representando o contexto investigado aos algoritmos, que aprendem o comportamento dos dados através dos padrões identificados. Este trabalho avalia um conjunto de algoritmos de ML com diferentes pressuposições indutivas sobre os dados para construir uma função de estimação da taxa de entrega a partir das características do tráfego simulado. O que se deseja é obter o estimador expresso na Equação 2, onde $dist$, $tamPacote$, $intPacote$ e $taxaEntrega$ correspondem à distância entre os nós, tamanho do pacote, intervalo entre os pacotes e taxa de entrega, respectivamente.

$$f(dist, tamPacote, intPacote) \rightarrow taxaEntrega \tag{2}$$

Cada algoritmo possui sua particularidade e, portanto, são configurados de forma diferente. Usa-se a técnica de otimização de hiperparâmetros denominada *Grid search*, que recebe uma lista de parâmetros de cada algoritmo e faz inúmeras combinações entre eles, retornando os melhores parâmetros de configuração para os algoritmos [Moro et al. 2019].

Os algoritmos de *Machine Learning* (ML) são baseados em diferentes conceitos matemáticos que influenciam na escolha de determinado algoritmo para resolver um problema. Cada algoritmo faz parte de um tipo de ML (supervisionado, não supervisionado ou por reforço). Então, dependendo das características do problema investigado, determinados algoritmos são mais adequados que outros. Neste trabalho, os algoritmos definidos para tratar da predição da taxa de entrega fazem parte do tipo supervisionado que necessita de um *dataset* rotulado com um atributo alvo. Assim, os algoritmos aprendem e fornecem suposições sobre o alvo. Especificamente, este trabalho usa a tarefa de regressão devido ao comportamento contínuo da taxa de entrega. A seguir, são apresentados os algoritmos de ML utilizados para a avaliação de desempenho [Faceli et al. 2011]:

- **Rede *Multilayer Perceptron* (MLP)**, uma abstração da rede neural biológica que lida com problemas não lineares e possui vários Perceptrons conectados. A arquitetura dessa rede neural funciona como uma rede *feedforward*. Ela é composta por uma camada de entrada, uma camada de saída que toma uma decisão sobre a entrada e, entre essas duas camadas, podem existir várias camadas ocultas que são o verdadeiro mecanismo computacional da MLP. Nas camadas ocultas, são utilizadas funções de ativação não lineares.
- ***Decision Tree* (DT)**, uma técnica de aprendizado supervisionado que tem comportamento recursivo usando a estratégia dividir para conquistar. Formalmente, uma árvore de decisão é um grafo acíclico direcionado em que cada nó ou é um nó de divisão, com dois ou mais sucessores, ou um nó folha. Ela funciona para ambas variáveis categóricas e contínuas de entrada e de saída. Dessa forma, pode ser facilmente utilizada em problemas de regressão.
- ***Support Vector Machine* (SVM)** para regressão, cujo aprendizado é encontrar um hiperplano com margem máxima representando a melhor reta que separa bem os dados minimizando o erro dos dados em relação ao hiperplano encontrado. A SVM resolve problemas de regressão usando o conceito de *Kernel Trick* que transforma o espaço não linear em linear, facilitando os demais cálculos necessários do algoritmo.
- ***Random Forest* (RF)**, que segue o mesmo princípio de uma DT e separa os atributos do dataset para formar um conjunto de árvores através da abordagem *Ensamble Learning* [Moro et al. 2019]. A partir dessa separação, o algoritmo consegue classificar um novo objeto. No caso da regressão, considera-se o valor da média das respostas de cada árvore.

Para a avaliação dos algoritmos, são utilizadas diferentes partições do *dataset* de simulações através de um processo de *k-folds* de validação cruzada, que consiste em dividir o conjunto de exemplos em *k* partições aproximadamente iguais. Para cada partição, estima-se o método sem presença de uma determinada parte e verifica-se o erro médio na partição não utilizada durante o treino [Faceli et al. 2011]. Seguindo essa abordagem, o *dataset* é dividido na proporção 80/20, ou seja, 80% da base para treinamento e 20% para teste a cada iteração da validação cruzada. Com as proporções dos *datasets* divididas, são executadas 30 execuções com o método de validação cruzada com $k = 10$ e, a cada execução, faz-se uma troca dos *datasets* de treinamento e teste com diferentes dados das diferentes partições. Ao final de cada rodada são armazenados os valores do coeficiente de determinação R^2 e *Root Mean Squared Error* (RMSE) dos algoritmos para que seja possível a escolha do melhor modelo de ML.

O R^2 fornece valores situados entre 0 e 1, que também podem ser interpretados em termos percentuais. Essa métrica de avaliação da qualidade de um modelo é calculada pela Equação 3. Sendo x e y os pontos de um dado conjunto em uma reta, usa-se $mx + b$ para calcular a inclinação dessa reta. O numerador da Equação 3 representa o erro quadrado da reta em relação ao x, enquanto o denominador representa o erro quadrado médio em relação a média do conjunto, expressada por $\widehat{y}$. Esse procedimento resulta no valor que explica o quanto da variação total não é descrita pela reta. Então, a Equação 3 resulta o quanto da variação total em y é correspondida pela variação em x [Viana et al. 2019]. Já a RMSE é definida como a raiz quadrada da distância quadrática média entre a observação real y_i e a previsão $\widehat{y}_i$ (Equação 4). A raiz quadrada é calculada para fazer a escala dos erros ser a mesma que a escala dos alvos, ou seja, tornar os valores de erros mais próximos dos possíveis valores reais que esses erros possam ser representados [Viana et al. 2019].

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - (mx_i + b))^2}{\sum_{i=1}^{n}(y_i - \widehat{y})^2} \quad (3) \qquad RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2} \quad (4)$$

Salienta-se que um fator importante para um bom desempenho dos algoritmos é obter um baixo erro de predição dado pela um R^2 e RMSE entre 0 e 1. Quanto mais próximo de 1, melhor a generalização do modelo de ML sob novos dados.

Etapa 6: Comparação das simulações com os modelos

A Etapa 6 mostra os resultados comparativos entre as simulações realizadas na Etapa 2 e os modelos gerados pelas Etapas 4 e 5, ou seja, a comparação das simulações com as predições fornecidas pelos modelos de *Machine Learning*. Os resultados desta etapa são detalhados na próxima seção.

4. Resultados

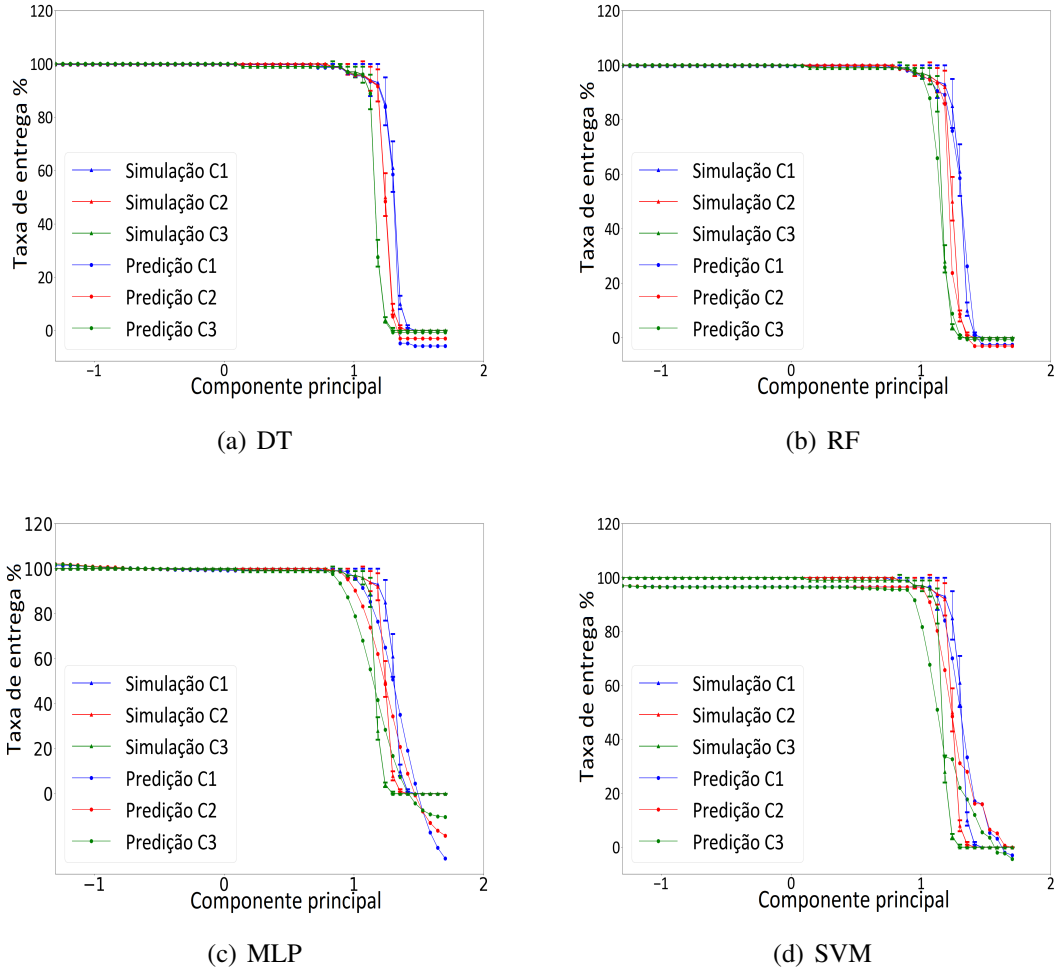
Nesta seção, são apresentados os principais resultados da avaliação de desempenho dos algoritmos de ML (MLP, DT, SVM e RF) em relação aos resultados das simulações realizadas no NS-3. Para facilitar a compreensão dos resultados, são definidos 6 cenários distintos conforme mostra a Tabela 2. Para cada cenário, o parâmetro *distância entre nós da rede* assume valores inteiros linearmente espaçados no intervalo $[50,\ 110]$, de forma que há um total de 60 combinações de valores de parâmetro por cenário. Desta maneira, uma vez estabelecido os fatores de variação para cada um dos três parâmetros de simulação considerados, possibilita-se a observação e detecção de possíveis diferenças significantes entre os parâmetros e a taxa de entrega obtida.

Tabela 2. Cenários de avaliação.

Parâmetro	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Intervalo entre pacotes (s)	0.01	0.01	0.01	0.1	0.1	0.1
Tamanho dos pacotes (B)	256	512	1024	256	512	1024

As Figuras 4 e 5 mostram as predições dos algoritmos após o a fase de treinamento dos algoritmos. Tais predições são comparadas com os valores das simulações e observa-se o quão próximo as predições estão dos resultados das simulações. É possível observar

também que as curvas geradas pelas predições dos algoritmos estão de acordo com o intervalo de confiança dado pelos valores das simulações. As Tabelas 3, 5, 4 e 6 mostram os resultados das métricas de avaliação RMSE (que também pode ser representada como a taxa de acerto dos algoritmos) e R^2 de cada algoritmo considerando os cenários definidos na Tabela 2. A seguir, os resultados de cada cenário são discutidos.



(a) DT (b) RF

(c) MLP (d) SVM

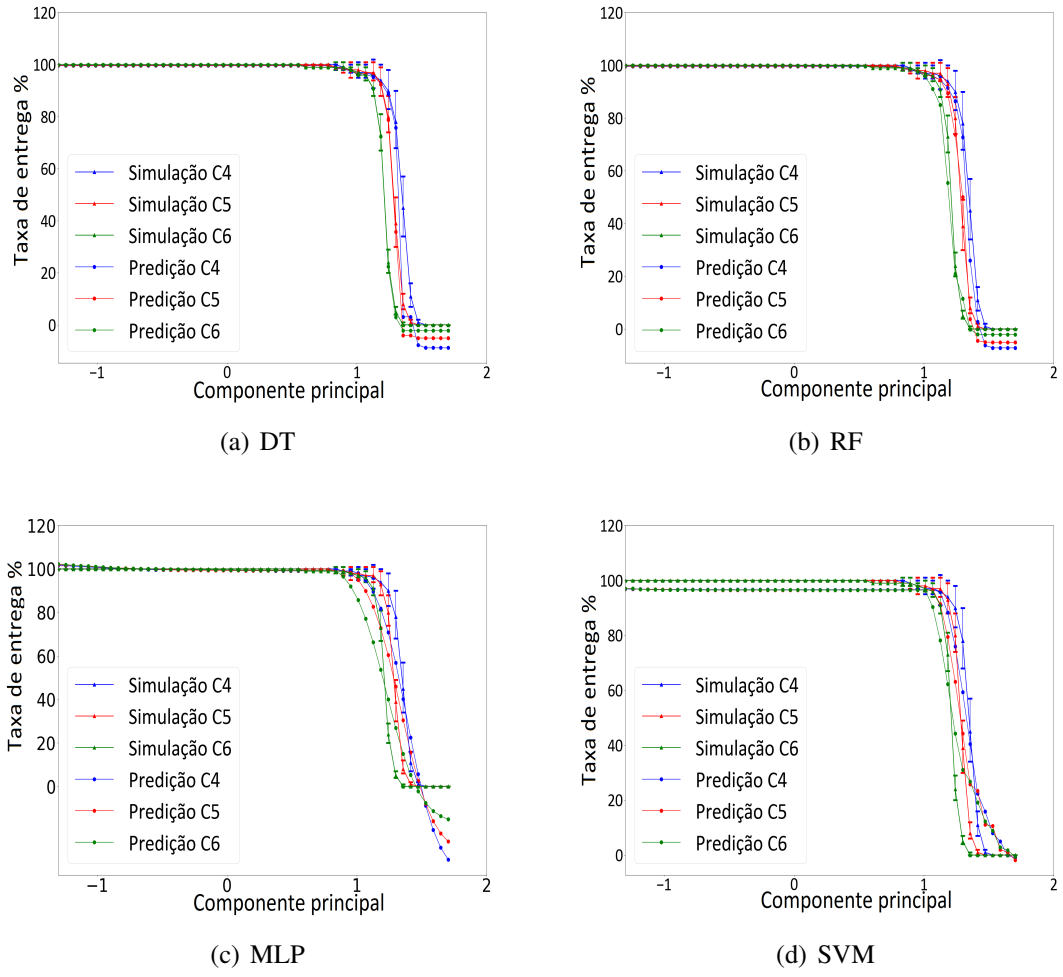
Figura 4. Comparações entre os valores das simulações e as predições dos algoritmos para o intervalo entre pacotes de 0.01s (cenários 1, 2 e 3).

Cenário 1: a MLP possui 93% de taxa de acerto ou R^2 nas suas predições e 0.24 de erro estimado pela RMSE em relação à reta real formada com os dados do *dataset*. A DT possui resultados significantes para esse cenário, tendo 99% de taxa de acerto e um erro de 0.08, consideravelmente um valor baixo de RMSE. O RF também consegue uma taxa de acerto de 99% e um erro de 0.07. Também temos a SVM com 96% de acerto e 0.18 de erro nas predições. Assim, observa-se que a DT e o RF têm os melhores valores para as métricas avaliadas, enquanto a MLP e a SVM possuem os piores resultados.

Cenário 2: a MLP consegue uma taxa de acerto de 94% e 0.23 de erro estimado pela RMSE, resultados bem próximos do cenário anterior. Assim como no cenário anterior, a DT também consegue uma taxa de acerto de 99% e 0.03 de erro. O RF obtém 99%

de acerto nas predições e erro de 0.09. Já a SVM tem 95% de taxa de acerto e um erro 0.22 neste cenário. A DT e o RF também possuem os melhores resultados neste cenário.

Cenário 3: a MLP possui o 94% de acerto e um valor de 0.22. Já a SVM tem uma taxa de acerto de 93% e um erro de 0.25. A DT e RF possuem 99% de acerto nas suas predições e erros de 0.01 e 0.09, respectivamente. Novamente, os algoritmos baseados na estrutura de árvore, DT e RF, têm os melhores resultados, sendo a DT o melhor algoritmo para predizer novos casos para este tipo de cenário.



(a) DT (b) RF

(c) MLP (d) SVM

Figura 5. Comparações entre os valores das simulações e as predições dos algoritmos para o intervalo entre pacotes de 0.1s (cenários 4, 5 e 6).

Cenário 4: para este cenário, a MLP obtém 93% de acerto nas predições e 0.26 de erro. A DT possui 95% de acerto e 0.20 de erro nas suas predições. Já a SVM possui o seu melhor resultado de todos os cenários, tendo 97% de acerto nas predições e 0.17 de erro. O RF tem 98% de acerto e um erro de 0.11, sendo este algoritmo o melhor no cenário 4. Diferentemente dos cenários anteriores, a DT não tem os melhores valores em termos de R^2 e RMSE.

Cenário 5: A MLP possui 94% de acerto e de 0.23 de erro estimado pela RMSE. A SVM obtém 96% de acerto nas predições e 0.18 de erro. Os algoritmos DT e RF

possuem 99% de acerto e erros de 0.07 e 0.06, respectivamente. Assim, nota-se que a DT e o RF são os melhores algoritmos também neste cenário.

Cenário 6: já neste cenário, a MLP consegue seu melhor resultado em comparação com os outros cenários, ela obtém 95% de acerto nas predições e 0.21 de erro. Porém, não é o algoritmo mais adequado para o cenário. A SVM também obtém 95% de acerto e 0.21 de erro. A DT e o RF aparecem como os melhores algoritmos, os dois possuem 99% de acerto e erros de 0.02 e 0.06, respectivamente.

Nota-se que os dois algoritmos baseados na estrutura de árvore decisão, DT e RF, possuem os melhores resultados em todos os cenários. Esses dois algoritmos aprendem o comportamento da taxa de entrega com maior precisão e com isso, conseguem resultados mais próximos dos reais. Além da avaliação das métricas R^2 e RMSE, vale ressaltar dois fatores que influenciam diretamente nos resultados. O primeiro é o tempo de processamento realizado pelos algoritmos na busca pelos padrões nos dados e o segundo refere-se as combinações realizadas pela técnica de hiperparâmetro *Grid search* [Moro et al. 2019].

Os algoritmos buscam os padrões nos dados seguindo um viés indutivo que afeta diretamente o tempo de processamento. Durante os experimentos, observa-se que os dois algoritmos baseados na estrutura de árvore (DT e RF) têm os menores custos de processamento na busca pelos padrões. Já a SVM pode ser prejudicada durante o treinamento por necessidade de realizar cálculos mais complexos na busca pelo hiperplano que melhor representa os dados. Essa necessidade da SVM faz com que seu processamento seja maior que os demais algoritmos utilizados neste trabalho. A MLP também possui um tempo de processamento maior que DT e RF, mas não ultrapassando o tempo da SVM. Com o uso do *Grid search* os algoritmos têm um processamento ainda maior, uma vez que a técnica de hiperparâmetro faz inúmeras combinações dos parâmetros de cada algoritmo.

Tabela 3. Valores para R^2

Algoritmo	C1	C2	C3
MLP	93%	94%	94%
DT	**99%**	**99%**	**99%**
SVM	96%	95%	93%
RF	**99%**	**99%**	**99%**

Tabela 4. Valores para RMSE

Algoritmo	C1	C2	C3
MLP	0.24	0.23	0.22
DT	0.08	**0.03**	**0.01**
SVM	0.18	0.22	0.25
RF	**0.07**	0.09	0.09

Tabela 5. Valores para R^2

Algoritmo	C4	C5	C6
MLP	93%	94%	95%
DT	95%	**99%**	**99%**
SVM	97%	96%	95%
RF	**98%**	**99%**	**99%**

Tabela 6. Valores para RMSE

Algoritmo	C4	C5	C6
MLP	0.26	0.23	0.21
DT	0.20	0.07	**0.02**
SVM	0.17	0.18	0.21
RF	**0.11**	**0.06**	0.06

Conforme os resultados observados nas Figuras 4 e 5 e Tabelas 3, 4, 5 e 6 apresentados, os algoritmos utilizados neste trabalho são capazes de auxiliar nas predições das simulações e fornecer resultados precisos a partir de um conjunto de discriminantes estatísticos obtidos para cada cenário de rede considerado. Como observado nas Tabelas, os modelos de *Decision Tree* e *Random Forest* possuem os melhores resultados e conseguem

predizer o comportamento da taxa de entrega com maior precisão. Pode-se observar que o *Random Forest* obteve desempenho igual ou superior à *Decision Tree* nos cenários avaliados, provavelmente pela característica da técnica de *Ensemble Learning* em combinar múltiplas árvores de decisão no intuito de reduzir a variância e *overfitting*. Além disso, para cada cenário, pode-se utilizar o modelo preditivo mais apropriado que combine os melhores valores para ambos os critérios.

5. Conclusões

Este trabalho apresenta uma avaliação de desempenho dos algoritmos de *Machine Learning* na otimização de simulações de redes de computadores a fim de reduzir o tempo e os recursos computacionais das simulações. O trabalho tem como estudo de caso uma Rede em Malha Sem Fio e a taxa de entrega como medida de desempenho de redes de computadores. A *Decision Tree* e *Random Forest* possuem resultados mais concretos e próximos dos valores das simulações com 99% de taxa de acerto (representada pelos valores de R^2) na maioria dos cenários e baixos erros de predições expressados pela RMSE. A *Support Vector Machine* e a Rede *Multilayer Perceptron* também possuem resultados significantes. Os resultados obtidos são satisfatórios em relação ao desempenho observado nas predições e mostram a capacidade dos algoritmos de auxiliar nas simulações de redes de computadores. Com uma série de novos experimentos e outros cenários de redes, os resultados podem melhorar em termos de precisão.

Em relação às perspectivas de trabalhos futuros, destaca-se a criação de novos *datasets* baseados em outros cenários de redes com maior complexidade, aplicação de técnicas de seleção de atributos, utilização de *traces* de tráfego reais, realização de novos experimentos de *Machine Learning*, estudo de caso usando outras medidas de desempenho de redes e o desenvolvimento de uma prova de conceito.

Referências

Abreu, C. E. M., Gonzaga, D. R. B., dos Santos, F. J., de Oliveira, J. F., de Morais Oliveira, K. D., Figueiredo, L. M., Nascimento, M. P., de Oliveira, P. G., de Souza Yoshinaga, S. T., de Oliveira, T. T., da Mata, V. S., and dos Santos Gonçalves, G. A. (2017). Indústria 4.0: Como as empresas estão utilizando a simulação para se preparar para o futuro. *Revista de Ciências Exatas e Tecnologia*.

Akyildiz, I. F., Wang, X., and Wang, W. (2005). Wireless mesh networks: a survey. *Computer Networks - Science Direct*, 47(4):445 – 487.

Andreev, K., Boyko, P., and Andreev, Kirill, P. B. (2010). IEEE 802.11s Mesh Networking NS-3 Model. In *NS-3 Workshop*.

Faceli, K., Lorena, A. C., Gama, J., and de Carvalho, A. C. P. L. F. (2011). *Inteligência artificial: uma abordagem de aprendizado de máquina*. Grupo Gen - LTC.

Hwang, A., Lee, J., and Kim, B. (2017). Design and performance evaluation of tcp performance enhancement algorithm with machine learning in wireless environments. *International Journal of Applied Engineering Research*, 12:14370–14376.

IEEE 802.11s (2011). IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks

- Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 10: Mesh Networking, IEEE Std.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley professional computing. Wiley.

Jolliffe, I. (2011). *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kajita, S., Yamaguchi, H., Higashino, T., Urayama, H., Yamada, M., and Takai, M. (2015). Throughput and delay estimator for 2.4 ghz wifi aps: A machine learning-based approach. In *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 223–226. IEEE.

Lee, K. Y., Suh, Y.-K., and Cho, K. W. (2017). Development of a simulation result management and prediction system using machine learning techniques. *International Journal of Data Mining and Bioinformatics*.

Moro, F., Amaral, A., Amaral, A., and Nogueira, R. (2019). Análise do impacto da agregação dos fluxos ip nos algoritmos de aprendizado de máquina supervisionado voltados para a detecção de intrusão. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 946–957, Porto Alegre, RS, Brasil. SBC.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Riley, G. F. and Henderson, T. R. (2010). The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer.

Saggioro, L. F. Z., Gonzaga, F. B., and Ribeiro, R. R. (2012). Tracemetrics - uma ferramenta para a obtenção de medidas de interesse no ns-3. *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Saha, B. (2018). Green computing: Current research trends. *International Journal of Computer Sciences and Engineering*, 6.

Santos, B. P., Silva, L. A. M., Celes, C. S. F. S., Neto, J. B. B., Peres, B. S., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. A. F. (2016). Internet das coisas: da teoria à prática. *Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)*.

Silva, C., Oliveira, Y., Celes, C., Braga, R. B., and de Oliveira, C. T. (2018). Performance evaluation of wireless mesh networks in smart cities scenarios. In *Proceedings of the Euro American Conference on Telematics and Information Systems, EATIS 2018, Fortaleza, Brazil, November 12-15, 2018.*, pages 7:1–7:7.

Viana, J. D. F., Braga, O., Silva, L., and Neto, F. M. (2019). Analyzing patterns of a bicycle sharing system for generating rental flow predictive models. In *Anais do III Workshop de Computação Urbana*, pages 57–70, Porto Alegre, RS, Brasil. SBC.