UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Uso de Teoria dos Jogos como política de alocação de recursos em nuvens veiculares

Henrique Andrews Prado Marques

São Carlos – SP

Uso de Teoria dos Jogos como política de alocação de recursos em nuvens veiculares

Henrique Andrews Prado Marques

Orientador: **Prof. Dr. Rodolfo Ipolito Meneguette**

Monografia final de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.
Área de Concentração: Redes de Computadores, Sistemas Distribuídos, Computação em Nuvem/Névoa.

USP – São Carlos
Junho de 2021

Marques, Henrique Andrews Prado
Uso de Teoria dos Jogos como política de alocação de recursos em nuvens veiculares / Henrique Andrews Prado Marques. - São Carlos - SP, 2021.
55 p.; 29,7 cm.

Orientador: Rodolfo Ipolito Meneguette.
Monografia (Graduação) - Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos - SP, 2021.

1. Alocação de Recursos. 2. Nuvens Veiculares. 3. Teoria dos Jogos. I. Meneguette, Rodolfo Ipolito. II. Instituto de Ciências Matemáticas e de Computação (ICMC/USP). III. Título.

Este trabalho é dedicado a todos os professores,
cujo papel na busca pelo conhecimento é indispensável.

AGRADECIMENTOS

O desenvolvimentos deste trabalho contou com a ajuda, direta ou indireta, de diversas pessoas, dentre as quais agradeço:

Ao meu professor orientador, que durante o semestre me acompanhou pontualmente, dando todo auxílio necessário para a elaboração do projeto.

Aos professores do curso de Engenharia de Computação, que através dos seus ensinamentos permitiram que eu pudesse hoje estar concluindo este trabalho.

Aos meus pais, que me incentivaram a cada momento, não permitindo que eu desistisse.

“A menos que modifiquemos a nossa maneira de pensar,
não seremos capazes de resolver os problemas causados
pela forma como nos acostumamos a ver o mundo.”
(Alber Einstein)

RESUMO

MARQUES, H. A. P.. **Uso de Teoria dos Jogos como política de alocação de recursos em nuvens veiculares** . 2021. 55 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

A cada dia que passa, mais veículos com poder computacional e capacidade de conexão com as redes são inseridos nas estradas, possibilitando e tornando cada vez mais interessante a implementação de sistemas de transporte inteligentes (ITS) em áreas urbanas ou rodovias. Os ITS buscam resolver ou pelo menos mitigar alguns dos problemas enfrentados pelo setor rodoviário através do fornecimento de serviços computacionais. Para auxiliar na provisão desses serviços, a criação de nuvens veiculares surgiu como um meio de aproximar os recursos computacionais e serviços das aplicações veiculares. No entanto, na maior parte das vezes os recursos computacionais presentes nos veículos que compõem as nuvens veiculares são subutilizados. Sendo assim, para melhor aproveitar estes recursos este trabalho propõe uma política de alocação de recursos baseada em Teoria dos Jogos para maximizar a utilização dos mesmos e posteriormente comparar esta abordagem com outras conhecidas na literatura e averiguar seu desempenho com relação a outras métricas de avaliação além da utilização de recursos.

Palavras-chave: Alocação de Recursos, Nuvens Veiculares, Teoria dos Jogos.

ABSTRACT

MARQUES, H. A. P.. **Uso de Teoria dos Jogos como política de alocação de recursos em nuvens veiculares** . 2021. 55 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

With each passing day, more vehicles with computational power and capacity to connect to networks are inserted on the roads, enabling and making the implementation of intelligent transport systems (ITS) in urban areas or highways increasingly interesting. ITS seeks to solve or at least mitigate some of the problems faced by the road sector through the provision of computer services. To assist in the provision of these services, the creation of vehicular clouds emerged as a way to bring computational resources and services closer to vehicular applications. However, most of the times the computational resources present in the vehicles that make up the vehicle clouds are underused. Therefore, in order to make the best use of these resources, this work proposes a resource allocation policy based on Game Theory to maximize their use and then compare this approach with others known in the literature and verify its performance in relation to other evaluation metrics beyond the use of resources.

Key-words: Resource Allocation, Vehicle Clouds, Game Theory.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Motivação e Contextualização

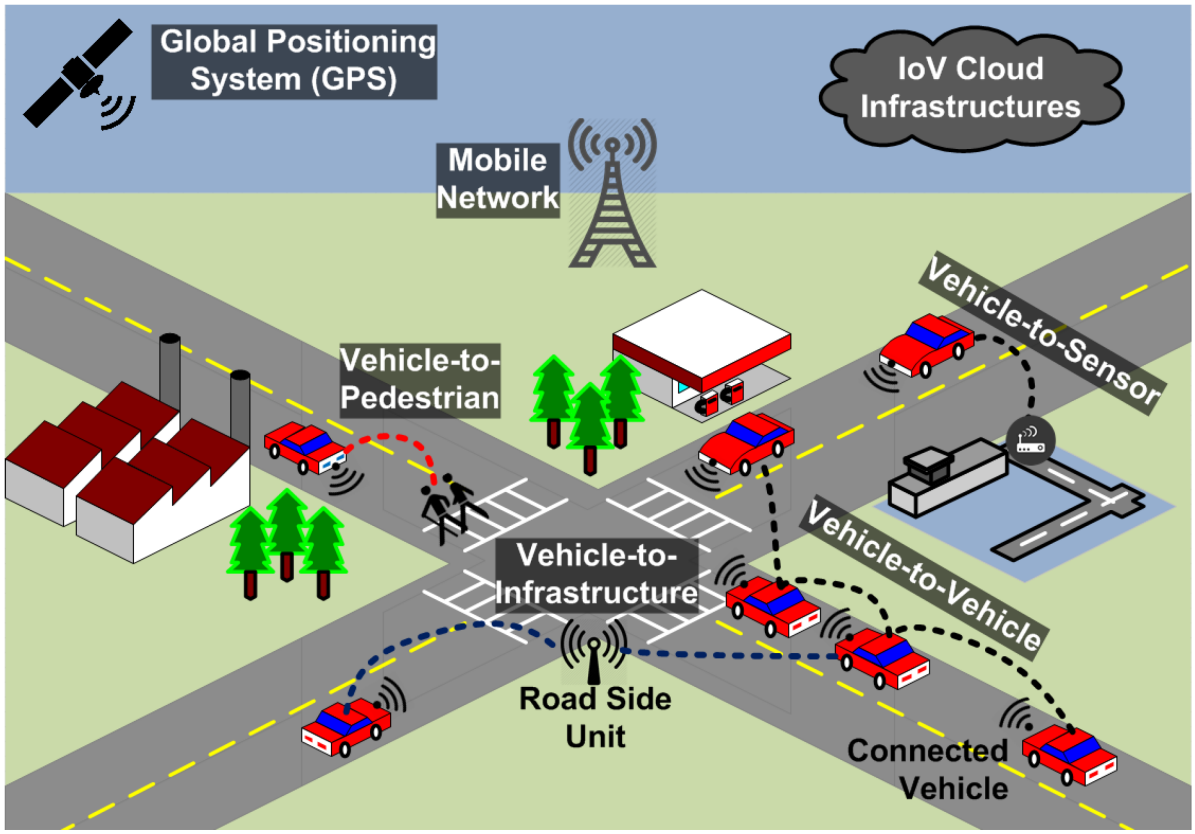
A cada ano que passa, há um aumento significativo no número de veículos em todo mundo. Consequentemente, a quantidade de veículos conectados que circulam nas ruas entre nós também cresce, compartilhando mais dados do que nunca. Estima-se que até 2025 possa haver quase 2 bilhões de veículos conectados nas estradas ao redor do mundo (CISCO, 2020). Acompanhado desse crescimento, a indústria automobilística em parceria com indústrias de tecnologia vem ao longo dos anos investindo consideravelmente na adição de mais e melhores recursos tecnológicos nos veículos, tais como sensores e *chipsets* específicos para o cenário, de forma que se tenha um maior poder de processamento, comunicação e variedade de casos de uso (Qualcomm, 2020), contribuindo assim para uma ampliação e criação de novas frentes de pesquisa.

Especificando um pouco mais o contexto acima, adentra-se ao conceito de Sistemas de Transporte Inteligente, terminologia originária do inglês *Intelligent Transport Systems* (ITS). São sistemas os quais visam fornecer serviços inovadores relacionados à gestão e eficiência de tráfego, permitindo que os usuários estejam mais bem informados e façam um uso mais seguro, coordenado e inteligente das redes de transporte, consequentemente reduzindo ou até evitando alguns dos problemas mais recorrentes e conhecidos do cotidiano de grandes centros urbanos como congestionamentos, acidentes, alta emissão de gás carbônico, tempo em trânsito, entre outros (Meneguette; De Grande; Loureiro, 2018, p. 1-21).

Nos ITS, os veículos são equipados com sensores e dispositivos de comunição para que seja possível essa conectividade entre eles, permitindo assim o compartilhamento de recursos, serviços e troca de informações de forma a resolver os desafios relacionados à mobilidade urbana citados anteriormente (Meneguette; De Grande; Loureiro, 2018, p.1-21). A comunicação realizada entre os veículos e outros dispositivos é feita por meio de uma Rede Ad Hoc Veicular, ou como será referenciada posteriormente *Vehicular Ad Hoc Network* (VANET) (Meneguette; De Grande; Loureiro, 2018, p. 23-36). Nas VANETs, a comunicação pode ser efetuada de veículo para veículo *Vehicle-to-Vehicle* (V2V), veículo para infraestrutura *Vehicle-to-Infrastructure* (V2I) (AL-SULTAN *et al.*, 2014; Meneguette; De Grande; Loureiro, 2018, p. 53-112), ou até mesmo de veículo para pedestre *Vehicle-to-Pedestrian* (V2P) em alguns casos (Qualcomm, 2020). Na figura 1, ilustra-se um exemplo dos possíveis casos de comunicação entre elementos de uma

VANET.

Figura 1 – Ilustração representativa das possíveis conexões de uma VANET

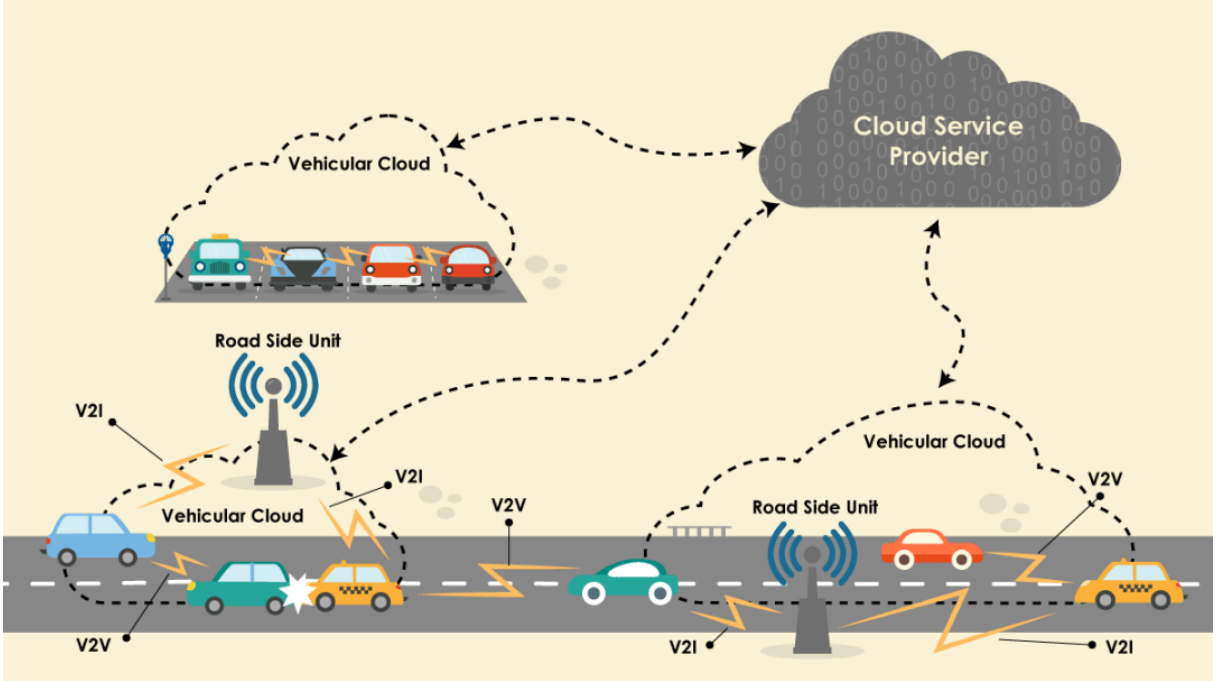


Fonte: https://www.mdpi.com/1424-8220/16/6/879

E para fazer com que os ITS tenham uma performance mais robusta e responsiva, a computação em nuvem veicular almeja a agregação dos recursos computacionais presentes nos veículos conectados a fim de tornar possível a execução de aplicações diversas através da criação de nuvens veiculares, do inglês *Vehicular Clouds* (VCs) (THAKUR; MALEKIAN, 2019). As VCs visam uma cooperação eficiente na comunicação, alocação de tarefas e compartilhamento de recursos nas VANETs, e elas têm um impacto notável em diversas aplicações de ITS por utilizar rapidamente estes recursos, tais como processamento, armazenamento e comunicação para tomadas de decisão sem necessitar da computação em nuvem tradicional para tal (BRIK *et al.*, 2019). Na figura 2, é possível observar exemplos de formações de VCs na VANET.

Ao agregar o excesso de recursos que estão na borda da rede, cria-se um conjunto de serviços que são disponibilizados para os veículos, podendo ser utilizados em ambientes estáticos e dinâmicos (HAGENAUER *et al.*, 2019). Sendo assim, os veículos se tornam componentes chave no contexto de aplicações de ITS, já que passam a atuar, além da gestão de tráfego, como fonte para coleta e processamento de dados em tempo real (HATTAB *et al.*, 2019). Portanto, mecanismos que façam uso dos recursos das VCs de maneira eficiente são imprescindíveis, de forma que estes serviços sejam usados da melhor maneira possível, fornecendo pode computacional

Figura 2 – VANET com nuvens veiculares



Fonte: http://mahbubulalam.com/convergence-of-secure-vehicular-ad-hoc-network-and-cloud-in-iot/

mais próximo das aplicações veiculares e com qualidade para quem os consome.

Os principais desafios que envolvem a proposição de abordagens para uma eficiente alocação de recursos e tarefas em VCs giram em torno da natureza dinâmica das VANETs (YANG *et al.*, 2019; COUTINHO; BOUKERCHE, 2019). A alocação e processamento de tarefas com requisitos intolerantes ao atraso, como detecção de acidentes, ou tarefas que exigem alto poder computacional, como processamento de imagens de trânsito e outras aplicações de multimídia de modo geral, acaba sendo bastante dasafiador por conta da alta mobilidade dos veículos (MENEGUETTE *et al.*, 2019; SORKHOH *et al.*, 2019). Desta forma, assegurar o atendimento das tarefas independente de seus requisitos ou quaisquer outras características é imprescindível. Por isso, é de grande interesse o estudo e elaboração de mecanismos que atendam o maior número de tarefas possíveis e que façam uso ao máximo dos recursos ociosos a fim de prover serviços com poder computacional suficiente o mais próximo das aplicações de ITS (COSTA *et al.*, 2020).

1.2 Objetivos

Este trabalho visa a proposição de uma política de alocação de recursos e atendimento de tarefas baseada em Teoria dos Jogos, onde o foco principal se encontra na maximização do uso dos recursos computacionais, e posteriormente comparar esta abordagem a outras já conhecidas na literatura, utilizando algumas métricas como ganho/valor agregado pelas tarefas alocadas, percentual de tarefas atendidas e percentual de recursos utilizados.

1.3 Organização

No capítulo 2 é apresentada uma revisão de conceitos e técnicas relevantes para o desenvolvimento do projeto, bem como trabalhos da literatura relacionados ao presente projeto.

A seguir, no Capítulo 3, é descrito com mais detalhes o cenário do problema em questão juntamente com os objetivos do projeto, as atividades realizadas para chegar na solução proposta, o processo de obtenção dos resultados juntamente com uma análise dos mesmos e por fim as principais dificuldades e limitações encontradas durante a condução do trabalho.

Finalmente, no capítulo 4, são apresentadas as principais contribuições do trabalho realizado, tanto no âmbito profissional quanto pessoal, as conclusões a respeito do trabalho desenvolvido e experiências adquiridas. Também são feitas considerações gerais a respeito de como Curso de Graduação agregou para o desenvolvimento do projeto, e por último, sugestões de trabalhos futuros para a continuação deste projeto.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

2.1 Considerações Iniciais

Neste capítulo serão apresentados alguns conceitos e técnicas relevantes para o desenvolvimento do trabalho, assim como trabalhos da literatura que tratam do problema da alocação de recursos/tarefas e possuem relação com este projeto.

2.2 Conceitos e Técnicas Relevantes

2.2.1 Teoria dos Jogos

Teoria dos Jogos é uma teoria de matemática aplicada que se tornou mais conhecida e consolidada como um campo de pesquisa interdisciplinar após a publicação em 1944 da obra *Theory of Games and Economic Behaviour* do famoso matemático John von Neumann e economista Oskar Morgenstern. Apesar de inicialmente sua aplicação ter se voltado exclusivamente para o ramo da economia, atualmente é utilizada em diversos outros campos, inclusive na computação (ROSS, 2019).

Ela visa o estudo do comportamento/tomada de decisão dos participantes, no caso os jogadores, onde os resultados de suas ações estão interligados. Assim, um jogo constitui um cenário com interações estratégicas entre os participantes, onde a decisão de um jogador influencia nas ações dos demais. Os jogadores são considerados entes racionais que possuem preferências claras, analisam o cenário do jogo e buscam sempre executar a ação com resultado esperado ótimo para si mesmo, maximizando seus ganhos ou minimizando suas perdas (ROSS, 2019).

Os jogos por definição consistem em um grupo de jogadores, ações a serem realizadas por cada um deles e os ganhos resultantes referentes às estratégias adotadas. Há duas formas possíveis de representação de jogo, porém neste trabalho será evidenciada apenas uma, no caso a que foi utilizada (ROSS, 2019).

Forma Normal: É uma matriz de recompensas onde os elementos correspondem aos ganhos dos jogadores para cada combinação de estratégias dos mesmos. A figura 3 ilustra uma matriz de recompensas, a qual representa um jogo com 2 jogadores em que o jogador 1 pode escolher entre as ações *para cima* e *para baixo* e o jogador 2 pode escolher entre as ações

esquerda e *direita*. Os ganhos de cada combinação de estratégias estão representados pelos números nos campos da matriz, onde em cada campo o número azul (à esquerda) representa o ganho do jogador 1 e o número vermelho (à direita) o ganho do jogador 2.

Considera-se no jogo representado na forma normal que os jogadores atuam ao mesmo tempo, sem saber qual será a decisão do(s) outro(s).

Figura 3 – Exemplo de Matriz de Recompensas

	Jogador 2 escolhe esquerda	***Jogador 2 escolhe direita***
Jogador 1 escolhe para cima	4, 3	-1, -1
Jogador 1 escolhe para baixo	0, 0	3, 4

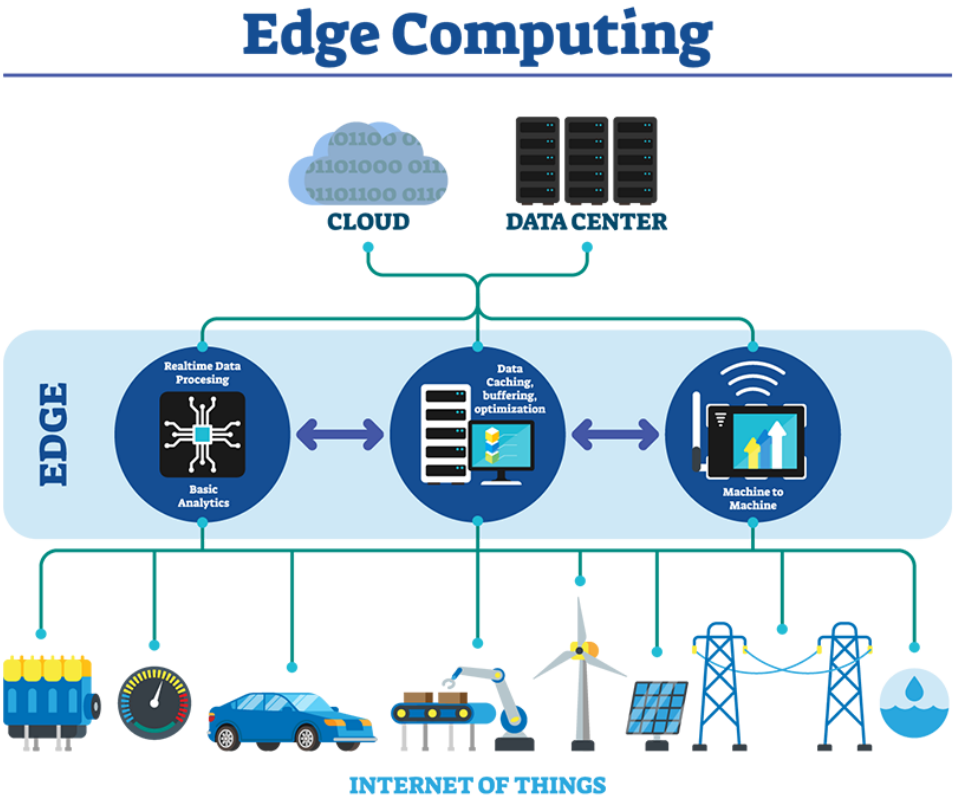
Fonte: https://pt.wikipedia.org/wiki/Teoria_dos_jogos

Outro conceito importante dentro da Teoria de Jogos é o Equilíbrio de Nash, conceito este desenvolvido pelo matemático John Nash em 1950. Configura um cenário de jogo em que há dois ou mais jogadores envolvidos, onde nenhum deles consegue aumentar seu ganho ao tentar mudar sua estratégia atual unilateralmente. Mesmo não havendo cooperação entre os participantes, é possível que a busca individual do melhor retorno acabe levando o jogo à um resultado em que se observe estabilidade, não havendo então incentivo para que nenhum dos jogadores altere seu comportamento. O equilíbrio de Nash é a forma mais comum de se definir a solução de um jogo não-cooperativo envolvendo dois ou mais participantes (ROSS, 2019).

2.2.2 Edge Computing (Computação de borda)

É um paradigma de computação distribuída que aproxima a computação e o armazenamento de dados do local onde são mais necessários para melhorar os tempos de resposta e economizar largura de banda. Com processamento mais próximo, os usuários se beneficiam de serviços mais rápidos e confiáveis. Um grande ponto positivo da computação de borda inclui a capacidade de fazer grandes análises e agregações de dados localmente, possibilitando a tomada de decisões quase que imediata (Red Hat, 2021). Vários trabalhos atuais envolvendo ITS e VANETs fazem uso desta topologia quando se trata de nuvens/névoas veiculares. Na figura 4 é possível observar alguns casos de uso da computação de borda.

Figura 4 – Casos de uso da computação de borda



Fonte: https://innovationatwork.ieee.org/real-life-edge-computing-use-cases/

2.2.3 Road-Side Unit

Road-Side Unit (RSU) é um termo que se refere à infraestrutura que se encontra no acostamento ou próximo às vias. São dispositivos especiais de comunicação *wireless* que fornecem conectividade, serviços e informações de tráfego aos veículos os quais a estrutura está conectada (Meneguette; De Grande; Loureiro, 2018, p.53-77).

2.3 Trabalhos Relacionados

Existem trabalhos na literatura que abordam o problema de alocação de tarefas e recursos em VANETs/VCs e alguns deles são referenciados nesta seção.

É apresentada em (NABI *et al.*, 2017) uma abordagem de alocação de tarefas em VCs que faz uso de conceitos do Problema da Mochila. Os autores proveem um esquema que soluciona de maneira aproximada a alocação de apenas uma tarefa em tempo polinomial, além de proporem uma solução gulosa com a mesma finalidade. Também estendem o algoritmo para que se resolva o problema de forma aproximada para o caso de n tarefas. Todavia, o cénario considera um ambiente *offline* e os algoritmos necessitam das informações das tarefas que serão alocadas previamente, não tendo então um desempenho satisfatório em situações de tomada de decisão em tempo real.

É proposto em (WANG *et al.*, 2018) um algoritmo para alocação de tarefas utilizando como base conceitos do Problema das Múltiplas Mochilas. É contextualizado neste trabalho o problema de Descarregamento de Computação, que significa transferência de processamento computacional para uma plataforma externa. Cada usuário pagará pelas tarefas descarregadas baseado em seus tamanhos, então o objetivo se torna maximizar o lucro total do descarregamento de computação da perspectiva da infraestrutura. Para obter a solução ideal, um algoritmo *Branch-and-Bound* foi proposto, e para obter um desempenho aproximado com custo computacional menor optou-se por um método heurístico Guloso. Entretanto, os algoritmos se mostraram computacionalmente custosos por em uma etapa encontrar a solução ótima para cada mochila e apenas depois fazer uso do resultado para realizar a alocação das tarefas nas diferentes VCs.

Apresenta-se em (BRIK *et al.*, 2018) uma abordagem para gerenciar o fornecimento de serviços em nuvens veiculares baseada em teoria dos jogos. O trabalho visa uma forma de como selecionar o melhor provedor de serviços que se adeque aos requisitos de qualidade de serviço e preço exigidos pelos consumidores. Paralelamente, os provedores devem ajustar os recursos dos serviços fornecidos e os preços de acordo com certas condições, tal como a taxa de requisição dos consumidores. É levado em consideração o benefício de cada jogador (os veículos consumidores de serviços e os provedores) e permite que os veículos consumidores encontrem o veículo provedor mais adequado baseado na probabilidade de interação entre eles. Os resultados do trabalho são obtidos através de uma simulação em um modelo de mobilidade urbana e mostram que a abordagem proposta é um esquema de seleção de serviços consideravelmente eficiente e confiável ao atingir uma qualidade de serviço elevada.

Em (PEREIRA *et al.*, 2019) é proposta uma política de alocação de recursos em VCs em ambiente rodoviário. A política proposta tem por objetivo maximizar a disponibilidade de recursos na VC. Veículos conectados devem cooperar para a criação de um reservatório de recursos, para dessa forma se ter conhecimento dos recursos disponíveis, os quais serão fornecidos pelos veículos e o conjunto de nevoeiros onde mais recursos serão gerados e serviços alocados. O paradigma de computação em névoa permite que, além da proximidade dos recursos aos veículos, os recursos da névoa sejam agregados aos outros recursos, aumentando a quantidade total de recursos e consequentemente a quantidade de serviços que podem ser oferecidos. Contudo, além de ser um cenário mais previsível que um cenário urbano, entra o fato de ter recursos além dos já oferecidos pelos veículos, o que torna menos complexo para a alocação dos serviços.

(COSTA *et al.*, 2020) propõem um mecanismo para alocação de tarefas em VCs baseado em otimização combinatória e o comparam com outras 3 abordagens. Os resultados obtidos mostram tanto um maior uso dos recursos disponvíes quanto um maior número de tarefas alocadas e maior ganho em comparação com os outros métodos. É simulado um cenário urbano em tempo real onde as tarefas são geradas ao longo da execução e são alocadas nas VCs que são recomputadas a cada minuto devido à mobilidade dos veículos. Para geração das VCs e

agregação dos recursos, é utilizada uma técnica de clusterização bastante conhecida na literatura, o *Density Based Spatial Clustering of Application with Noise* (DBSCAN) (ESTER *et al.*, 1996).

Em (LIEIRA *et al.*, 2020) propõem-se um algoritmo que adota a técnica meta-heurística conhecida como *Grey Wolf Optimization* (MIRJALILI; MIRJALILI; LEWIS, 2014) para a escolha da melhor *Edge* quando for alocar os recursos do veículo. Neste trabalho, é considerado que os veículos possuem recursos de processamento, armazenamento, tempo e memória. O algoritmo faz uso destes recursos para computar a adequação de cada *Edge* e decidir em qual delas alocar, se possível. Essa abordagem é comparada com outras duas políticas e obteve um número menor de serviços recusados e apresentou um baixo número de bloqueios durante a busca por uma *Edge*.

2.4 Considerações Finais

Este capítulo apresentou conceitos importantes para a contextualização e entendimento deste projeto em toda sua completude, além de trabalhos relacionados os quais é possível observar avanços ao longo do tempo quando se trata de alocação de recursos e tarefas, em um ponto onde já se considera e trabalha com o recebimento de tarefas e formação de VCs de forma dinâmica para alocação em tempo real.

O capítulo seguinte irá descorrer em detalhes a respeito do desenvolvimento do projeto, começando pelo cenário do problema, objetivos e proposta. Em seguida, uma descrição sobre as atividades realizadas e depois geração e análise dos resultados. Por fim, um levantamento das dificuldades e limitações envolvidas durante o desenvolvimento do trabalho.

Capítulo 3

DESENVOLVIMENTO

3.1 Considerações Iniciais

O presente capítulo aborda em detalhes todo o processo de desenvolvimento do projeto, que inclui descrição do modelo de sistema, definição do problema, atividades realizadas, metodologia para simulação, aquisição e análise dos resultados, dificuldades e limitações envolvidas no decorrer do trabalho e algumas considerações finais. Modelo de Sistema, Definição do Problema e Metodologia para Simulação foram baseados no que foi apresentado em (COSTA *et al.*, 2020).

3.2 O Projeto

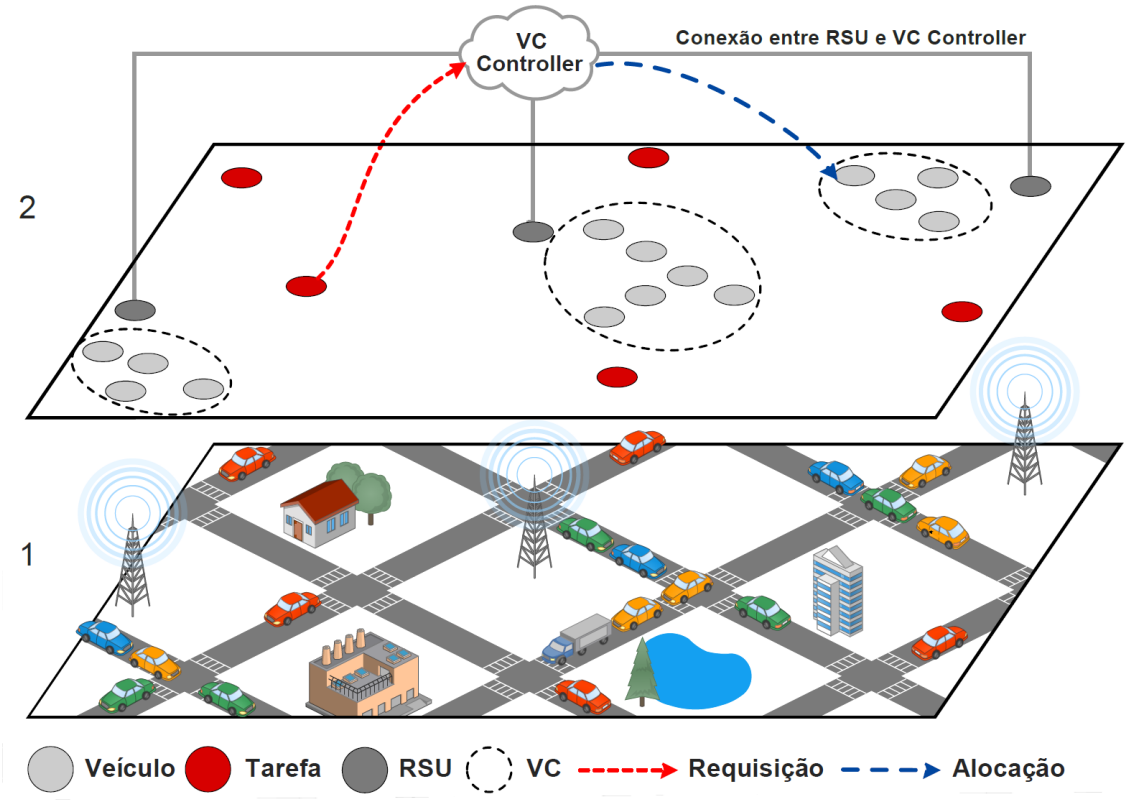
3.2.1 Modelo de Sistema

A figura 5 representa o modelo do sistema em duas visões distintas. A visão número 1, que está embaixo, apresenta o cenário físico com os veículos e as RSUs, e a visão 2 representa o que está por trás, no caso como funciona a interação entre os componentes do sistema. Neste cenário, os veículos em deslocamento formam VCs por meio de clusterizações. Quando um veículo não pertence à nenhuma VC, mas precisa realizar o processamento de alguma tarefa e seus recursos computacionais não são suficientes para tal, solicita-se ao controlador de nuvem veicular (*VC Controller*) os recursos necessários para o atendimento da tarefa.

Considera-se um cenário composto por x veículos, onde cada veículo v_i possui uma identificação única ($i \in [1,x]$) e é equipado com uma *On-Board Unit* (OBU) que permite tanto a comunicação entre veículos (V2V) quanto entre veículos e RSU (V2I). Por meio das RSUs é possível a comunicação com o *VC Controller*, cujo papel é definir, dado um conjunto de VCs disponíveis, qual delas é capaz de atender uma determinada tarefa que requer um certo poder computacional. As RSUs coletam informações dos veículos em tempo real, e quando os mesmos requisitam recursos para alocarem suas tarefas, o controlador de VC tem o conhecimento de quais nuvens possuem os recursos necessários para realizar a alocação das tarefas.

O *VC Controller* executa uma determinada política de alocação e seleciona alguma das VCs disponíveis para atender a requisição por recursos e alocar a tarefa, de forma mais rápida possível e fazendo uso do máximo de recursos computacionais ociosos da VC.

Figura 5 – Ilustração representativa da Arquitetura do Sistema



Fonte: (COSTA *et al.*, 2020)

3.2.2 Definição do Problema

Considerando um conjunto de tarefas T com $\{t = 1, ..., n\}$, em que cada tarefa t é representada por uma tupla (id_t, p_t, g_t), onde id_t corresponde ao identificador único de cada tarefa, p_t o peso da tarefa (quantidade de recursos necessários para ser alocada) e g_t o ganho/recompensa por alocação bem-sucedida. Neste cenário, as VCs são os grupos de veículos formados a partir de um processo de clusterização, este baseado em algum critério pré-estabelecido, e cada VC corresponde a soma dos recursos compartilhados por cada veículo que faz parte da respectiva nuvem.

Em (COSTA *et al.*, 2020) define-se o Problema de Alocação de Tarefas em Nuvem Veicular (PATNV), modelado a partir do Problema da Mochila 0/1 (PM0/1) por conta das características semelhantes entre os problemas. No PM0/1, tem-se uma mochila de capacidade W e diversos itens, cada um deles com um respectivo peso e valor agregado. O objetivo é maximizar o somatório do valor dos itens guardados na mochila sem que seu limite de capacidade seja ultrapassado. É definido que cada item só pode ser guardado uma única vez na mochila (0 para não escolhido, 1 para escolhido).

Análogo ao PM0/1, no PATNV as tarefas do conjunto T são os itens e as nuvens veiculares

$vc_j \in VC = \{vc_1, ..., vc_m\}$ são as mochilas. O recurso total Ω_j de cada VC é a soma dos recursos compartilhados ω_i por cada veículo v_i $(i = 1, ..., x)$ pertencente à essa nuvem, resultando na expressão $\Omega_j = \sum_{i=1}^{x} \omega_i$, $\forall v_i \in vc_j$. O ganho g_t de alocação de uma tarefa id_t cresce, em média, com o dobro de seu peso p_t. Sendo assim, tarefas mais pesadas (que requerem maior poder computacional para serem atendidas) tendem a possuir uma recompensa maior. Sendo assim, o PATNV se mostra como um problema de otimização combinatória em que almeja-se maximizar o ganho por alocar tarefas ao passo que o consumo de recursos computacionais também seja maximizado.

O PATNV é definido como:

$$\max \sum_{t=1}^{n} g_t \alpha_t$$

$$\text{e está sujeito a } \sum_{t=1}^{n} p_t \alpha_t \leq \Omega_j \text{ e } \alpha_t \in \{0, 1\}$$

onde,

$$\alpha_t = \begin{cases} 1, & \text{se a tarefa } t \text{ for adicionada à nuvem veicular,} \\ 0, & \text{caso contrário} \end{cases} \tag{3.1}$$

Porém, neste trabalho o foco da política de alocação baseada em Teoria dos Jogos será na maximização do uso dos recursos computacionais das nuvens veiculares dentro das limitações do modelo de jogo que será proposto. Portanto, o problema tem como intuito:

$$\max \sum_{t=1}^{n} p_t \alpha_t \leq \Omega_j$$

3.3 Atividades Realizadas

3.3.1 Modelagem do Jogo

A partir de uma análise a respeito do modelo de sistema e da descrição do problema, fica evidente que os jogadores mais adequados para o modelo de jogo são os veículos consumidores de serviços, os quais estão relacionandos com as tarefas a serem atendidas pelo *VC Controller*, e as VCs/veículos provedores, os quais estão relaciondas aos recursos computacionais compartilhados.

Definido os jogadores, pondera-se então a respeito das possíveis estratégias adotadas por cada um deles. Um veículo que demanda por um serviço pode escolher entre consumir ou não o serviço oferecido (permitir ou não a alocação de uma determinada tarefa) dado o custo para tal, enquanto que a VC também possui duas opções, oferecer ou não os recursos computacionais necessários para a execução da tarefa.

A ideia principal do modelo de jogo proposto é que ambas as partes envolvidas (veículos consumidores e VCs/veículos provedores) tenham algum tipo de ganho, seja ele monetário ao

oferecer um serviço em que há um custo envolvido ou valor agregado através do atendimento do serviço solicitado. A ideia secundária é o jogo operar de acordo com a oferta de recursos computacionais e peso das tarefas, fazendo então o custo variar em detrimento desses fatores. Por exemplo, no caso de haver poucos recursos disponíveis o preço por recurso aumenta e caso haja uma grande disponibilidade de recursos o preço deles reduz.

Segue abaixo as equações de ganho para cada combinação de estratégias dos jogadores:

- **Consumir & Oferecer:**

$$X_{11} = g_t - p_t \cdot c(p_t, \Omega_j)$$

$$Y_{11} = p_t \cdot c(p_t, \Omega_j)$$

- **Consumir & Não oferecer:**

$$X_{12} = -g_t$$

$$Y_{12} = -2 \cdot p_t \cdot c(p_t, \Omega_j)$$

- **Não consumir & Oferecer:**

$$X_{21} = -2 \cdot g_t$$

$$Y_{21} = -p_t \cdot c(p_t, \Omega_j)$$

- **Não consumir & Não oferecer:**

$$X_{22} = 0$$

$$Y_{22} = 0$$

⋆ $c(p_t, \Omega_j)$ é um custo por recurso em função da quantidade de recursos computacionais disponívieis Ω_j na VC com maior oferta e p_t o peso da tarefa em questão.

⋆ X_{ij} e Y_{ij} representam respectivamente os ganhos dos consumidores e provedores.

Matriz de Recompensa:

Tabela 1 – Jogo na forma normal

		Player Y	
		Oferecer	Não oferecer
Player X	Consumir	(X_{11}, Y_{11})	(X_{12}, Y_{12})
	Não consumir	(X_{21}, Y_{21})	(X_{22}, Y_{22})

Para encontrar o Equilíbrio de Nash, analisa-se cada combinação de estratégia:

– **Combinação de estratégia (Consumir & Oferecer)**
Neste primeiro caso, a recompensa de quem consome o serviço é dada por X_{11}, onde g_t, já definido anteriormente, é o valor agregado pela execução da tarefa t e p_t seu peso em quantidade de recursos necessários para o processamento. A recompensa da nuvem veicular vc_j que fornece os recursos computacionais para o atendimento da tarefa t se dá por Y_{11}, em que $c(p_t, \Omega_j)$ é uma função que varia de acordo com o peso p_t da tarefa t e a quantidade de recursos disponíveis Ω_j da VC. Dependendo do resultado de X_{11}, nenhum dos jogadores consegue aumentar seu ganho ao trocar de estratégia, já que os outros ganhos são neutros ou negativos, então essa combinação é um possível equilíbrio de Nash.

– **Combinação de estratégia (Consumir & Não oferecer)**
Neste segundo caso, onde há a escolha de consumir mas não de oferecer, o ganho do consumidor (X_{12}) é negativo pelo fato de não ter sua tarefa atendida. O provedor será penalizado com um ganho negativo dobrado (Y_{12}), pois antes da formação do jogo é considerado que há recursos suficientes para atendimento da tarefa t de peso p_t. Essa combinação não representa um possível equilíbrio de Nash já que tanto o jogador X poderia aumentar sua recompensa ao escolher não consumir quanto o jogador Y ao decidir oferecer.

– **Combinação de estratégia (Não consumir & Oferecer)**
Neste terceiro caso, em que as ações são de não consumir e oferecer, o consumidor tem uma recompensa negativa dobrada (X_{21}) como forma de penalização e o provedor tem uma recompensa negativa (Y_{21}) por não ter seus recursos computacionais utilizados. Da mesma maneira que a combinação anterior, esta não é um equilíbrio de Nash pelos mesmos motivos de que ambos jogadores poderiam aumentar seus ganhos ao mudarem suas ações.

– **Combinação de estratégia (Não consumir & Não oferecer)**
Neste último caso, como não há interesse de nenhuma das partes em consumir ou oferecer o serviço, não há necessidade de penalização e o ganho de ambos é neutro ($X_{22} = Y_{22} = 0$). Similarmente ao primeiro caso, nesta combinação nenhum dos jogadores consegue elevar sua recompensa ao escolher a outra estratégia, configurando então como um equilíbrio de Nash.

3.3.2 Implementação da política de alocação

No código-fonte 1 é possível ver como se dá o funcionamento da política que será denominada de *Game Theory Allocation* (GTA). São passados como parâmetros de função a lista de VCs W e o conjunto de tarefas T.

A política trabalha enquanto houver tarefas na lista, sendo que estas são ordenadas por peso p_t de forma decrescente. Isso é feito para garantir que a maior quantidade de recursos

seja utilizada, já que se tarefas mais leves forem atendidas primeiro, pode não haver recursos suficientes para as mais pesadas posteriormente. Se a VC de maior abundância de recursos tiver recursos suficientes para atender a tarefa da vez, então o jogo é montado e a política continua o trabalho, caso contrário segue com a próxima tarefa.

Cada jogo gerado na implementação segue o modelo descrito na subseção de Modelagem do Jogo. Para cada tarefa e quantidade de recursos da VC mais abundante é criado um jogo, pois cada matriz de recompensa e o ganho por estratégia dos jogadores variam com o valor da tarefa g_t, peso p_t e custo por recurso $c(p_t, \Omega_j)$.

Para cada jogo encontra(m)-se a(s) combinação(ões) de estratégia que configuram um equilíbrio de Nash. Caso a combinação de estratégia *Consumir & Oferecer* seja um equilíbrio de Nash, é realizada a alocação da tarefa. Portanto, são subtraídos os recursos utilizados e em seguida adiciona-se o peso da tarefa ao peso acumulado, o valor da tarefa ao valor acumulado e inclui a tarefa na lista das alocadas. Ao final do código, é retornado o valor total acumulado, a quantidade de tarefas atendidas e o peso total acumulado.

Código-fonte 1: Implementação do GTA

```
1 def gta(W, T):
2
3     VCC=[]
4     for i,j in W.items():
5         VCC.append(j)
6
7     Tarefas_alocadas = []
8     peso_acumulado = 0
9     valor_acumulado = 0
10    Tarefas = sorted(T, key=weight, reverse=True)
11
12    while len(Tarefas) > 0:
13        tarefa = Tarefas.pop(0)
14        index_max = VCC.index(max(VCC))
15        max_resource = max(VCC)
16        if (max_resource >= weight(tarefa)):
17
18            p1_11 = value(tarefa) - (weight(tarefa) *
      cost_per_weight_unit(weight(tarefa), max_resource))
19            p1_12 = -value(tarefa)
20            p1_21 = -2*value(tarefa)
21            p1_22 = 0
22            player1 = np.array([[p1_11, p1_12], [p1_21, p1_22]])
23
24            p2_11 = weight(tarefa) * cost_per_weight_unit(weight(
      tarefa), max_resource)
25            p2_12 = -2 * weight(tarefa) * cost_per_weight_unit(weight
```

```
    (tarefa), max_resource)
26              p2_21 = -weight(tarefa) * cost_per_weight_unit(weight(
    tarefa), max_resource)
27              p2_22 = 0
28              player2 = np.array([[p2_11, p2_12], [p2_21, p2_22]])
29
30              Matriz_de_Recompensa = nash.Game(player1, player2)
31
32              eqs = Matriz_de_Recompensa.support_enumeration()
33              nash_equilibrium = list(eqs)
34
35              if ((nash_equilibrium[0][0] == [1., 0.]).all() and (
    nash_equilibrium[0][1] == [1., 0.]).all()):
36                  Tarefas_alocadas.append(tarefa)
37                  VCC[index_max] -= weight(tarefa)
38                  peso_acumulado += weight(tarefa)
39                  valor_acumulado += value(tarefa)
40
41          else: continue
42
43      return valor_acumulado, len(Tarefas_alocadas), peso_acumulado
```

3.3.3 Metodologia para Simulação

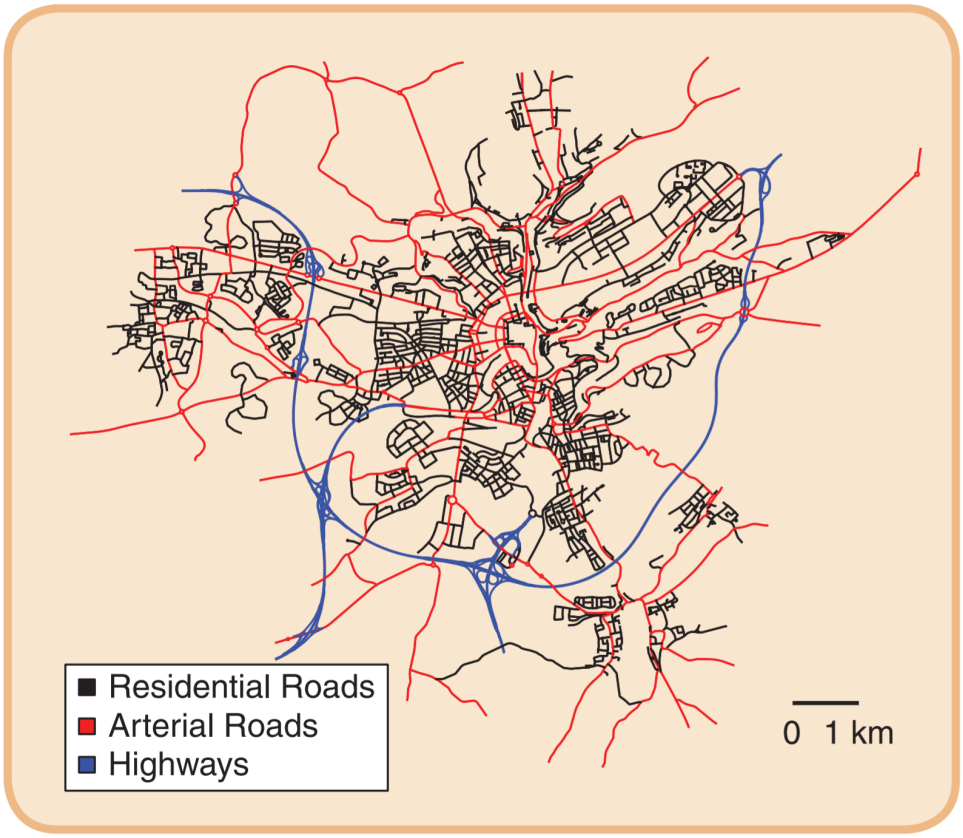
A simulação foi realizada utilizado o *Simulation of Urban Mobility* (SUMO)[1] versão 1.4.0, um simulador de mobilidade urbana de código aberto. Os algoritmos apresentados foram implementados na linguagem *Python*[2] e a conexão com o SUMO feita através do *Traffic Control Interface* (TraCI)[3]. De forma que o cenário de tráfego urbano seja o mais realista possível, foi utilizado o *trace* de mobilidade *Luxembourg SUMO Traffic (LuST)* (CODECA *et al.*, 2017), ilustrado na figura 6.

Este cenário contém 24 horas de mobilidade urbana com até aproximadamente 5000 veículos circulando nos horários de pico, isso pode ser evidenciado na figura 7. A faixa de horário escolhida foi das 11:00 às 12:00, quando há uma baixa densidade veicular e consequentemente uma maior escassez de recursos computacionais compartilhados nas VCs, para então avaliar como as políticas de alocação operam quando a situação é mais desafiadora.

O intervalo de clusterização definido foi de 60s por conta da dinamicidade da rede e alterações recorrentes nas VCs. As tarefas são geradas no momento em que o agrupamento dos veículos é feito e então as políticas as alocam nas VCs disponíveis. A ocorrência de tarefas no sistema, independentes umas das outras, é definida seguindo uma distribuição de Poisson

[1] https://sumo.dlr.de/docs/
[2] https://www.python.org/
[3] https://sumo.dlr.de/docs/TraCI.html

Figura 6 – Cenário LuST



Fonte: (CODECA *et al.*, 2017)

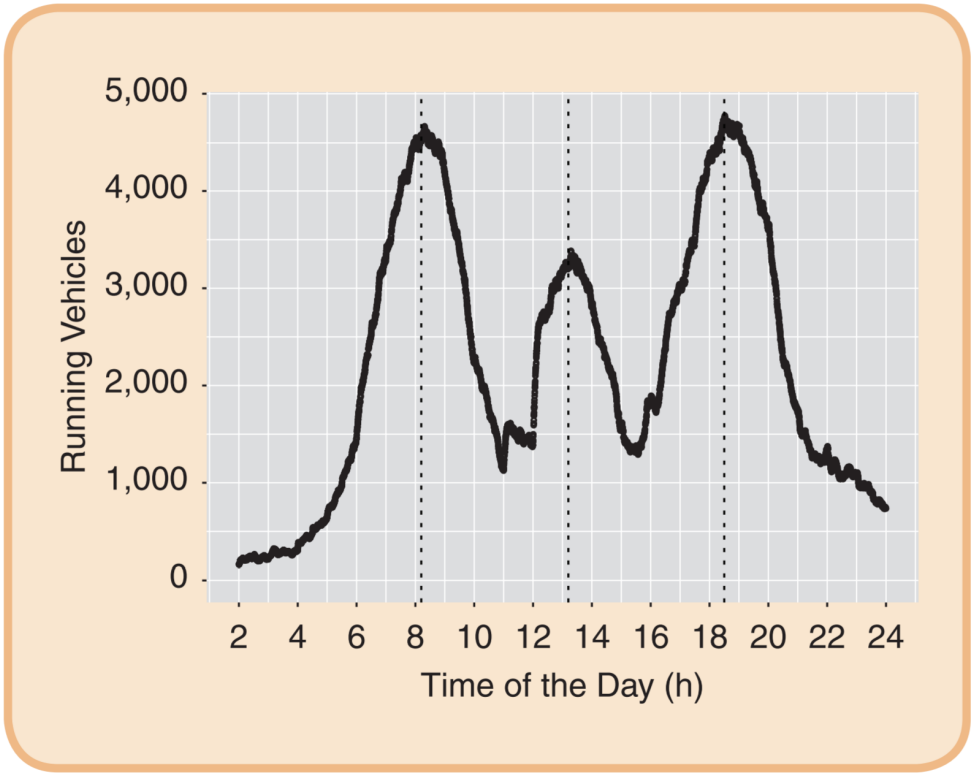
com média de tarefas $\lambda = 25$. O raio de comunicação escolhido foi de 100 metros e mínimo de 2 veículos por nuvem veicular. Houve também variação da média de peso p_t das tarefas geradas nas simulações com $\mu = \{1, 5, 10, 15, 20\}$, o ganho de alocação g_t variou com $2 \times \mu$ e a quantidade de recursos computacionais compartilháveis por veículo em $\omega_i = \{1, 2, 3\}$. Todos estes parâmetros utilizados estão sintetizados na Tabela 2

Tabela 2 – Parâmetros de Simulação

Parâmetro	**Valor**
Área do cenário	$155,95\,km^2$
Tempo de simulação	1h de LuST Scenario (11h-12h)
Intervalo de clusterização	60s
Algoritmo de clusterização	DBSCAN
Ocorrência de tarefas	Distribuição de Poisson
Número médio de tarefas (λ)	25
Peso médio das tarefas	$\{1, 5, 10, 15, 20\}$
Ganho médio das tarefas	$2 \times \{1, 5, 10, 15, 20\}$
Quantidade de recursos por veículo	{1, 2, 3}

Abaixo seguem as métricas utilizadas para avaliação de desempenho dos algoritmos:

Figura 7 – Veículos ao longo do dia



Fonte: (CODECA *et al.*, 2017)

1. **Ganho de alocação**
   Representa valor total agregado pela alocação de tarefas.
   $$ganho = \sum_{t=1}^{n} g_t \mid g_t \in \mathbb{S}$$
2. **Atendimento de tarefas**
   Representa a porcentagem de tarefas alocadas com sucesso.
   $$atendimento = \frac{\sum_{t=1}^{n} \alpha_t \mid \alpha_t \in \mathbb{S}}{|T|} \cdot 100$$
3. **Utilização de recursos**
   Representa a porcentagem de recursos computacionais de VCs utilizados.
   $$utilizacao = \frac{\sum_{t=1}^{n} p_t \mid p_t \in \mathbb{S}}{\sum_{j=1}^{m} \Omega_j} \cdot 100$$

⋆ $\mathbb{S}$ representa o conjunto de tarefas que tiveram uma alocação bem sucedida.

Para avaliar o desempenho da política de alocação deste trabalho, há comparação com outras, sendo elas uma abordagem envolvendo programção dinâmica a qual será referida nos resultados como *Dynamic Programming* (DP), uma gulosa referida como *Greedy* em que o algoritmo considera apenas uma VC e por fim uma gulosa referida como *Greedy-N* em que o algoritmo considera todas as VCs disponíveis.

3.4 Resultados

3.4.1 Aquisição dos Resultados

Os códigos-fonte 2, 3 e 4 funcionam de maneira similar e retornam respectivamente as informações dos grupos de tabelas (3, 4, 5), (6, 7, 8) e (9, 10, 11). De maneira resumida, em cada um dos códigos abaixo são acessados os arquivos de saída resultantes da simulação e são computadas respectivamente as informações de ganho de alocação, percentual de tarefas atendidas e percentual de utilização de recursos, cada uma das métricas por algoritmo, por peso médio de tarefa (λ) e por quantidade de recursos computacionais compartilhados por veículo (ω_i).

Código-fonte 2: Subrotina para computação do ganho de alocação

```
1 def print_alocacao_ganho():
2
3   print ("\nGANHO: ")
4
5   for algorithm in ALGORITHM:
6
7     print(LABEL[algorithm],)
8
9     gain_values = []
10    confidence_intervals = []
11
12    for tamanho in TAMANHO:
13
14      values = []
15      gain = 0
16
17      # 0 tempo 1 vcc 2 tarefas 3 algoritmo 4 ganho 5
      tarefas_alocadas
18      file_input_name = 'allocation_'+str(cenario_raio)+'_'+str(
      recursos)+'_'+str(tamanho)+'.txt'
19
20      for line in open(file_input_name):
21
22        if int(line.split()[0]) > tinicio and int(line.split()[0]) <
      tfim:
23
24          if int(line.split()[3]) == algorithm:
25            gain = int(line.split()[4])
26
27          values.append(gain)
28
29      gain_values.append(np.mean(values))
```

```
30        confidence_intervals.append(confidence_interval(values))
31
32      print(gain_values)
```

Tabela 3 – Recompensa das tarefas por peso médio de tarefa λ e recursos por veículo $\omega_i = 1$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	2	2	2	1	1
DP	1,75	1,75	1,75	1	1
Greedy-N	6	4,5	2,5	3	1
GTA	4,5	3,25	2,25	3	1

Tabela 4 – Recompensa das tarefas por peso médio de tarefa λ e recursos por veículo $\omega_i = 2$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	4	3	3	2	2
DP	3,5	3,5	3,5	3	1,5
Greedy-N	12	9	7	5,5	1
GTA	9	5,875	5	4	0,5

Tabela 5 – Recompensa das tarefas por peso médio de tarefa λ e recursos por veículo $\omega_i = 3$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	6	5	5	6	2
DP	5,25	5,25	4,75	5,25	1,75
Greedy-N	18	12,5	11	9	2
GTA	13,5	11,125	8,25	7,25	1,5

Código-fonte 3: Subrotina para computação do percentual de tarefas atendidas

```
1 def print_alocacao_eficiencia():
2
3   print("\nATENDIMENTO: ")
4
5   for algorithm in ALGORITHM:
6
7     print(LABEL[algorithm],)
8
9     number_values = []
10    confidence_intervals = []
11
12    for tamanho in TAMANHO:
13
14      values = []
15      number = 1
16
17      # 0 tempo 1 vcc 2 tarefas 3 algoritmo 4 ganho 5
      tarefas_alocadas
```

```
18      file_input_name = 'allocation_'+str(cenario_raio)+'_'+str(
    recursos)+'_'+str(tamanho)+'.txt'
19
20      for line in open(file_input_name):
21
22        if int(line.split()[0]) > tinicio and int(line.split()[0]) <
    tfim:
23
24          if int(line.split()[3]) == algorithm:
25
26            if int(line.split()[2]) > 0:
27              number = (int(line.split()[5]) * 100) / int(line.split
    ()[2])
28
29          values.append(number)
30
31      values = np.divide(values, 1)
32
33      number_values.append(np.mean(values))
34      confidence_intervals.append(confidence_interval(values))
35
36    print(number_values)
```

Tabela 6 – Porcentagem de atendimento das tarefas por peso médio de tarefa λ e recursos compartilhados $\omega_i = 1$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	5,41%	5,41%	5,41%	2,78%	2,78%
DP	4,88%	4,88%	4,88%	2,9%	2,9%
Greedy-N	16,48%	12,38%	7,12%	8,58%	3,03%
GTA	12,66%	9,22%	6,59%	8,71%	3,15%

Tabela 7 – Porcentagem de atendimento das tarefas por peso médio de tarefa λ e recursos compartilhados $\omega_i = 2$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	10,82%	8,04%	8,04%	5,41%	5,26%
DP	9,63%	6,85%	4,88%	4,88%	4,07%
Greedy-N	32,71%	20,43%	12,38%	11,07%	2,88%
GTA	24,94%	12,66%	9,22%	8,56%	1,69%

Tabela 8 – Porcentagem de atendimento das tarefas por peso médio de tarefa λ e recursos compartilhados $\omega_i = 3$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	16,23%	13,6%	13,45%	13,45%	5,41%
DP	14,38%	7,65%	6,85%	11,6%	4,88%
Greedy-N	48,93%	28,9%	19,26%	17,94%	5,66%
GTA	37,22%	13,97%	12,66%	14,78%	4,47%

Código-fonte 4: Subrotina para computação do percentual de recursos utilizados

```
def print_utilizacao():

  print("\nUTILIZACAO:")

  for algorithm in ALGORITHM:

    print(LABEL[algorithm],)

    number_values = []
    confidence_intervals = []

    number = 0

    for tamanho in TAMANHO:

      values = []

      # 0 tempo 1 vcc 2 tarefas 3 algoritmo 4 ganho 5 tarefas_alocadas
      file_input_name = 'allocation_'+str(cenario_raio)+'_'+str(recursos)+'_'+str(tamanho)+'.txt'

      for line in open(file_input_name):

        if int(line.split()[0]) > tinicio and int(line.split()[0]) < tfim:

          if int(line.split()[3]) == algorithm:
            if int(line.split()[1]) > 0:

              number = (int(line.split()[7]) * 100) / (int(line.split()[1]))

          values.append(number)

      values = np.divide(values, 1)

      number_values.append(np.mean(values))
      confidence_intervals.append(confidence_interval(values))

    print(number_values)
```

Tabela 9 – Porcentagem de utilização dos recursos por peso médio de tarefa λ e recursos compartilhados $\omega_i = 1$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	25%	25%	25%	16,67%	16,67%
DP	22,92%	25%	25%	18,75%	16,67%
Greedy-N	75%	79,17%	58,33%	54,17%	16,67%
GTA	62,5%	81,25%	66,67%	56,25%	16,67%

Tabela 10 – Porcentagem de utilização dos recursos por peso médio de tarefa λ e recursos compartilhados $\omega_i = 2$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	25%	16,67%	25%	12,5%	8,33%
DP	22,92%	25%	25%	25%	8,33%
Greedy-N	75%	91,67%	75%	56,25%	18,75%
GTA	62,5%	100%	79,17%	57,29%	23,96%

Tabela 11 – Porcentagem de utilização dos recursos por peso médio de tarefa λ e recursos compartilhados $\omega_i = 3$

Algoritmo	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$	$\lambda = 20$
Greedy	25%	22,22%	19,44%	25%	19,44%
DP	22,92%	25%	25%	25%	20,83%
Greedy-N	75%	86,11%	73,61%	72,22%	31,94%
GTA	62,5%	100%	93,75%	73,61%	34,03%

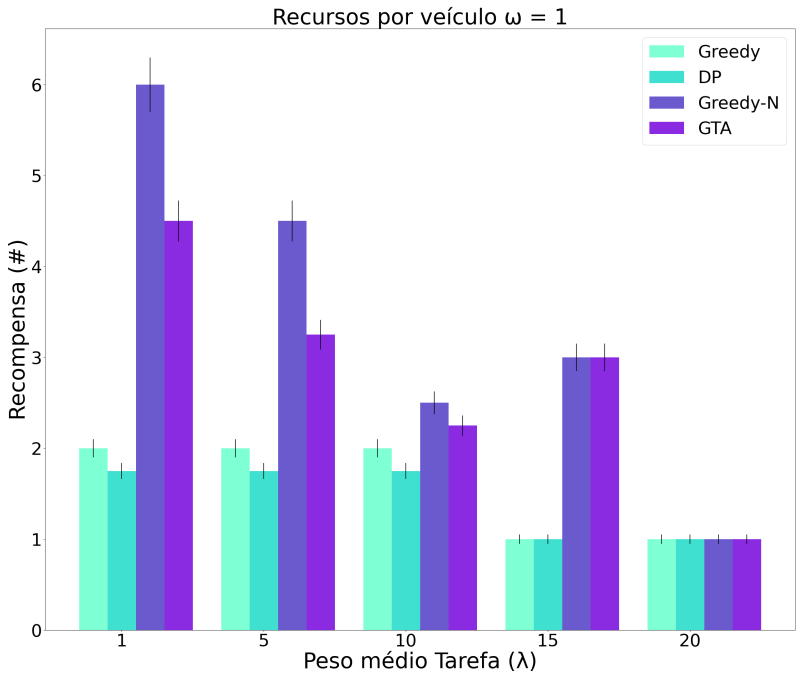
3.4.2 Análise dos Resultados

Para uma melhor visualização dos resultados, foram gerados gráficos a partir dos valores das tabelas apresentadas na subseção de aquisição dos resultados.

Nos gráficos das figuras 8, 9 e 10, gerados respectivamente a partir das informações das tabelas 3, 4 e 5, são apresentados os valores de recompensa que cada algoritmo obteve de acordo com as tarefas que foram alocadas por eles. Observa-se que de maneira geral, o GTA obteve ganhos maiores do que as abordagens *Greedy* e DP, as quais fazem uso apenas da VC com maior abundância de recursos computacionais, mas obteve ganhos menores quando comparado com a abordagem *Greedy-N*, que faz uso de todas as VCs disponíveis para realizar a alocação das tarefas. Também vale ressaltar que a medida que o peso médio das tarefas aumenta, o ganho das políticas vai se tornando próximo devido a dificuldade de atender as tarefas cada vez mais pesadas nas VCs com a mesma quantidade de recursos anterior.

Nos gráficos das figuras 11, 12 e 13, gerados respectivamente a partir dos valores das tabelas 6, 7 e 8, apresenta-se o percentual de tarefas atendidas por cada algoritmo. Os resultados de atendimento de tarefas tem um comportamento similar aos de ganho, já que essas duas métricas estão de certa forma relacionadas. Quanto mais tarefas uma política atende, a tendência é que maior seja o ganho total de alocação. Pode haver situações em que poucas tarefas de alto valor agregado sejam atendidas e resultem em um ganho elevado, que é o caso do GTA dado que ele prioriza o atendimento de tarefas mais pesadas, as quais tendem a ter uma recompensa maior, mas essa abordagem acaba se mostrando como não sendo a melhor possível dado que o

Figura 8 – Gráfico de recompensa por atendimento de tarefas ($\omega_i = 1$)



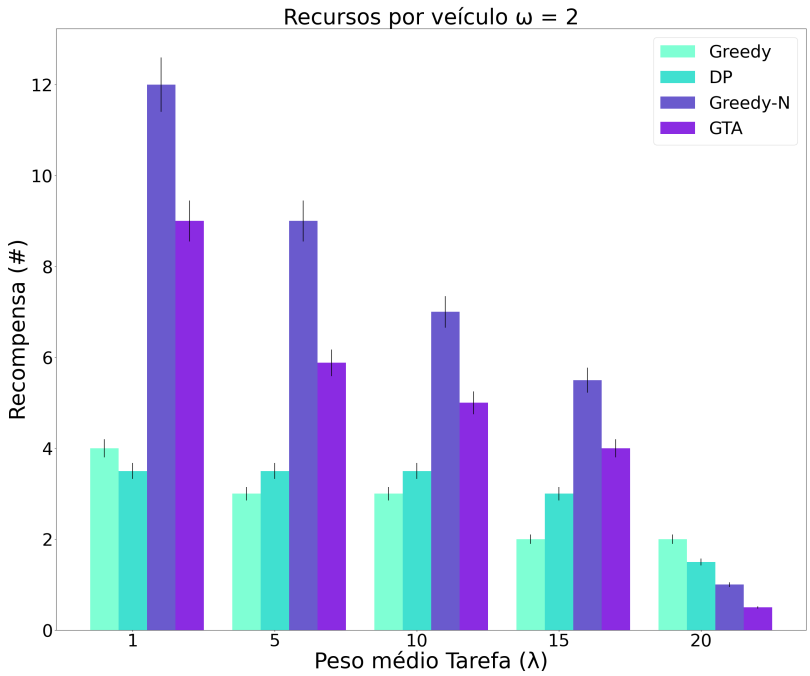
algoritmo *Greedy-N* obteve resultados melhores.

Nos gráficos das figuras 14, 15 e 16, gerados respectivamente a partir dos valores das tabelas 9, 10 e 11, mostra-se a porcentagem de recursos utilizados por cada algoritmo em suas alocações. É evidente ao observar os resultados que o GTA cumpre seu principal objetivo que é maximizar a utilização dos recursos computacionais disponíveis, tendo resultado melhor que todos os outros algoritmos. Todavia, é visível que maior utilização dos recursos não implica diretamente em um maior atendimento ou maior ganho, já que nas outras duas métricas o GTA teve um desempenho inferior ao *Greedy-N*. Ao priorizar tarefas mais pesadas, o GTA acaba inevitavelmente atendendo um número menor de tarefas, e as mesmas por mais pesadas que sejam, não é garantido que possuam um valor agregado consideravelmente maior que outras tarefas que exijam uma menor quantidade de recursos das VCs.

3.5 Dificuldades e Limitações

A primeira dificuldade encontrada, durante o levantamento bibliográfico, foi a escassez de trabalhos de pesquisa voltados para o uso de teoria dos jogos na seleção de serviços ou alocação de recursos e tarefas em nuvens veiculares, tornando o embasamento teórico bastante limitado. Além da limitação no embasamento teórico, tanto a implementação de maneira geral quanto especificamente os parâmetros recebidos pelos algoritmos usados como política de alocação engessavam a forma como o jogo poderia ser modelado, dado que as informações e

Figura 9 – Gráfico de recompensa por atendimento de tarefas ($\omega_i = 2$)



parâmetros úteis para cálculo da recompensa dos jogadores e posteriormente tomada de decisão eram restritos.

Outra dificuldade que se apresentou durante o desenvolvimento do projeto foi na tentativa de uso do simulador de redes OMNeT++[4] juntamente com o *framework* de simulação de redes veiculares Veins[5] e o SUMO[6] (este último já mencionado anteriormente). Tanto a inexperiência com as ferramentas quanto a dificuldade de adequar o projeto a elas tornou inviável o seu uso. Para que houvesse tempo hábil de finalização do trabalho foi decidido por uma mudança de abordagem para a descrita na subseção de Metodologia para Simualação.

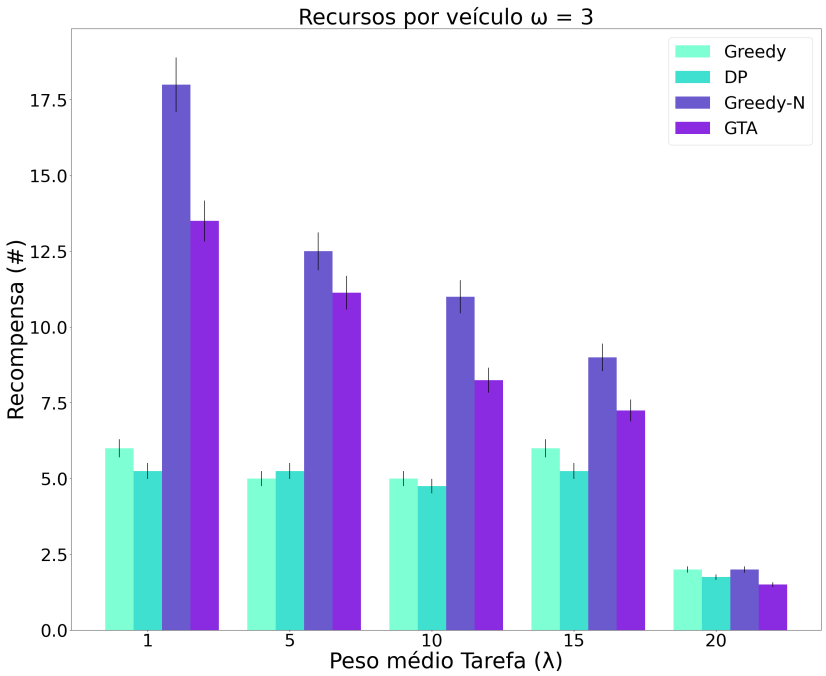
Um fator que também limitou o projeto foi o peso da simulação e tempo para obtenção dos resultados. Esse ponto tornou consideravelmente longa a realização de testes e refinamento do modelo de jogo/implementação.

3.6 Considerações Finais

Este capítulo apresentou e detalhou diversos elementos importantes do projeto, começando pelo modelo de sistema e descrição da problemática em que o projeto está inserido, em seguida discorrendo a respeito das atividades realizadas para que se chegasse à solução proposta,

[4] https://omnetpp.org/
[5] https://veins.car2x.org/
[6] https://www.eclipse.org/sumo/

Figura 10 – Gráfico de recompensa por atendimento de tarefas ($\omega_i = 3$)



posteriormente comentando sobre a aquisição e análise dos resultados obtidos e por fim expondo as dificuldades e limitações envolvidas na condução do trabalho.

No próximo e último capítulo, tem-se um fechamento deste trabalho com relação às contribuições gerais do projeto, o quanto o curso de Engenharia de Computação agregou para esta realização e por fim algumas diretrizes de como melhorar e dar continuidade em trabalho futuros.

Figura 11 – Gráfico do percentual de atendimento de tarefas ($\omega_i = 1$)

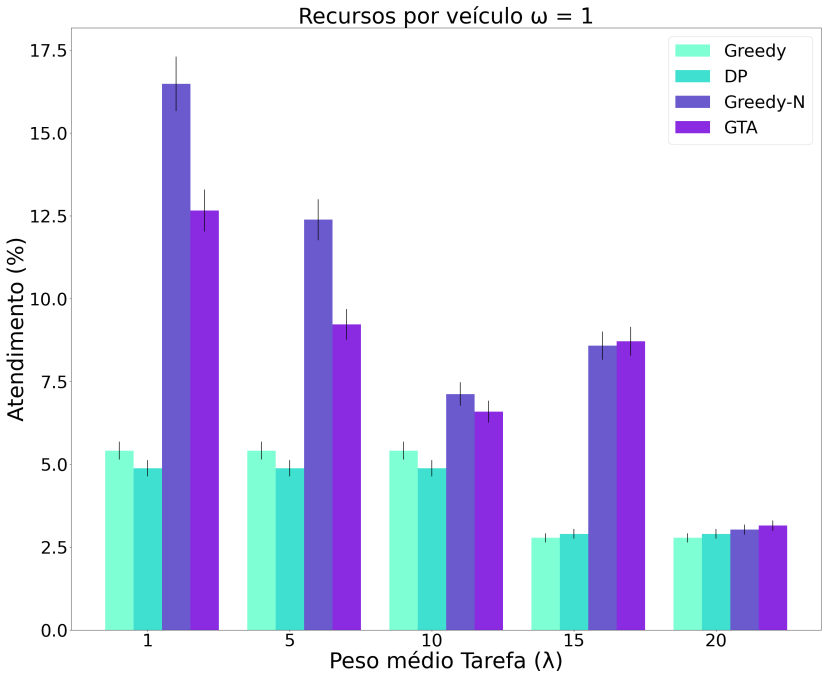


Figura 12 – Gráfico do percentual de atendimento de tarefas ($\omega_i = 2$)

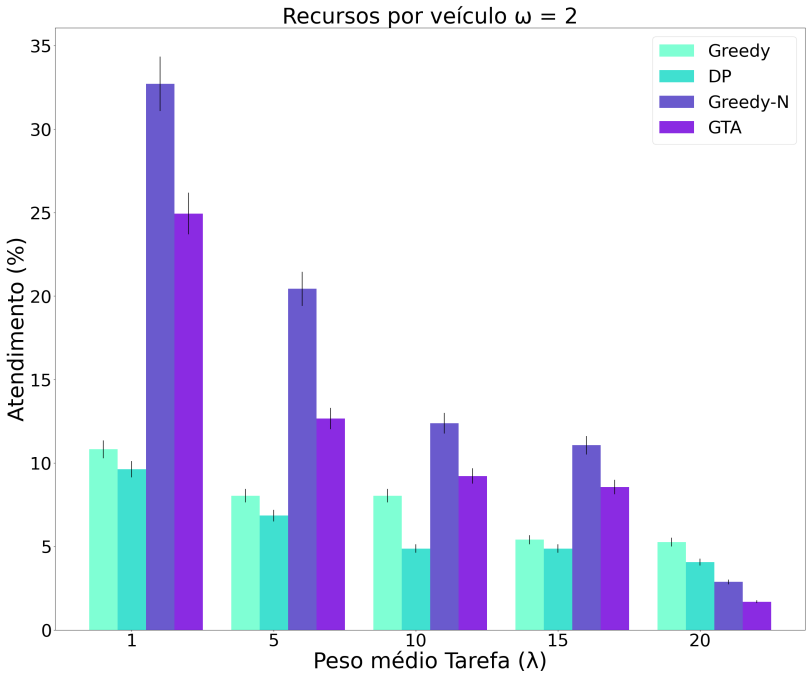


Figura 13 – Gráfico do percentual de atendimento de tarefas ($\omega_i = 3$)

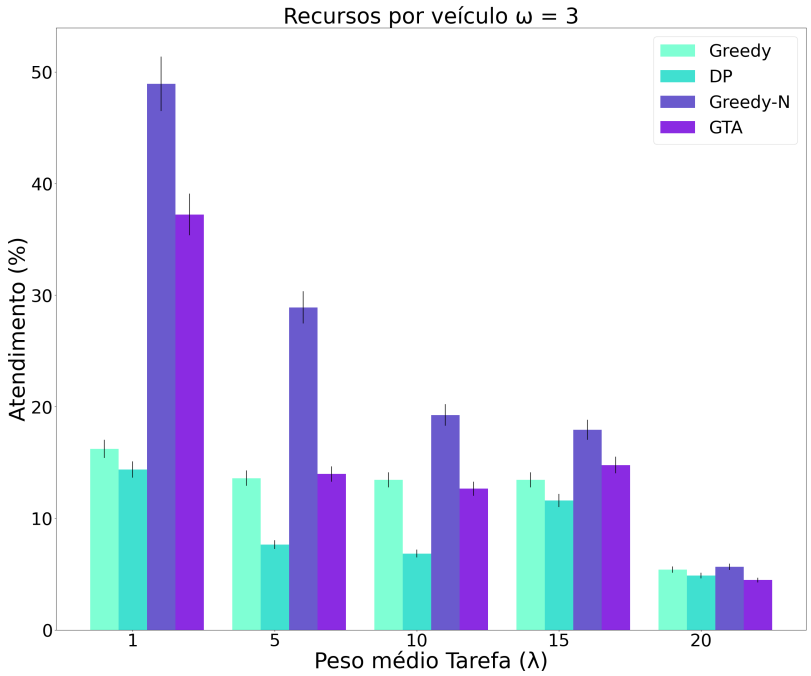


Figura 14 – Gráfico do percentual de utilização dos recursos ($\omega_i = 1$)

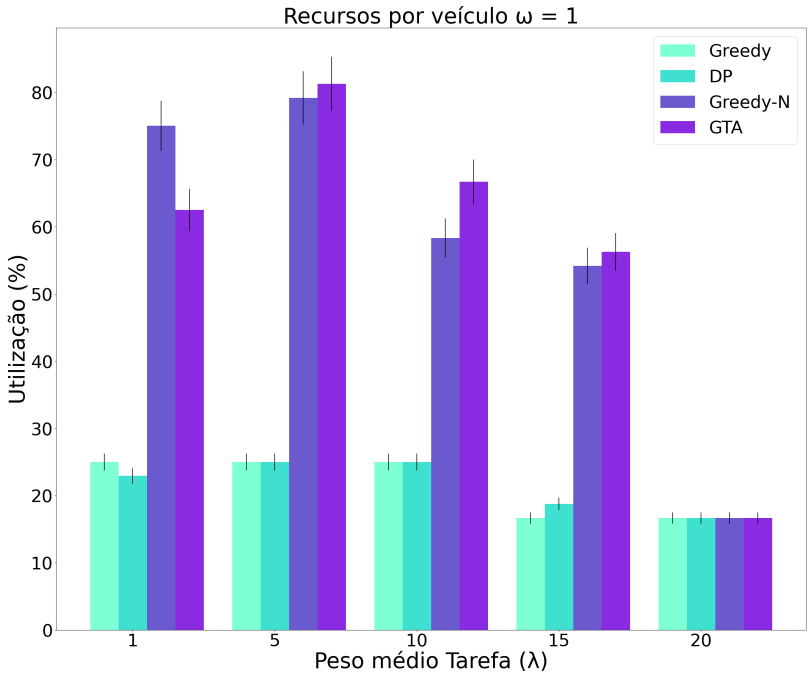


Figura 15 – Gráfico do percentual de utilização dos recursos ($\omega_i = 2$)

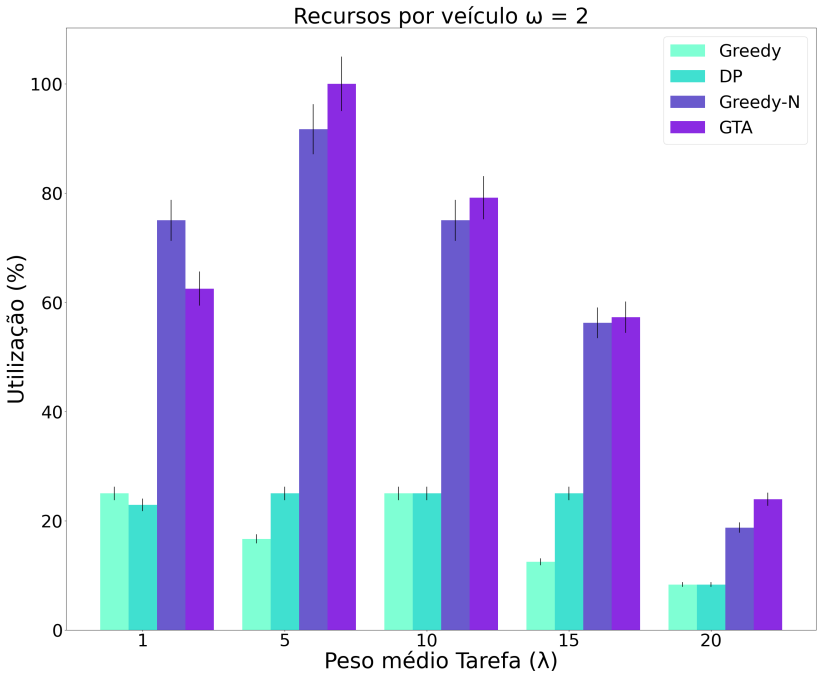
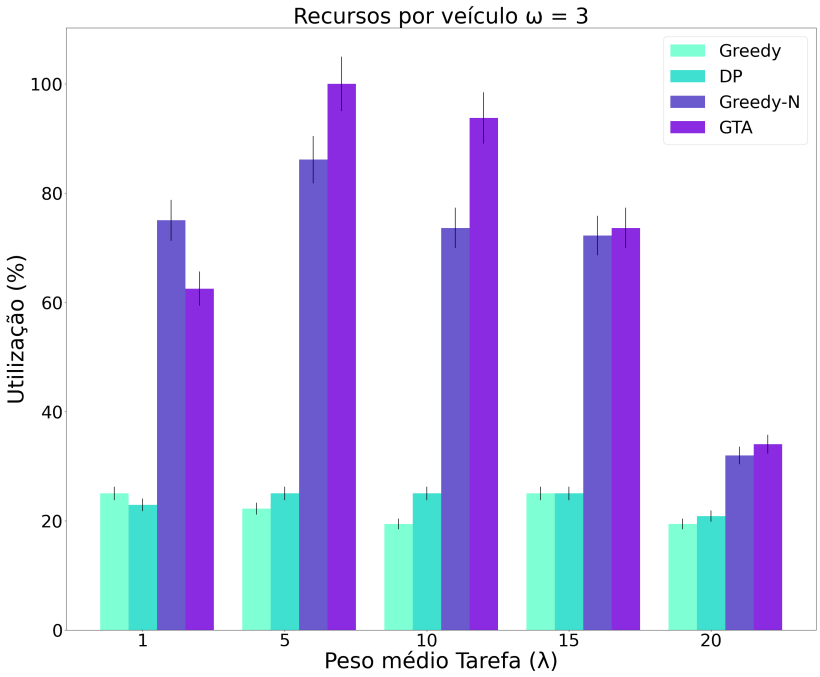


Figura 16 – Gráfico do percentual de utilização dos recursos ($\omega_i = 3$)



Capítulo 4

CONCLUSÃO

4.1 Contribuições

As principais contribuições deste trabalho para o momento atual e pesquisas futuras envolvem a fomentação dos primeiros passos para o estabelecimento do uso de Teoria dos Jogos como uma política de alocação de recursos viável e de grande interesse para todos os participantes do cenário, visto que são levados em consideração os comportamentos de cada um deles, os possíveis ganhos nas estratégias adotadas e a racionalidade em suas decisões, se adequando melhor a uma realidade onde há um custo envolvido na utlização dos recursos e serviços fornecidos nos ITS e VANETs. Este trabalho também comprova que uma maior utilização dos recursos computacionais não vem necessariamente acompanhada de um maior ganho de alocação ou maior número de tarefas atendidas, fato este salientado na parte de análise dos resultados, significando que ainda há bastante espaço para melhorias e trabalho a ser desenvolvido nesta política.

Pessoalmente acredito que o estímulo e realização de pesquisas nesse âmbito se torna ainda mais importante pelo fato do Brasil ser um país em que os meios de transporte são majoritariamente rodoviários, permitindo então que os avanços acadêmicos nesta área se mostrem extremamente valiosos e de grande utilidade em um futuro onde a grande maioria dos carros estarão conectados à rede ou entre si.

As experiências adquiridas no desenvolvimento deste projeto foram mais voltadas para o lado acadêmico e técnico. As experiëncias acadêmicas obtidas se deram através da necessidade de realizar um bom levantamento bibliográfico de forma a ter o melhor embasamento teórico possível, trabalhar a escrita e estruturação correta de um projeto de pesquisa e citação de referências. As experiências técnicas adquiridas foram através do estudo e uso de simuladores de mobilidade urbana.

4.2 Trabalhos Futuros

Algumas diretrizes para trabalhos futuros que seriam interessantes:

– Refinamento do modelo de jogo e implementação do mesmo para que a política performe melhor com relação às métricas de ganho de alocação e atendimento de tarefas.

- Comparação com outras políticas que possuam um desempenho mais elevado do que as presentes neste trabalho.
- Uso de novos cenários de simulação
- Adição de elementos mais restritivos ao cenário como um prazo limite de atendimento das tarefas.

REFERÊNCIAS

AL-SULTAN, S.; AL-DOORI, M. M.; AL-BAYATTI, A. H.; ZEDAN, H. A comprehensive survey on vehicular ad hoc network. **Journal of Network and Computer Applications**, v. 37, p. 380–392, 2014. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S108480451300074X>. Citado na página 21.

BRIK, B.; KHAN, J. A.; GHAMRI-DOUDANE, Y.; LAGRAA, N.; LAKAS, A. Gss-vc: A game-theoretic approach for service selection in vehicular cloud. In: **2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)**. [S.l.: s.n.], 2018. p. 1–6. Citado na página 28.

BRIK, B.; KHAN, J. A.; GHAMRI-DOUDANE, Y.; LAGRAA, N. Publish: A distributed service advertising scheme for vehicular cloud networks. In: **2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)**. [S.l.: s.n.], 2019. p. 1–6. Citado na página 22.

CISCO. **Driving Profits from Connected Vehicles**. [S.l.], 2020. Disponível em: <https://www.cisco.com/c/en/us/solutions/service-provider/mobile-internet/driving-profits-from-connected-vehicles.html>. Citado na página 21.

CODECA, L.; FRANK, R.; FAYE, S.; ENGEL, T. Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation. **IEEE Intelligent Transportation Systems Magazine**, v. 9, n. 2, p. 52–63, 2017. Citado 3 vezes nas páginas 37, 38 e 39.

COSTA, J. da; PEIXOTO, M.; MENEGUETTE, R.; ROSáRIO, D.; VILLAS, L. Morfeu: Mecanismo baseado em otimização combinatória para alocação de tarefas em nuvens veiculares. In: **Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. Porto Alegre, RS, Brasil: SBC, 2020. p. 505–518. ISSN 2177-9384. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/12305>. Citado 4 vezes nas páginas 23, 28, 31 e 32.

COUTINHO, R. W. L.; BOUKERCHE, A. Guidelines for the design of vehicular cloud infrastructures for connected autonomous vehicles. **IEEE Wireless Communications**, v. 26, n. 4, p. 6–11, 2019. Citado na página 23.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining**. [S.l.]: AAAI Press, 1996. p. 226–231. Citado na página 29.

HAGENAUER, F.; HIGUCHI, T.; ALTINTAS, O.; DRESSLER, F. Efficient data handling in vehicular micro clouds. **Ad Hoc Networks**, v. 91, p. 101871, 2019. ISSN 1570-8705. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1570870518309107>. Citado na página 22.

HATTAB, G.; UCAR, S.; HIGUCHI, T.; ALTINTAS, O.; DRESSLER, F.; CABRIC, D. Optimized assignment of computational tasks in vehicular micro clouds. In: **Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking**. New York, NY, USA: Association for Computing Machinery, 2019. (EdgeSys '19), p. 1–6. ISBN 9781450362757. Disponível em: <https://doi.org/10.1145/3301418.3313937>. Citado na página 22.

LIEIRA, D. D.; QUESSADA, M. S.; CRISTIANI, A. L.; MENEGUETTE, R. I. Resource allocation technique for edge computing using grey wolf optimization algorithm. In: **2020 IEEE Latin-American Conference on Communications (LATINCOM)**. [S.l.: s.n.], 2020. p. 1–6. Citado na página 29.

Meneguette, R. I.; De Grande, R. E.; Loureiro, A. A. F. **Intelligent Transport System in Smart Cities: Aspects and Challenges of Vehicular Networks and Cloud**. [S.l.]: Springer, 2018. Citado 2 vezes nas páginas 21 e 27.

MENEGUETTE, R. I.; RODRIGUES, D. O.; COSTA, J. B. D. da; ROSARIO, D.; VILLAS, L. A. A virtual machine migration policy based on multiple attribute decision in vehicular cloud scenario. In: **ICC 2019 - 2019 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2019. p. 1–6. Citado na página 23.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey wolf optimizer. **Advances in Engineering Software**, v. 69, p. 46–61, 2014. ISSN 0965-9978. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0965997813001853>. Citado na página 29.

NABI, M.; BENKOCZI, R.; ABDELHAMID, S.; HASSANEIN, H. S. Resource assignment in vehicular clouds. In: **2017 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2017. p. 1–6. Citado na página 27.

PEREIRA, R. S.; LIEIRA, D. D.; SILVA, M. A. da; PIMENTA, A. H.; COSTA, J. B. da; ROSÁRIO, D.; MENEGUETTE, R. I. A novel fog-based resource allocation policy for vehicular clouds in the highway environment. In: **2019 IEEE Latin-American Conference on Communications (LATINCOM)**. [S.l.: s.n.], 2019. p. 1–6. Citado na página 28.

Qualcomm. **Connecting vehicles to everything with C-V2X**. [S.l.], 2020. Disponível em: <https://www.qualcomm.com/research/5g/cellular-v2x>. Citado na página 21.

Red Hat. **O que é *Edge Computing*?** [S.l.], 2021. Disponível em: <https://www.redhat.com/pt-br/topics/edge-computing/what-is-edge-computing>. Citado na página 26.

ROSS, D. Game Theory. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Winter 2019. [S.l.]: Metaphysics Research Lab, Stanford University, 2019. Citado 2 vezes nas páginas 25 e 26.

SORKHOH, I.; EBRAHIMI, D.; ATALLAH, R.; ASSI, C. Workload scheduling in vehicular networks with edge cloud capabilities. **IEEE Transactions on Vehicular Technology**, v. 68, n. 9, p. 8472–8486, 2019. Citado na página 23.

THAKUR, A.; MALEKIAN, R. Fog computing for detecting vehicular congestion, an internet of vehicles based approach: A review. **IEEE Intelligent Transportation Systems Magazine**, v. 11, n. 2, p. 8–16, 2019. Citado na página 22.

WANG, J.; LIU, T.; LIU, K.; KIM, B.; XIE, J.; HAN, Z. Computation offloading over fog and cloud using multi-dimensional multiple knapsack problem. In: **2018 IEEE Global Communications Conference (GLOBECOM)**. [S.l.: s.n.], 2018. p. 1–7. Citado na página 28.

YANG, C.; LIU, Y.; CHEN, X.; ZHONG, W.; XIE, S. Efficient mobility-aware task offloading for vehicular edge computing networks. **IEEE Access**, v. 7, p. 26652–26664, 2019. Citado na página 23.