MORFEU: Mecanismo baseado em Otimização Combinatória para Alocação de Tarefas em Nuvens Veiculares

Joahannes B. D. da Costa[1], Maycon L. M. Peixoto[1,2], Rodolfo I. Meneguette[3], Denis L. Rosário[4], Leandro A. Villas[1]

[1]Instituto de Computação (UNICAMP), Campinas – São Paulo – Brasil
[2]Universidade Federal da Bahia (UFBA), Salvador – Bahia – Brasil
[3]Universidade de São Paulo (USP), São Carlos – São Paulo – Brasil
[4]Universidade Federal do Pará (UFPA), Belém – Pará – Brasil

joahannes@lrc.ic.unicamp.br, maycon.leone@ufba.br,
meneguette@icmc.usp.br, denis@ufpa.br, leandro@ic.unicamp.br

Abstract. *Intelligent Transportation Systems (ITS) make up a service framework that seeks to mitigate problems in the road sector. ITS services are facilitated by creating vehicular clouds by using the communication capabilities of other vehicles to provide cloud services closer to vehicular applications. However, often the computational resources present in vehicles are underused. To mitigate this limitation, this work proposes a mechanism based on combinatorial optimization for task allocation in vehicular clouds, called MORFEU. The mechanism considers combinatorial optimization to efficiently allocate computational tasks to be performed on vehicular clouds. Simulation results show that MORFEU can allocate more tasks, with higher gain, and less waste of computational resources compared to other evaluated allocation approaches.*

Resumo. *Os sistemas de transporte inteligentes (ITS) compõem um arcabouço de serviços que busca mitigar problemas no setor rodoviário. Os serviços de ITS são facilitados pela criação de nuvens veiculares por meio do uso dos recursos de comunicação de outros veículos para fornecer serviços em nuvem mais próximos das aplicações veiculares. No entanto, frequentemente os recursos computacionais presentes nos veículos são subutilizados. Para mitigar essa limitação, este trabalho propõe um mecanismo baseado em otimização combinatória para alocação de tarefas em nuvens veiculares, denominado de MORFEU. O mecanismo considera otimização combinatória para alocar com eficiência tarefas computacionais a serem executadas em nuvens veiculares. Resultados de simulação mostram que o MORFEU consegue alocar mais tarefas, com um ganho superior, e menor desperdício de recursos computacionais em comparação com outras abordagens de alocação avaliadas.*

1. Introdução

Nos últimos anos, o número de veículos vem crescendo de forma significativa em todo o mundo, onde existirá quase 2 bilhões de veículos conectados nas estradas até 2025 [CISCO 2019]. Juntamente com esse crescimento, a indústria automobilística vem investindo na incorporação de mais recursos tecnológicos nos veículos, tal como sensores, poder de processamento e comunicação [Qualcomm 2018], contribuindo para o surgimento

de novas direções de pesquisa. Nesse contexto, os Sistemas de Transporte Inteligentes (ITS, de *Intelligent Transportation Systems*) visam melhorar a segurança nas estradas e garantir sistemas inteligentes de tráfego urbano [Costa et al. 2018]. Para realçar o desempenho dos ITS, a Computação em Nuvem Veicular busca agregar recursos computacionais presentes nos veículos para a execução de aplicações diversas dos ITS através da criação de nuvens veiculares (VC, de *Vehicular Clouds*) [Thakur and Malekian 2019].

As VCs têm um impacto notável em diversas aplicações de ITS por utilizar instantaneamente recursos veiculares, como computação, armazenamento e comunicação para tomadas de decisão sem necessitar da computação em nuvem tradicional para isso [Brik et al. 2019]. Com tais recursos, cria-se um conjunto de serviços que são disponibilizados para outros veículos, que podem ser usados em ambientes estáticos e dinâmicos relacionados à mobilidade de veículos [Hagenauer et al. 2019]. Desta forma, os veículos se tornam uma importante ferramenta no contexto de aplicações de ITS, pois além de sua conhecida atuação no gerenciamento do tráfego de veículos, agora podem atuar como uma fonte para coleta e processamento de dados em tempo real [Hattab et al. 2019]. Portanto, é essencial considerar mecanismos que usem os recursos das VCs de maneira eficiente, reduzindo o desperdício desses recursos e fornecendo poder computacional mais próximo das aplicações veiculares.

Os principais desafios que circundam a proposição de abordagens para alocação eficiente de tarefas e recursos em VCs se voltam para a natureza dinâmica das redes veiculares [Coutinho and Boukerche 2019, Yang et al. 2019]. A alta mobilidade dos veículos torna desafiador o processo de alocação para processamento de tarefas com requisitos intolerantes ao atraso (como detecção de acidentes) [Meneguette et al. 2019]. Além disso, tarefas que exigem alto poder computacional (como processamento de imagens de trânsito e outras aplicações multimídia) também são desafiadoras para o processo como um todo [Sorkhoh et al. 2019]. Desta forma, garantir que a tarefa seja atendida independente de suas características e requisitos é um fator fundamental. É importante considerar mecanismo de alocação de tarefas que utilizem os recursos das VCs de forma eficiente, para assim consumir o máximo desses recursos e fornecer poder computacional mais próximo das aplicações de ITS.

Para atender a tais problemas, neste artigo é proposto um **M**ecanismo baseado em **O**timização combinató**R**ia para alocação de tare**F**as em nuv**E**ns veic**U**lares, chamado de **MORFEU**. O mecanismo proposto considera que as tarefas chegam de forma dinâmica e independente para alocação em tempo real também de forma dinâmica. Desta forma, o mecanismo MORFEU considera uma abordagem de otimização combinatória para seleção do conjunto ótimo de tarefas para serem alocadas em tempo real em cada uma das nuvens veiculares disponíveis. O MORFEU é executado em uma nuvem formada a partir da cooperação entre as infraestruturas de comunicação do cenário, tal como, *Road-side Units* (RSU).

Em resumo, aponta-se as principais contribuições de pesquisa deste trabalho da seguinte forma: introdução de um algoritmo eficiente para alocação de tarefas sem necessidade de conhecimento prévio do ambiente veicular e utilização eficiente dos recursos computacionais veiculares outrora subutilizados. O MORFEU foi comparado à outras abordagens para resolução do problema de alocar tarefas em VCs por meio de simulação considerando o trace de mobilidade de Luxemburgo. Os resultados mostram que o MOR-

FEU é capaz de alocar mais tarefas, considerar tarefas que remetem maior ganho por alocá-las e reduzir o desperdício de recursos computacionais dos veículos em 14.87%, 25.22% e 28.35%, respectivamente.

O restante deste artigo está organizado como se segue. A Seção 2 apresenta os principais trabalhos relacionados à esta pesquisa. A Seção 3 introduz a metodologia do trabalho, com modelo de sistema e detalhes do mecanismo MORFEU. A Seção 4 apresenta questões sobre a metodologia utilizada para avaliação do mecanismo, bem como parâmetros de simulação e discussão acerca dos resultados obtidos. Por fim, a Seção 5 apresenta as conclusões e direções para trabalhos futuros.

2. Trabalhos Relacionados

Yu et al. [Yu et al. 2013] introduziram uma estrutura teórica de jogo para o processo de alocação de recursos em VCs. Consideram uma arquitetura hierárquica para redes de veículos baseadas na nuvem, que visa facilitar o compartilhamento de recursos computacionais, como recursos de armazenamento. Para isso, a estrutura desenvolveu um ambiente de nuvem abrangente para veículos, através da integração de recursos físicos redundantes nas infraestruturas, incluindo Data Centers, RSUs e veículos. A estrutura oferece um esquema de reserva de recursos e uma estrutura teórica de jogos para resolver a competição de recursos entre máquinas virtuais e abordar a migração de recursos virtuais.

Pereira et al. [Pereira et al. 2019] propuseram uma política para alocação de recursos em nuvens veiculares em ambiente rodoviário. O objetivo da política é maximizar a disponibilidade de recurso na nuvem veicular. Especificamente, os veículos devem cooperar para criar um *pool* de recursos, a fim de conhecer os recursos disponíveis, que serão abastecidos pelos veículos e o conjunto de nevoeiros nos quais eles gerarão os recursos e a alocação de serviços. O uso do paradigma de computação em névoa permite que os recursos fiquem próximos aos veículos, permitindo que os recursos da Fog sejam agregados aos recursos fornecidos pelos veículos, aumentando assim a quantidade de serviços que podem ser oferecidos. No entanto, os autores não consideram apenas os recursos dos veículos para alocação dos serviços, além de considerar um cenário de rodovia que possui uma mobilidade mais previsível que um cenário urbano.

Hattab et al. [Hattab et al. 2019] introduziram um algoritmo com complexidade de tempo polinomial para alocação de tarefas em VCs com diferentes recursos computacionais. O algoritmo proposto tem dois estágios, no primeiro estágio o algoritmo classifica as tarefas de acordo com a proporção entre tempo de conclusão e espera. No segundo estágio seleciona um subconjunto de tarefas com a menor proporção e depois resolve uma sequência de Programas Lineares. Este trabalho formula o problema de gargalo de alocação, cujo objetivo é minimizar o tempo de conclusão das tarefas alocadas nas VCs disponíveis. Porém, os autores desconsideram a mobilidade veicular para formação das VCs, com as VCs sendo estacionárias, além do algoritmo considerar apenas uma VC.

Wang et al. [Wang et al. 2018] propuseram um algoritmo para alocação de tarefas que usa como base os conceitos do Problema das Múltiplas Mochilas. Este trabalho contextualiza o problema de Descarregamento de Computação e consideram que cada usuário pagará pelas tarefas de computação descarregadas de acordo com seus tamanhos, então o objetivo é maximizar os lucros totais do descarregamento de computação da pers-

pectiva da infraestrutura. Assim, é proposto um algoritmo *Branch-and-Bound* para obter a solução ideal, e um método heurístico *Greedy* para obter um desempenho aproximado com sobrecarga computacional muito menor. No entanto, os algoritmos se mostram custosos computacionalmente por computar a solução ótima para cada mochila em uma etapa e só depois utilizar essa informação para alocar de fato as tarefas nas diferentes VCs.

Nabi et al. [Nabi et al. 2017] apresentaram um esquema para alocação de tarefas em nuvens veiculares que utiliza conceitos do Problema da Mochila. Os autores fornecem um esquema que resolve de forma aproximada a alocação para uma única tarefa em tempo polinomial, além de fornecer uma solução gulosa com a mesma finalidade. Além disso, estendem o algoritmo para resolver de forma aproximada o problema de alocação para *n* tarefas. Porém, os autores consideram um ambiente *offline* em que a instância do problema está completamente disponível e é conhecida antes do início da simulação. Assim, os algoritmos necessitam de conhecimento prévio das tarefas que serão alocadas, não tendo desempenho satisfatório em um ambiente de tomada de decisão em tempo real.

Com base na análise dos trabalhos relacionados é possível observar que os trabalhos existentes não consideram que as tarefas chegam no sistema de forma dinâmica e independente, ou ainda que a formação das VCs seja dinâmica para alocar as tarefas em tempo real. Além disso, é importante considerar que as VCs sejam gerenciadas por um controlador, p.e., implantado em um nó da computação em nevoa, que seja alcançável por meio da infraestrutura de comunicação.

3. Alocação de Tarefas em Nuvens Veiculares

Esta seção descreve o mecanismo MORFEU, o qual considera uma abordagem centralizada para selecionar as melhores nuvens veiculares para alocação das tarefas no cenário veicular. MORFEU considera uma abordagem de otimização combinatória para seleção do conjunto ótimo de tarefas para serem alocadas em tempo real em cada uma das nuvens veiculares disponíveis.

3.1. Modelo de Sistema

A Figura 1 apresenta o modelo do sistema em duas dimensões, onde a dimensão 1 apresenta o cenário em si, com os veículos e RSUs, e a dimensão 2 representa como se dá a interação entre esses componentes do sistema. Nesse cenário, os veículos em movimento oportunamente formam nuvens veiculares (VCs) por meio de clusterizações. Alguns desses veículos, que não fazem parte de nenhuma VC, necessitam processar algum dado mas seus recursos computacionais não são capazes de executar tal tarefa. Assim, tais veículos solicitam à uma entidade na borda da rede, denominada de *VC Controller* que executa o MORFEU, recursos para executar sua tarefa.

Considera-se um cenário composto por x veículos, onde cada veículo u_i possui uma identificação individual ($i \in [1, x]$) e é equipado com OBU (*On Board Unit*) que possibilita a comunicação tanto entre veículos (V2V, de *Vehicle-to-Vehicle*) quanto entre veículos e RSU (V2I, de *Vehicle-to-Infrastructure*). As RSUs fornecem comunicação ao *VC Controller*, que tem como função decidir o grupo destes veículos capaz de executar uma tarefa que exige certo poder computacional, baseado no conjunto de VCs disponíveis no cenário. As RSUs coletam informações dos veículos em tempo real e quando um veículo requisita recursos para alocar sua tarefa, representado pelo circulo vermelho

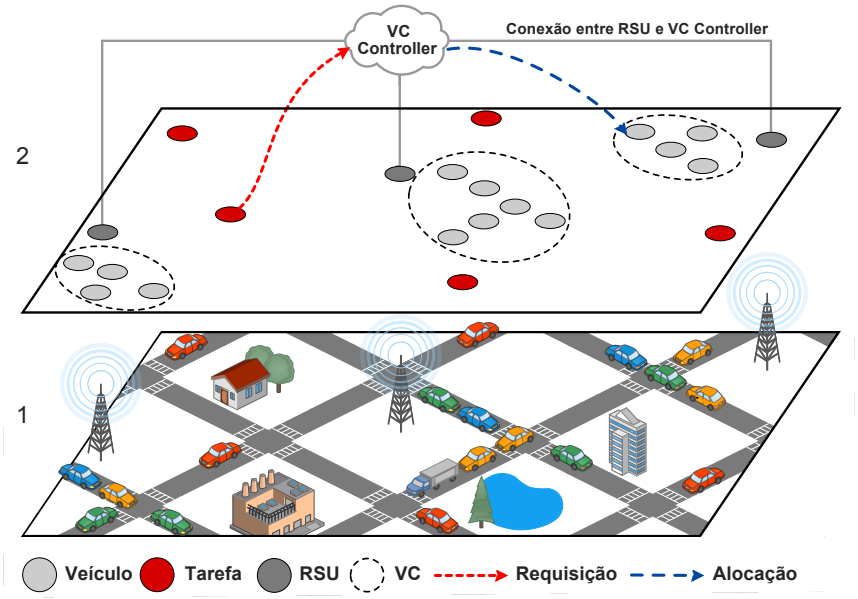


Figura 1. Representação da Arquitetura do Sistema

na dimensão 2 da Figura 1, o *VC Controller* tem conhecimento de qual nuvem veicular (circulo pontilhado) possui recursos suficientes para alocar essa tarefa.

O *VC Controller* executa o MORFEU para atender a requisição por recursos (seta vermelha) e alocar a tarefa (seta azul) em alguma das VCs disponíveis, de forma mais rápida possível e consumindo o máximo de recursos computacionais da VC. Levando em consideração o fato da maximização do consumo de recursos, mais de uma tarefa deve ser alocada em uma mesma VC se assim for possível.

3.2. Definição do Problema

Dado um conjunto de tarefas T com $\{k = 1, \ldots, n\}$, sendo cada uma representada como uma tupla $\langle id_k, p_k, g_k \rangle$, e cada tarefa possui identificação id_k, um peso p_k que representa a quantidade de recursos que necessita para ser alocada (Milhões de Instruções - MI) e uma recompensa por alocação g_k. Após um processo de clusterização, baseado em algum critério pré-definido, há a formação de grupos entre os veículos que são nomeados aqui de nuvens veiculares (VCs) [Hagenauer et al. 2019]. Sendo assim, tem-se como objetivo alocar tais tarefas para serem executadas nas diversas VCs que se formam no cenário de comunicação veicular. Uma VC $v_j \in V = \{v_1, \ldots, v_m\}$ é composta por veículos capazes de compartilhar até 3 tipos de recursos computacionais: $i)$ largura de banda, $ii)$ processamento e/ou $iii)$ armazenamento [Meneguette et al. 2018].

Define-se então o Problema da Alocação de Tarefas em Nuvem Veicular (PATNV), que pode ser modelado como um Problema da Mochila 0/1 (PM0/1) por conta de suas características semelhantes. No PM0/1, existem a itens com peso b e valor agregado c, e uma mochila de capacidade W. O objetivo é maximar o valor dos itens guardados na mochila não ultrapassando sua capacidade e considerando que um item só pode ser guardado uma única vez (0 para não alocado, 1 para alocado). O PM0/1 é um problema de otimização combinatória e não tem resolução conhecida em tempo polinomial, ou seja, é um problema $\mathcal{NP}$-difícil com complexidade de tempo pseudo-polinomial $\mathcal{O}(a \cdot W)$,

sendo a o número de itens e W a capacidade da mochila [Martello and Toth 1990].

No PATNV, define-se que as tarefas em T são os itens e as nuvens veiculares $v_j \in V$ são as mochilas. O recurso total de cada nuvem veicular Ω_j é a soma dos recursos compartilhados ω_i por cada veículo u_i ($i = 1, \ldots, x$) que faz parte dessa nuvem, sendo $\sum_{i=1}^{x} \omega_i, \forall u_i \in v_j$. A recompensa g_k de alocação de uma tarefa id_k cresce uniformemente como o dobro de seu peso p_k. Desta forma, atribuísse uma recompensa maior para tarefas que são mais pesadas, ou seja, tarefas que exigem maior poder computacional para serem resolvidas. Com isso, se tem um problema de otimização combinatória onde busca-se maximar a recompensa por alocar tarefas ao passo que o consumo de recursos computacionais veiculares também seja maximizado. Desta forma, o PATNV é formalmente definido como:

$$\begin{aligned} &\text{maximizar} \sum_{k=1}^{n} g_k t_k \\ &\text{sujeito a} \sum_{k=1}^{n} p_k t_k \leq \Omega_j, \text{e } t_k \in \{0, 1\} \end{aligned} \tag{1}$$

onde,

$$t_k = \begin{cases} 1 & \text{se a tarefa } k \text{ foi adicionada à nuvem veicular,} \\ 0 & \text{caso contrário} \end{cases} \tag{2}$$

3.3. MORFEU

Conforme discutido, o problema de alocar tarefas em nuvens veiculares é $\mathcal{NP}$-difícil [Sorkhoh et al. 2019]. Assim, para contornar a complexidade de resolução de tal problema, utilizamos uma abordagem baseada em programação dinâmica que resolve o PM0/1 em tempo pseudo-polinomial, conforme já discutido. Como as tarefas devem ser alocadas eficientemente em mais de uma nuvem, se for o caso, necessita-se de uma abordagem de considere tal cenário. Porém, a adição de mais nuvens ao problema implica no aumento da complexidade de resolução do mesmo.

Existe na literatura uma abordagem que considera múltiplas mochilas, conhecido como Problema de Múltiplas Mochilas 0/1 (*Multiple 0/1 Knapsack Problem - MKP*), porém já se foi demonstrado que utilizar programação dinâmica para esse problema é impraticável por conta do aumento de sua complexidade [Martello and Toth 1990]. Com isso, outras soluções buscam listar todas as possibilidades possíveis e ir cortando a árvore de soluções baseando-se em relaxações, ou seja, em limites superiores e inferiores do problema. Assim, além de algumas técnicas chegarem apenas à soluções aproximadas, outras são dispendiosas computacionalmente para soluções exatas do problema.

Levando isso em consideração, objetiva-se exigir o mínimo de poder computacional para resolução do PATNV em um tempo suficientemente pequeno. Desta forma, o MORFEU considera um algoritmo de programação de dinâmica que soluciona o PM0/1 tradicional, porém passando a considerar múltiplas nuvens veiculares, uma por rodada. Assim, sua complexidade permanece pseudo-polinomial com $\mathcal{O}(\max\{\Omega_j\} + (n \cdot \Omega_j))$, sendo n o número de tarefas e Ω_j a capacidade da VC $v_j \in V$. Como pré-requisito para o MORFEU operar, tem-se a necessidade da identificação das nuvens veiculares no cenário.

O MORFEU não depende de uma técnica específica para clusterização, a única exigência é que as nuvens veiculares sejam identificadas e fornecidas. Sendo assim, como objetiva-se primariamente avaliar a política de alocação de recursos, é utilizado neste trabalho uma técnica de clusterização bastante conhecida e difundida na literatura, o DBSCAN (*Density Based Spatial Clustering of Application with Noise*) [Ester et al. 1996].

O DBSCAN identifica clusters baseando-se na densidade espacial dos indivíduos[1], possibilitando a identificação de *clusters* de diferentes tamanhos em uma base de dados contendo dados de espaço métrico, além de possibilitar a passagem de parâmetros como raio eps e número mínimo de membros por *cluster* $min\_samples$. A ideia chave do método DBSCAN é que, para cada ponto de um *cluster*, a vizinhança para um dado raio contém, no mínimo, certo número de pontos, ou seja, a densidade na vizinhança tem que exceder um limiar. A complexidade de tempo do DBSCAN é $\mathcal{O}(d^2)$, sendo d o número de pontos à serem clusterizados. Assim, o DBSCAN se mostrou suficiente para nossas análises.

Desta forma, conforme mostrado no Algoritmo 1, os conjuntos de tarefas T e nuvens veiculares V são fornecidas ao MORFEU. Primariamente, enquanto existirem tarefas não alocadas, o sistema irá operar (Linha 4). Se o conjunto V não possuir nenhum item, o sistema irá parar pois não haverá mais recursos no sistema (Linha 12). Caso contrário, o maior item $\Omega_j \in V$ é escolhido, retirado de V, adicionado à chamada do procedimento PROBLEMAMOCHILA que recebe como parâmetros o conjunto de tarefas T e o Ω_j atual selecionado, e retorna as tarefas que foram alocadas em Ω_j (Linhas 6-8). Conforme as tarefas são alocadas, estas são removidas de T e adicionadas ao conjunto de tarefas alocadas $\mathbb{S}$, as tarefas que restaram em T serão realocados (Linhas 9-10). Se todas as tarefas forem alocadas nessa rodada, o mecanismo encerra sua operação. Em resumo, o MORFEU retorna a alocação ótima para Ω_j dado o conjunto T atual.

Algoritmo 1: MORFEU

Entrada: conjuntos de tarefas T e nuvens veiculares V
Saída: tarefas alocadas $\mathbb{S}$

```
1  início
2      T ← {1, ..., n}
3      V ← {1, ..., m}
4      enquanto T ≠ ∅ faça
5          se V ≠ ∅ então                    ▷ há recurso Ω para alocar tarefas
6              Ω_j ← max{V}
7              V ← V \ {Ω_j}
8              S' ← ProblemaMochila({T, Ω_j})
9              T ← T \ S'
10             S ← S'
11         senão                                  ▷ sistema sem recursos
12             break
13     retorna S
```

O Algoritmo 2 mostra como o procedimento PROBLEMAMOCHILA funciona. Tal procedimento recebe como entrada o conjunto de tarefas T e a capacidade Ω_j da VC

[1]Neste caso, são os veículos e suas coordenadas fornecidas pelo GPS (*Global Positioning System*).

$v_j \in V$ selecionada no Algoritmo 1. Seu retorno conterá um conjunto S com as tarefas que foram alocadas em v_j de capacidade Ω_j. Então, uma matriz de dimensão $n \times m$, onde $n = |T|$ e $m = \Omega_j$ (Linha 10), é construída para resolução do problema. A ideia principal é guardar os cálculos que são realizados em um $m' < m$ para reutilização de tais informações dinamicamente. Se o peso de um item $i \in T$ couber na mochila de tamanho j, seu valor g_i será guardado na célula $K[i][j]$ (Linha 16). Caso contrário, o valor já calculado da última linha será o melhor atual e será considerado (Linha 18), daí o motivo de $K[i][j] = 0$ quando $i = 0$ e $j = 0$ (Linha 14). Se no futuro for identificado que um outro item $i' \neq i$ cabe na mochila e possui valor maior, o item i será substituído por i' se a soma $i + i'$ não couber na mochila. Mais informações sobre o problema da mochila 0/1 podem ser consultadas em [Martello and Toth 1990].

Algoritmo 2: PROBLEMAMOCHILA

Entrada: conjunto de tarefas T e capacidade Ω da nuvem veicular $v_j \in V$
Saída: tarefas alocadas S

```
1  início
2      K ← ∅
3      pes ← ∅
4      val ← ∅
5      para cada t ∈ T faça
6          pes.append(t[1])                          ▷ peso da tarefa
7          val.append(t[2])                          ▷ recompensa da tarefa
8      n ← |T|
9      m ← Ω
10     K[n + 1][m + 1]
11     para i até n + 1 faça                                  ▷ tarefas
12         para j até m + 1 faça                  ▷ capacidade da VC v_j
13             se i = 0 ou j = 0 então                        ▷ caso base
14                 K[i][j] ← 0
15             senão se pes[i − 1] ≤ j então
16                 K[i][j] ← max(val[i − 1] + K[i − 1][j − pes[i − 1]], K[i − 1][j])
17             senão
18                 K[i][j] ← K[i − 1][j]
19     S ← ∅
20     para n até i faça          ▷ recupera os ids das tarefas alocadas
21         i ← i − 1
22         se K[n][Ω] ≤ 0 então
23             break
24         se K[n][Ω] = K[i − 1][Ω] então
25             continue
26         senão
27             S.append([i − 1])
28             S ← S − val[i − 1]
29             Ω ← Ω − pes[i − 1]
30     retorna S
```

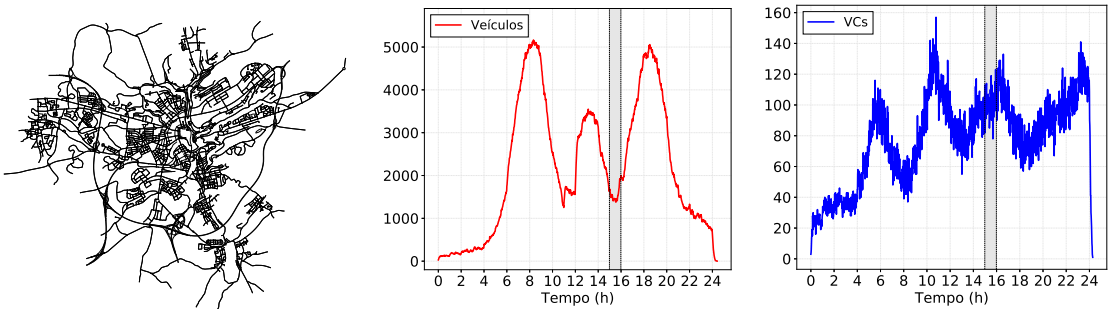
4. Avaliação

Esta seção descreve a metodologia e métricas utilizadas para avaliar a eficiência do mecanismo MORFEU em termos de ganho de alocação, tarefas alocadas, tarefas não alocadas e utilização de recursos das VCs. Foram realizadas simulações com um *trace* de mobilidade de Luxemburgo para analisar a eficiência do MORFEU comparado-o à outras abordagens de alocação de tarefas.

4.1. Metodologia

Os experimentos foram realizados no Simulador de Mobilidade Urbana (*Simulation of Urban MObility – SUMO*), na versão $0.32.0$[2]. Os algoritmos foram implementados na linguagem Python e conectados ao SUMO através da interface TraCI[3]. O *trace* de mobilidade utilizado foi o *Luxembourg SUMO Traffic (LuST)* [Codeca et al. 2017], exibido na Figura 2(a). O tráfego rodoviário é gerado com base nas demandas estatísticas do tráfego derivado de censos, enquanto o SUMO simula o comportamento individual de veículos com uma visão microscópica [Higuchi et al. 2018].

O *trace* contém a mobilidade veicular de 24 horas com até 5.000 veículos em seus horários de pico, conforme pode ser observado na Figura 2(b). Para este trabalho optou-se por considerar apenas uma faixa com densidade veicular mais baixa, para demonstrar que o mecanismo MORFEU opera satisfatoriamente mesmo nesse cenário mais desafiador, ou seja, com menor quantidade de recursos disponíveis. Para isso, selecionou-se o intervalo entre 15h e 16h, conforme área sombreada na Figura 2(b). Por outro lado, a Figura 2(c) apresenta o número de VCs identificadas no *trace*, a qual segue a densidade veicular, mas eles são inversamente proporcionais. Conforme aumento da densidade veicular, maior é a chance de veículos se juntarem à *clusters*.



(a) Cenário LuST (b) Veículos no cenário (c) VCs identificadas

Figura 2. Dados do cenário veicular utilizado na simulação

Nesta avaliação, considerou-se que o *deadline* (tempo mínimo para execução no sistema) para cada tarefa é igual a 1, ou seja, uma vez que a tarefa é alocada, seu sucesso na alocação já é contabilizado. O intervalo de clusterização foi de 60 segundos. Então, no momento em que o agrupamento acontece, as tarefas são geradas e o MORFEU as aloca nas VCs disponíveis. A taxa de chegada de tarefas no sistema segue uma distribuição de Poisson com uma média de $\lambda = 25$, pois suas ocorrências são independentes umas das outras [Kadhim and Seno 2019]. O raio de comunicação considerado foi de 100 metros

[2]`http://sumo.dlr.de/`
[3]`http://sumo.dlr.de/docs/TraCI.html`

($eps = 100$). O número mínimo de membros por VC foi definido em 2. Além disso, variou-se o peso atribuído às tarefas em $\mu = \{1, 25, 50, 100, 200 \text{ e } 300\}$, o ganho de alocação em $2 \times \{1, 25, 50, 100, 200 \text{ e } 300\}$, e o número de recursos compartilháveis por cada veículo em $1, 2$ e 3 (denotado como configuração 1, 2 e 3, respectivamente). Aumentar a quantidade de recursos por veículo é importante para observar o impacto que o compartilhamento de diferentes unidades de recursos traz para o problema da alocação como um todo. A Tabela 1 resume os parâmetros usados na simulação.

Tabela 1. Parâmetros de simulação

Parâmetro	**Valor**
Faixa de comunicação	100 m
Ocorrência de tarefas	Distribuição de Poisson
Número médio de tarefas (λ)	25
Peso médio das tarefas	1, 25, 50, 100, 200, 300
Ganho médio das tarefas	$2 \times \{1, 25, 50, 100, 200, 300\}$
Quantidade de recursos	1, 2, 3
Intervalo de clusterização	60 s
Algoritmo de clusterização	DBSCAN
| *eps*	100
| *min_samples*	2
Número de veículos	$1500 \sim 2000$
Tempo de simulação	1 h de LuST Scenario
Área do cenário	155.95 km^2

As métricas utilizadas para avaliação foram: $i)$ ganho de alocação representa a quantidade de recompensa obtida pela alocação de tarefas, calculado conforme Eq. 3; $ii)$ serviços atendidos refere-se à porcentagem de tarefas alocadas com sucesso, calculado conforme Eq. 4; e $iii)$ utilização de recursos que denota a porcentagem de recurso de nuvem usado, calculado conforme Eq. 5.

$$ganho = \sum_{i=1}^{n} g_i \mid g_i \in \mathbb{S} \tag{3}$$

$$servicos\_atendidos = \frac{\{\sum_{i=1}^{n} t_i \mid t_i \in \mathbb{S}\} \cdot 100}{|T|} \tag{4}$$

$$utilizacao = \frac{\{\sum_{i=1}^{n} p_i \mid p_i \in \mathbb{S}\} \cdot 100}{\sum_{j=1}^{m} \Omega_j} \tag{5}$$

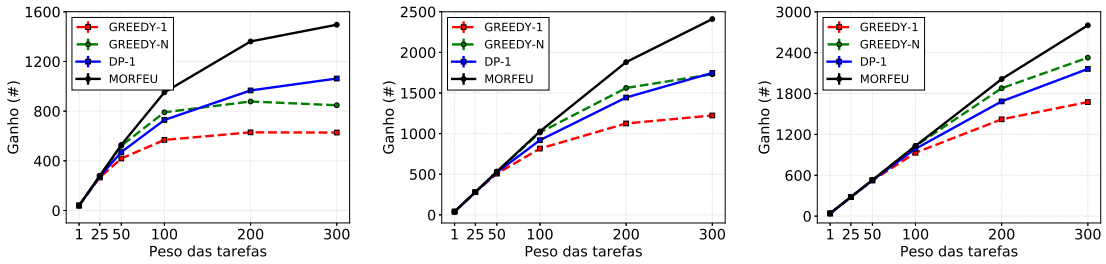
O MORFEU foi comparado à três abordagens para alocação de tarefas em VC. Duas delas são estratégias gananciosas (*Greedy Algorithm*) que classificam as listas de tarefas em ordem não crescente de peso e fazem a seleção para alocar com base nessa classificação, mas a primeira considera apenas uma VC (GREEDY-1) e a outra considera todas as VCs disponíveis (GREEDY-N), sendo esta baseada em [Nabi et al. 2017]. A terceira estratégia é baseada em programação dinâmica, chamada DP, porém considera

apenas uma VC para alocação. Todas as abordagens usam primeiramente a VC de maior capacidade para alocar o maior número possível de tarefas.

4.2. Análise dos Resultados

A Figura 3 apresenta o ganho por alocação das tarefas com diferentes pesos das tarefas e quantidade de recursos disponíveis para os mecanismos de alocação avaliados. Com base na análise dos resultados é possível observar que o mecanismo de alocação MORFEU possui maior ganho de alocação nas configurações analisadas. Ou seja, o fato de considerar todas as VCs disponíveis e ir selecionando sempre a de maior capacidade resulta na alocação de mais tarefas por rodada e, consequentemente, aumento no ganho médio. O GREEDY-1 tem pior desempenho mesmo fazendo a escolha gulosa com base no valor das tarefas, isso ocorre porque o algoritmo termina assim que a verificação do valor da tarefa já computada é maior que o da próxima tarefa verificada. O mesmo acontece com o GREEDY-N, porém sua estratégia gulosa considera todas as VCs disponíveis. O DP verifica o peso e valor da tarefa ao mesmo tempo, assim encontra a solução ótima para cada configuração, porém apenas na VC de maior capacidade.

Pode ser observado também que à medida que o número de recursos compartilháveis nos veículos aumenta, os algoritmos tendem a se comportar de maneira semelhante (Figuras 3(b) e 3(c)). Isso acontece em função da média de tarefas ser a mesma para todas as configurações, ou seja, aumentar a quantidade de recursos disponíveis sem aumentar o nível de consumo de tais recursos é menos desafiador do que consumir recursos em um cenário de baixa disponibilidade de recursos (Figura 3(a)). Quanto menor é o peso das tarefas, menor será o ganho médio por alocá-las. De modo geral, o MORFEU empregou melhoria de 19.40%, 19.65% e 36.32% em relação ao GREEDY-N, DP e GREEDY-1, respectivamente.

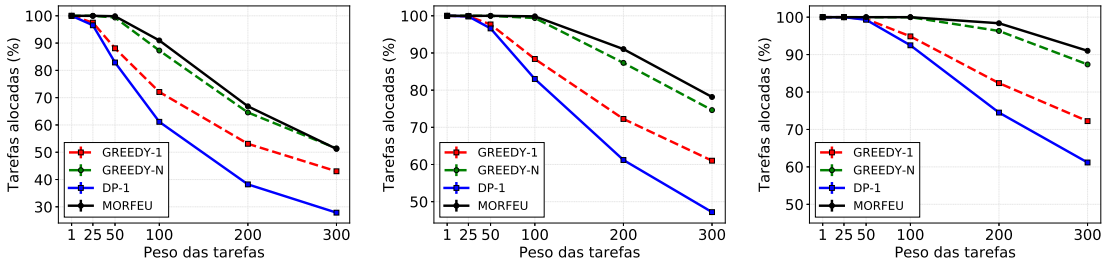


(a) Configuração 1 (b) Configuração 2 (c) Configuração 3

Figura 3. Ganho de alocação para diferentes pesos das tarefas e quantidade de recursos disponíveis

A Figura 4 exibe a taxa de tarefas que foram alocadas com sucesso variando os pesos das tarefas e quantidade de recursos disponíveis. Como pode ser visto, o MORFEU consegue alocar mais tarefas em todas as configurações avaliadas. Especificamente, o MORFEU alocou tarefas 1.28%, 14.76% e 29.05% a mais que o GREEDY-N, GREEDY-1 e DP, respectivamente. O GREEDY-1 e GREEDY-N conseguem alocar mais tarefas que o DP em função de selecionarem as tarefas com base em suas estratégias gulosas, onde nem sempre essas tarefas fazem parte do conjunto ótimo de alocação. Ou seja, tais abordagens consideram primariamente o valor das tarefas enquanto houver recurso disponível. MORFEU e GREEDY-N se destacam por alocarem 100% das tarefas com

pesos variando de 1 a 25, com leve vantagem para o MORFEU quando pesos maiores são considerados. O sucesso de alocação de todas as abordagens aumenta ao passo que a quantidade de recursos disponíveis aumenta, conforme pode ser observado nos resultados da Figura 4.



(a) Configuração 1 (b) Configuração 2 (c) Configuração 3

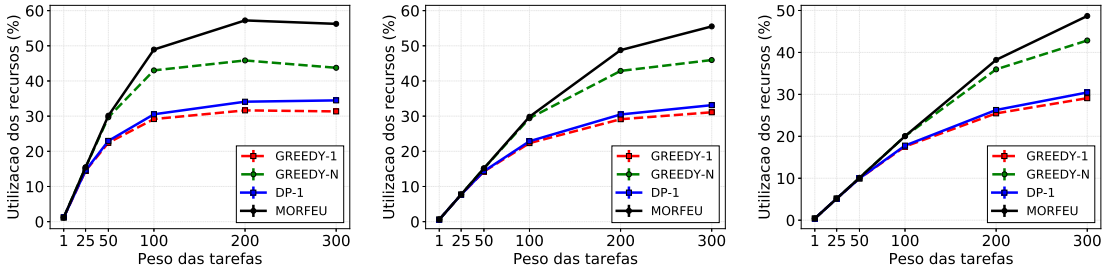
Figura 4. Tarefas atendidas para diferentes pesos das tarefas e quantidade de recursos disponíveis

A Figura 5 apresenta a utilização dos recursos disponíveis nas VCs identificadas no cenário de simulação. Como pode ser visto, o MORFEU faz melhor uso dos recursos disponíveis por conta de ficar operando enquanto existirem tarefas e/ou enquanto existir recursos computacionais nas VCs passíveis de serem alocados. Não apenas isso, mas o fato do MORFEU empregar uma abordagem de otimização combinatória que busca utilizar o máximo dos recursos disponíveis fazendo todas as verificações possíveis para o conjunto de tarefas fornecido. Logo, a abordagem de programação dinâmica retorna uma alocação ótima para um conjunto de tarefas e uma única VC, mas o MORFEU estende essa alocação para cada uma das VCs em cada rodada de alocação. Quando o peso das tarefas é menor e todas elas são alocadas, a taxa de utilização dos recursos é menor em relação à tarefas com peso maior.

O GREEDY-N também tenta estender a alocação das tarefas para todas as VCs em uma rodada por vez, porém sua estratégia gananciosa de considerar primariamente o ganho da tarefa faz com que as tarefas mais "valiosas", e consequentemente as mais pesadas também, sejam alocadas primeiro. Isso faz com que os recursos das VCs não sejam utilizados de forma ótima. Tanto DP quanto GREEDY-1 possuem os piores resultados em todas as configurações por considerarem apenas a VC de maior capacidade para alocar todas as tarefas geradas, ou seja, em uma única rodada mesmo com outras VCs disponíveis. Assim, o MORFEU faz melhor uso dos recursos, diminuindo o desperdício de recursos em 13.22%, 32.98% e 35.16% em relação ao GREEDY-N, DP e GREEDY-1, respectivamente.

5. Conclusão

A popularização e crescimento do setor automobilístico tem contribuído para o surgimento de novas direções de pesquisa. Uma delas é a computação em nuvem veicular, que busca criar nuvens veiculares e utilizar os recursos computacionais presentes nos veículos para execução de tarefas para aplicações diversas de ITS. Assim, tarefas podem ser executadas mais próximo do usuário veicular, diminuindo latência por exemplo. Pensando nisso, esse artigo apresentou um mecanismo denominado de MORFEU, o qual é baseado em otimização combinatória para alocação eficiente de tarefas em nuvens veiculares



(a) Configuração 1 (b) Configuração 2 (c) Configuração 3

Figura 5. Utilização dos recursos para diferentes pesos das tarefas e quantidade de recursos disponíveis

a fim de aproveitar o máximo dos recursos disponíveis em tais nuvens. Os resultados obtidos mostram que o mecanismo proposto cumpre seus objetivos no sentido de maximizar o consumo de recursos ao passo que aloca um número considerável de tarefas em comparação com outras abordagens.

Como trabalhos futuros, pretende-se considerar localização e *deadlines* mais desafiadores nas tarefas. Assim, além de maximizar o ganho por alocar e o consumo de recursos nas VCs, pretende-se alocar uma tarefa em uma VC se e somente se essa VC conseguir atender requisitos como latência e *deadline* da tarefa. Além disso, computar a evolução das VCs ao longo do tempo é importante para construir um conhecimento espaço-temporal do cenário e aumentar a disponibilização dos recursos para alocação das tarefas. Considerar o problema de múltiplas mochilas com a adição de restrições de latência e *deadline* das tarefas também é algo interessante para se analisar.

Agradecimentos

Os autores agradecem o apoio financeiro concedido pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) por meio dos processos de nº 2015/24494-8 e 2018/16703-4.

Referências

Brik, B., Khan, J. A., Ghamri-Doudane, Y., and Lagraa, N. (2019). Publish: A distributed service advertising scheme for vehicular cloud networks. In *16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE.

CISCO (2019). Driving Profits from Connected Vehicles. Technical report, CISCO.

Codeca, L., Frank, R., Faye, S., and Engel, T. (2017). Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63.

Costa, J., Rosário, D., de Souza, A. M., Villas, L. A., and Cerqueira, E. (2018). Protocolo para disseminação de dados em vanets baseado em métricas de redes complexas: Um estudo de caso com sistema de gerenciamento de trânsito. In *XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

Coutinho, R. W. and Boukerche, A. (2019). Guidelines for the design of vehicular cloud infrastructures for connected autonomous vehicles. *IEEE Wireless Communications*, 26(4):6–11.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, pages 226–231.

Hagenauer, F., Higuchi, T., Altintas, O., and Dressler, F. (2019). Efficient data handling in vehicular micro clouds. *Ad Hoc Networks*, 91:101871.

Hattab, G., Ucar, S., Higuchi, T., Altintas, O., Dressler, F., and Cabric, D. (2019). Optimized assignment of computational tasks in vehicular micro clouds. In *2nd International Workshop on Edge Systems, Analytics and Networking*, pages 1–6. ACM.

Higuchi, T., Dressler, F., and Altintas, O. (2018). How to keep a vehicular micro cloud intact. In *IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE.

Kadhim, A. J. and Seno, S. A. H. (2019). Maximizing the utilization of fog computing in internet of vehicle using sdn. *IEEE Communications Letters*, 23(1):140–143.

Martello, S. and Toth, P. (1990). *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons Ltd.

Meneguette, R. I., Boukerche, A., and Pimenta, A. H. (2018). Avarac: An availability-based resource allocation scheme for vehicular cloud. *IEEE Transactions on Intelligent Transportation Systems*.

Meneguette, R. I., Rodrigues, D. O., da Costa, J. B. D., Rosario, D., and Villas, L. A. (2019). A virtual machine migration policy based on multiple attribute decision in vehicular cloud scenario. In *IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

Nabi, M., Benkoczi, R., Abdelhamid, S., and Hassanein, H. S. (2017). Resource assignment in vehicular clouds. In *IEEE International Conference on Communications (ICC)*, pages 1–6. Ieee.

Pereira, R., Lieira, D., da Silva, M., Pimenta, A., da Costa, J. B. D., Rosario, D. L., and Meneguette, R. I. (2019). A novel fog-based resource allocation policy for vehicular clouds in the highway environment. In *IEEE 11th Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE.

Qualcomm (2018). Connecting vehicles to everything.

Sorkhoh, I., Ebrahimi, D., Atallah, R., and Assi, C. (2019). Workload scheduling in vehicular networks with edge cloud capabilities. *IEEE Transactions on Vehicular Technology*.

Thakur, A. and Malekian, R. (2019). Fog computing for detecting vehicular congestion, an internet of vehicles based approach: A review. *IEEE Intelligent Transportation Systems Magazine*, 11(2):8–16.

Wang, J., Liu, T., Liu, K., Kim, B., Xie, J., and Han, Z. (2018). Computation offloading over fog and cloud using multi-dimensional multiple knapsack problem. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE.

Yang, C., Liu, Y., Chen, X., Zhong, W., and Xie, S. (2019). Efficient mobility-aware task offloading for vehicular edge computing networks. *IEEE Access*, 7:26652–26664.

Yu, R., Zhang, Y., Gjessing, S., Xia, W., and Yang, K. (2013). Toward cloud-based vehicular networks with efficient resource management. *IEEE Network*, 27(5):48–55.