ARCANE: Algoritmo Meta-heurístico para Alocação de Tarefas em Nuvens Veiculares

Matheus S. Quessada[1,5], Douglas D. Lieira[1,2], Joahannes B. D. da Costa[3], Geraldo P. Rocha Filho[4], Robson E. De Grande[5], Rodolfo I. Meneguette[6]

[1]Departamento de Ciência da Computação – Universidade Estadual Paulista (UNESP)
São José do Rio Preto, SP – Brazil.

[2]Departamento de Informática – Instituto Federal de São Paulo (IFSP)
Catanduva, SP – Brasil.

[3]Departamento de Ciência da Computação, Universidade de Campinas (UNICAMP)
Campinas, SP – Brasil.

[4]Departamento de Ciência da Computação, Universidade de Brasília (UnB)
Brasília, DF – Brasil.

[5]Departamente de Ciência da Computação – Brock University (BrockU)
St. Catharines, ON – Canada.

[6]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
São Carlos, SP – Brasil.

{matheus.quessada, douglas.lieira}@unesp.br, joahannes@lrc.ic.unicamp.br,
geraldof@unb.br, rdegrande@brocku.ca, meneguette@icmc.usp.br

Abstract. *Advancements in Intelligent Transport Systems (ITS) help mitigate several transportation challenges that bring substantial socio-economic issues. Network dynamics in which the ITS operates, high mobility of vehicles, and constant changes in topology make the problem of resource allocation and tasks even more complex. Therefore, we introduce ARCANE, a meta-heuristic algorithm for allocating tasks in vehicular clouds. ARCANE is a bio-inspired approach based on the Bat Algorithm (BAT). Its goal consists of optimizing the search process to provide sub-optimal solutions when allocating resources and tasks in a vehicular cloud. Compared to other recent resource allocation approaches, ARCANE proved to be effective in allocating tasks, taking better advantage of the vehicular clouds resources in all scenarios.*

Resumo. *O avanço dos Sistemas de Transporte Inteligentes (ITS) vem para auxiliar a resolver problemas de tráfego, que atualmente geram problemas socioeconômicos. Entretanto, devido a dinamicidade da rede em que os ITS atuam, a alta mobilidade dos veículos e a constante mudança de topologia faz com que o problema de alocação de recursos e tarefas se tornem ainda mais desafiador. Diante desse desafio, é proposto o ARCANE, um algoritmo meta-heurístico para alocação de tarefas em nuvens veiculares. O ARCANE é um método bioinspirado baseado no Algoritmo do Morcego (BAT). O objetivo do ARCANE é otimizar o processo de busca para fornecer soluções subótimas no processo de alocação de recursos e tarefas em uma nuvem veicular. Quando comparado com*

outas soluções da literatura, o ARCANE mostrou ser efetivo em alocar tarefas, aproveitando melhor os recursos das nuvens veiculares em todos os cenários.

1. Introdução

Com a tendência dos carros sustentáveis, elétricos e autônomos, o avanço do mercado automobilístico vem gerando bilhões de dólares em investimentos no setor [Assis 2021]. O crescente aumento no número de veículos no mundo aliado com a má estruturação que os grandes centros urbanos possuem, causam diversos problemas socioeconômicos que merecem destaque, tais como congestionamentos nas vias, problemas de saúde causado pelo estresse no trânsito, alta emissão de CO_2, além de perdas monetárias e frustrações gerados pelos engarrafamentos [Pereira et al. 2019]. Esses problemas afetam não apenas a economia local, mas também a qualidade de vida da população da região [Meneguette et al. 2021].

Uma das soluções viáveis para auxiliar a sanar tais problemas é por meio do uso dos Sistema de Transporte Inteligente (ITS, do inglês *Intelligent Transportation System*). No ITS, os veículos são equipados com sensores e dispositivos de comunicação para que possam se comunicar entre si, compartilhar recursos e serviços e realizar trocas de informações para resolver os desafios relacionados ao transporte nos centros urbanos [Pereira et al. 2020]. A comunicação realizada entre os veículos e os dispositivos são efetuadas por meio de uma Rede Ad-Hoc Veicular (VANET, do inglês *Vehicular Ad Hoc Networks*). Nas VANETs, a comunicação pode ser realizada tanto de veículo-para-veículo (V2V, do inglês *Vehicle-to-Vehicle*) quanto veículo-para-infraestrutura (V2I, do inglês *Vehicle-to-Infrastructure*) [Correa et al. 2014, Rocha Filho et al. 2020].

Nesse contexto, as Nuvens Veiculares (VCs, do inglês *Vehicular Clouds*) [Meneguette and Boukerche 2017] visam realizar uma cooperação eficiente na comunicação, alocação e compartilhamento de recursos e tarefas nas VANETs. As VCs são formadas por um conjunto de veículos e dispositivos que compartilham recursos computacionais através do paradigma de nuvem com características de qualidade de serviço (QoS, do inglês *Quality of Service*), recursos sob demanda e escalabilidade. O foco das VCs é alocar excesso de recursos que estão na borda da rede, sendo eles processamento, armazenamento e comunicação, de forma mais rápida quando comparado com a nuvem [Zhang et al. 2015, Meneguette et al. 2021]. Diversos desafios são encontrados nas VANETs devido ao fato da dinamicidade da rede. A alta mobilidade dos veículos e a constante mudança de topologia faz com que o processo de alocação de tarefas de forma eficiente seja desafiador e não trivial de ser resolvida [Correa et al. 2014, Meneguette et al. 2021].

Os algoritmos meta-heurísticos surgem como uma estratégia para resolver os problemas de alocação de recursos e tarefas nas VANETs [Midya et al. 2018, Liao et al. 2017, Feng et al. 2018, Jayswal 2020, Lieira et al. 2021, Demir et al. 2018]. Os algoritmos meta-heurísticos enquadram-se na categoria de otimização com base estocástica e não determinística, sendo que dado uma determinada entrada, o comportamento do algoritmo é realizado de forma aleatória para cada execução. Esses algoritmos são divididos em algoritmos de solução única e baseados em população [Katoch et al. 2020]. O interesse nesse tipo de solução, considerada não convencional, vem crescendo nos últimos anos. Além disso, vem sendo aplicado para

resolução de problemas em grandes amostras e para algoritmos convencionais que não conseguem atuar de forma satisfatória [Faris et al. 2016, Vikhar 2016], os quais são explorados nesta pesquisa.

Com isso em mente, este artigo apresenta o ARCANE, um **A**lgoritmo meta-heu**R**isti**C**o para **A**locação de Tarefas em **N**uvens V**E**iculares. Para tanto, o ARCANE foi modelado com base no Algoritmo do Morcego (BAT, do inglês *Bat Algorithm*) e considera o comportamento de caça dos micro-morcegos para encontrar sua presa. Nesse cenário, a presa corresponde a uma VC e o morcego corresponde a tarefa, i.e., para que a tarefa seja alocada, o morcego tem que chegar a sua presa. Dessa forma, o ARCANE realiza o cálculo do valor do *fitness*[1] das tarefas até encontrar a melhor solução dentro da população de morcegos para alocá-las nas VCs. Esta pesquisa apresenta duas contribuições chaves:

- **C1 -** Desenvolvimento de um algoritmo meta-heurístico para alocação de tarefas em VCs;
- **C2 -** Alocação eficiente de tarefas para maximizar o uso dos recursos disponíveis.

O restante deste artigo está organizado da seguinte forma. Na Seção 2, são apresentados trabalhos relacionados a alocação de recursos e tarefas. Na Seção 3, são discutidos sobre a formulação do problema, o cenário da aplicação e o algoritmo desenvolvido. Na Seção 4, são apresentados e discutidos os resultados obtidos através das avaliações. Por fim, a Seção 5 apresenta as conclusões finais e diretrizes para trabalhos futuros.

2. Trabalhos Relacionados

Esta seção apresenta as soluções encontradas na literatura que tratam do problema da alocação de recursos e tarefas, além de identificar e discutir as lacunas de pesquisa que este trabalho investiga.

No trabalho de [Pereira et al. 2019] foi desenvolvido o Nancy, um algoritmo para gerenciamento e alocação de recursos em nuvens veiculares utilizando um método matemático de multicritério. O Nancy foi proposto com o método de Processamento de Hierarquia Analítica para computar o fator de influência de cada parâmetro, aplicar uma matriz de decisão e selecionar uma *Fog* para receber os recursos de um veículo. Para validação, foi realizada uma simulação com a distribuição de Poisson, sendo responsável por simular as entradas e saídas de veículos. Quando comparado com técnicas tradicionais, apresentou resultados satisfatórios, considerando as métricas de alocação, negação e bloqueio de recursos. No entanto, o algoritmo além de não utilizar tráfego real, não compartilha os recursos ociosos.

Os autores em [Nabi et al. 2017] propuseram uma solução para otimizar o problema de alocação de recursos e associação de usuários em redes heterogêneas. Para tanto, considerou as restrições de tecnologias de acessos de rádio, as preferências de usuários e as requisições de taxas de dados dos equipamentos de usuários. Para definir as preferências de usuários, foram considerados o consumo de energia dos equipamentos de usuários e a qualidade do sinal. Para esse problema, um novo algoritmo heurístico guloso de complexidade de tempo polinomial foi proposto. Os resultados apontaram a solução próxima da ótima. No entanto, o trabalho não utiliza como métrica de comparação a

[1] Valor indicativo de quão perto a solução está do objetivo.

maximização de utilização de recursos disponibilizados e de usuários atendidos, diferentemente desta pesquisa.

No trabalho de [Midya et al. 2018], foi utilizado o algoritmo meta-heurístico de Otimização de Enxame de Partículas como mecanismo de decisão para uma arquitetura de redes em três camadas em um ITS. Os autores consideraram como camadas os veículos nas estradas, *cloudlets* instaladas em unidades de beira de estrada e uma nuvem centralizada. Para cada camada considera-se seus objetivos de qualidade de serviço específico e é aplicada uma função *fitness* específica. Para simulação, utilizou-se o Sumo, o Network Simulator 3 e o Matlab. Quando comparado com técnicas tradicionais de alocação, o algoritmo apresentou resultados satisfatórios considerando as métricas de tempo de resposta e convergência do algoritmo. O trabalho também utiliza um algoritmo meta-heurístico, mas não utiliza VCs para compartilhamento de recursos ociosos, sendo explorado nesta pesquisa.

O algoritmo meta-heurístico do morcego foi utilizado para análise de desempenho em alocação de recursos nos trabalhos de [Feng et al. 2018] e [Jayswal 2020]. Em [Feng et al. 2018] foi proposto um algoritmo *bat* baseado na diminuição do peso da inércia, com o objetivo de melhorar as redes *femtocell* de acesso múltiplo por divisão ortogonal de frequência. O algoritmo foi comparado com um algoritmo genético e apresentou-se mais eficiente para alocar subcanais para melhorar a qualidade do sistema em uma rede heterogênea. No trabalho de [Jayswal 2020] os autores analisaram o tempo de execução, o tempo médio de início e o tempo médio de término para o agendamento de tarefas em uma nuvem. Para simulação, utilizaram o Cloudsim 3. A comparação foi feita com um algoritmo genético utilizado para agendamento. O resultado com o algoritmo bat provou ser mais eficiente nas três métricas consideradas. Ambos os trabalhos utilizaram como base o mesmo algoritmo que o ARCANE, mas não são direcionados para maximização da utilização de recursos, e não utilizam VCs para aproveitar os recursos ociosos da rede.

Em [Lieira et al. 2021] também utilizou-se um algoritmo meta-heurístico para alocação de recursos desenvolvendo o GROMEC. Os autores adaptaram o algoritmo de otimização do lobo cinzento para maximizar a alocação de recursos de dispositivos de usuários, considerando a limitação dos equipamentos de borda. Considerou-se um cenário urbano com quatro estruturas de borda, que através de uma simulação, utiliza a distribuição de Poisson e simula requisições de entradas e saídas de dispositivos da rede. O GROMEC foi comparado com técnicas tradicionais da literatura e apresentou os melhores resultados em relação a dispositivos atendidos, negados e bloqueados. O trabalho busca maximizar o atendimento dos usuários, porém não utiliza um simulador de ambiente real e também não faz o aproveitamento de recursos ociosos.

Na Tabela 1, são apresentadas as características dos trabalhos citados, além de destacar a principal contribuição desta pesquisa. A maioria dos trabalhos aplica algoritmos meta-heurísticos [Midya et al. 2018, Feng et al. 2018, Jayswal 2020, Lieira et al. 2021] e outros algoritmos tradicionais [Pereira et al. 2019, Nabi et al. 2017]. Há também trabalhos que utilizam como base o algoritmo meta-heurístico do morcego [Feng et al. 2018, Jayswal 2020], outros que utilizam cenários de VCs [Nabi et al. 2017, Midya et al. 2018], no entanto, salienta-se que somente o ARCANE utiliza um ambiente de tráfego real e busca maximizar a utilização dos recursos disponibilizados e a quantidade de tarefas aten-

didas. Sendo assim, é possível observar que os trabalhos supracitados não abrangem essa perspectiva, por isso a seguir é apresentado o ARCANE.

Tabela 1. Relação entre os trabalhos relacionados

Trabalho	Algoritmo Base	Aplicação (Cenário)	Alocação
[Pereira et al. 2019]	Processamento de Hierarquia Analítica (AHP)	FOG	Recursos
[Nabi et al. 2017]	Guloso (Greedy-N)	Nuvens Veiculares (VCs)	Recursos
[Midya et al. 2018]	Enxame de Partículas (PSO)	Nuvens Veiculares (VCs)	Recursos
[Feng et al. 2018]	Morcego (BAT)	Redes Femtocell	Recursos
[Jayswal 2020]	Morcego (BAT)	Computação em Nuvem	Tarefas
[Lieira et al. 2021]	Lobo Cinzento (GWO)	Computação de Borda	Recursos
ARCANE	Morcego (BAT)	Nuvens Veiculares (VCs)	Tarefas

3. Algoritmo Meta-heurístico para Alocação de Tarefas em Nuvens Veiculares

Esta seção apresenta o ARCANE, um algoritmo meta-heurístico para alocação de tarefas em uma VCs. Para facilitar o entendimento do ARCANE, será descrita a seguir definição do problema e sua modelagem e por fim o algoritmo proposto.

3.1. Definição do Problema e Modelo

Considerando um conjunto de tarefas T com $\{t = 1, ..., n\}$, em que cada tarefa possui as características (id_t, p_t, vt_t), onde id_t corresponde ao identificador de cada tarefa, p_t peso da tarefa (quantidade de processamento necessária para a tarefa) e vt_t a recompensa da alocação dessa tarefa. Essas tarefas podem ser classificadas como troca de mensagens, sugestão de rota, algum tipo de *streaming*, etc.

Dessa forma, o objetivo é de maximizar a recompensa das tarefas alocadas nas VCs, alocando assim o conjunto de tarefas T nessas VCs. O processamento de cada VC W corresponde a soma dos recursos compartilhados w_i para cada veículo v_i, representado por v_i $(i = 1, ..., z)$, que faz parte da VC. Uma tarefa com determinado id_t tem seu vt_t atribuído de acordo com o dobro do valor de p_t, sendo assim quanto maior a necessidade de processamento da tarefa, maior o valor dos recursos armazenados. Caso a VC selecionada não possua os recursos necessários para atender determinada tarefa, a mesma será recusada.

O modelo do sistema em que o ARCANE foi considerado é apresentada na Figura 1. Os veículos que estão perto uns dos outros e em movimento, formam as VCs por meio de uma clusterização utilizando o algoritmo DBSCAN (*Density Based Spatial Clustering of Application with Noise*) [da Costa et al. 2020]. Os veículos que não estão em uma VC e não tem capacidade de processar determinada tarefa, precisam pedir ao controlador da nuvem veicular para que aloque os recursos necessários e execute sua tarefa [da Costa et al. 2020].

Os *Roadside Units* (RSUs) são espalhados pela cidade em pontos estratégicos. Cada veículo possuí uma unidade de bordo (OBU, do inglês *On Board Unit*) para

realização da comunicação. Nesse cenário as comunicações da rede veicular são feitas de duas formas, *vehicle-to-vehicle* (V2V) e *vehicle-to-infrastructure* (V2I). Para realizar a alocação de recursos para as tarefas, o controlador da nuvem veicular utiliza o ARCANE, definindo qual VC está disponível para atender essa requisição, de forma rápida e utilizando o máximo de recursos computacionais disponíveis na VC.

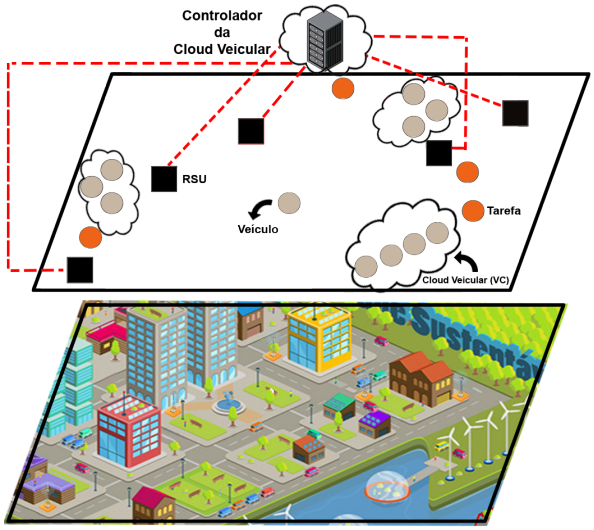

Figura 1. Representação do modelo e arquitetura do sistema.

3.2. ARCANE

O ARCANE tem como objetivo realizar o processo de tomada de decisão de alocação de tarefas nas VCs. Para tanto, o ARCANE foi modelado com base no algoritmo meta-heurístico do BAT e o *framework* conhecido como Evolopy [Yang 2010, Faris et al. 2016]. Os algoritmos meta-heurísticos surgem como uma alternativa aos algoritmos tradicionais baseados em modelos matemáticos. Os métodos tradicionais enfrentam desafios na modelagem e tratamento adequados de grandes quantidades de dados, ao mesmo tempo que fornecem soluções com desempenho significativo. Portanto, trabalhos exploram cenários de computação urbana em que abordagens meta-heurísticas passam a atuar na complexidade e lacuna de desempenho. O BAT é um dos algoritmos meta-heurísticos mais eficientes e pode ser facilmente adaptado a diferentes cenário [Katoch et al. 2020, Yang 2010, Okwu and Tartibu 2021].

O algoritmo BAT tem como base o comportamento de ecolocalização dos micro-morcegos utilizado na caça de suas presas. A maioria dos micro-morcegos são do tipo insetívoro (se alimentam de insetos). Essa espécie de morcego utiliza um tipo de sonar (ecolocalização), para detectar presas, localizar fendas e evitar obstáculos no escuro. O pulso sonoro emitido por eles é muito alto e quando esse pulso bate nas superfícies e o eco retorna, eles conseguem se localizar no ambiente [Yang 2010]. Nesse cenário, o ARCANE considera que os morcegos são as tarefas e sua presa são as VCs. Dessa forma o morcego (i.e., tarefa) "caça" utilizando a ecolocalização para encontrar sua presa (i.e., VC) e a tarefa seja alocada. Na Figura 2 é possível ver a aplicação desse efeito.

Para modelar o ARCANE as seguintes premissas foram estabelecidas:

P1 - Todas as tarefas utilizam o recurso de ecolocalização para encontrar suas VCs;

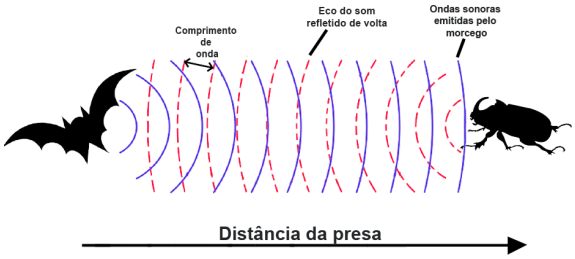

Figura 2. Representação do efeito de ecolocalização dos morcegos, adaptado de [Srivastava and Sahana 2019].

P2 - As tarefas voam aleatoriamente com velocidade v_i e posição x_i com uma frequência f_{min} fixa variando o comprimento de onda λ e volume do pulso A_0 para procurar a VC. É possível ajustar automaticamente o comprimento de onda ou a frequência dos pulsos emitidos e ajustar também a taxa de emissão desses pulsos $r \in [0, 1]$ de acordo com a proximidade do alvo;

Usando a abstração de que as tarefas são os morcegos, no ARCANE as tarefas têm uma frequência de ecolocalização f, velocidade v e sua posição no espaço x. A frequência f é determinada em um intervalo de $f \in [0, 2]$, considerando 0 a f_{min} onde os morcegos estão apenas voando (procurando) e 2 a f_{max} onde os morcegos estão atacando a presa. Dado que, onde se inicia o processo de atualização das melhores soluções, que podemos chamar de movimento de ataque, são atualizados os valores da frequência f. A velocidade v também é aumentada durante este processo pela junção dos valores da melhor solução atual e a frequência atualizada. Finalmente, a posição no espaço x são as soluções geradas. Esses valores são atualizados no decorrer do código, de acordo com o valor do *fitness* e as melhores soluções encontradas.

Seguindo o comportamento dos morcegos, quanto mais próximo a tarefa se aproxima da VC, o que significa encontrar a melhor solução, os valores de frequência e velocidade são alterados, atualizando e aumentando os valores até que a VC seja encontrada. Os morcegos não possuem um comportamento específico quanto ao seu movimento para caça, por isso o movimento foi considerado de forma aleatória. Para atualizar a frequência (f), velocidade (v) e as novas soluções (x), foram utilizadas as seguintes equações [Yang 2010]:

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{3}$$

onde $\beta \in [0, 1]$ é um vetor aleatório formado por distribuição uniforme. E x_* é a atual melhor posição (solução) da tarefa, que será selecionada como definitiva dentre as outras após a comparação com todas as VCs.

$$euclidiana = \sqrt{\sum_{j=1}^{n} (s_{ij} - f_{ij})^2} \tag{4}$$

Para gerar a população de tarefas, esta é gerada usando o cálculo da distância euclidiana (Equação 4) entre os parâmetros de recurso solicitados das tarefas e a capacidade disponível das VCs. Com os valores recebidos desses parâmetros, eles são usados como *input* na Equação 4. O uso da distância euclidiana vem normalizar os valores das soluções geradas. Além disso, os algoritmos meta-heurísticos geralmente usam uma técnica baseada no cálculo de *fitness*. Para realizar o cálculo do *fitness*, o ARCANE utiliza uma função de *benchmark*. Serão abordadas duas funções diferentes como forma de avaliar o impacto que a função de cálculo de *fitness* possui no processo de alocação de tarefas e no desempenho do algoritmo. Foram escolhidas a Função de Rosenbrock - Equação 5 e a Função de Ackley - Equação 6, sendo conhecidas na literatura como umas das principais utilizadas em testes de desempenho e benchmark em algoritmos de otimização [Liang et al. 2005]. O processo de cálculo do valor de *fitness* consiste em usar os valores gerados como soluções pela Equação 4 (Distância Euclidiana) como parâmetro passado para as funções de *benchmark* escolhidas. As funções de *benchmark* usam os valores das soluções passadas como parâmetro para calcular o *fitness*. Assim, um valor de *fitness* é gerado para cada solução da população de morcegos (população de tarefas). Este *fitness* é utilizado no algoritmo como forma de verificar qual a melhor solução e, para isso, busca-se o menor valor.

$$Rosenbrock(x) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right] \tag{5}$$

$$\begin{aligned} Ackley = &- 20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) \\ &- \exp(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + e \end{aligned} \tag{6}$$

No Algoritmo 1, é apresentado o algoritmo meta-heurístico para alocar tarefas nas VCs. Como parâmetro o ARCANE recebe os recursos das VCs e as tarefas. Nas primeiras linhas (1 a 2), os valores de frequência f, taxa de pulso r e volume A são iniciados. Na linha 3, a variável dim recebe o valor de 2. A variável dim é referente a dimensão das matrizes de populações, logo recebe o valor de 2 que corresponde ao peso e valor das tarefas. Na linha 4 $numTarefas$ recebe o $len(tarefas)$ em uma variável local. Para gerar as primeiras soluções/populações, é realizado o cálculo da Distância Euclidiana (Eq. 4) utilizando como parâmetros as tarefas e recursos disponíveis para obter uma população normalizada. Esse processo é executado em dois laços de repetição que usam como base o valor de dim e do $numTarefas$ (linhas 5 a 9). Após isso é calculado o valor do *fitness* através de uma das funções de *benchmark* escolhidas (Ackley ou Rosembrock) (linhas 10 a 12). Na linha 13 a variável $menorFitness$ recebe o menor valor de *fitness* gerado anteriormente. Nas linhas 14 a 31 é realizado o laço principal do algoritmo em que a lógica baseada no comportamento de ecolocalização dos morcegos é abstraída e utilizada. Dois laços são utilizados levando como base o $numTarefas$. Na linha 16, são atualizados os valores de f_i, v_i e x_i através das Equações (1, 2 e 3). Após isso, uma verificação de se

um número aleatório for maior que a taxa de pulso r_i então é selecionado uma solução aleatória (linhas 17 a 19). Na linha 20 é calculado um novo *fitness* para que nas linhas 21 a 24 outra comparação seja feita com o $novoFitness$ e o *fitness* atual daquela iteração e também é gerado um valor aleatório para verificar se o valor é menor que o volume A_i. Caso seja verdadeiro, a melhor solução é atualizada e o melhor *fitness* é atualizado. Nas linhas 25 a 27 é verificado se o $novoFitness$ é menor ou igual ao $menorFitness$ conhecido e caso verdade, a melhor solução é atualizada e o menor *fitness* é atualizado. Por último a melhor tarefa para ser alocada é retornada (linha 31).

Algoritmo 1 ARCANE

Entrada: $recursos, tarefas$
Saída: $melhorTarefa$
1: *Define a frequência f_i em x_i*
2: *Inicia a taxa de pulso r_i e o volume A_i*
3: $dim \leftarrow 2$
4: $numTarefas \leftarrow len(tarefas)$
5: **para** *i in range(0, dim)* **faça**
6: **para** *j in range(0, numTarefas)* **faça**
7: *Solução[i,j] ← Euclidiana (tarefas, recursos) (Equação 4)*
8: **fim para**
9: **fim para**
10: **para** *j in range (0, numTarefas)* **faça**
11: $fitness[j] \leftarrow Calcular\_Fitness(Solução[j, :])$
12: **fim para**
13: $menorFitness \leftarrow min(fitness)$
14: **para** *t in range (0, numTarefas)* **faça**
15: **para** *i in range (0, numTarefas)* **faça**
16: *Gera uma nova solução atualizando f_i, v_i, x_i com as equações 1, 2, 3*
17: **se** $(random() > r_i)$ **então**
18: *Seleciona uma nova solução aleatória entre as melhores soluções*
19: **fim se**
20: $novoFitness \leftarrow Calcular\_Fitness(novaSolucao)$
21: **se** $(novoFitness \leq fitness[i])$ **&** $(random() < A_i)$ **então**
22: *Atualiza a melhor solução*
23: *Atualiza o melhor Fitness*
24: **fim se**
25: **se** $(novoFitness \leq menorFitness)$ **então**
26: *Atualiza a melhor solução*
27: *Atualiza o menor Fitness*
28: **fim se**
29: **fim para**
30: **fim para**
31: **retorna** $melhorTarefa$

4. Avaliação e resultados

Esta seção avalia o desempenho do ARCANE para lidar com o problema de alocação de recursos e tarefas em uma VC. Para tanto, foi utilizo um *trace* de mobilidade real, e

o ARCANE foi comparado com outras soluções da literatura. A seguir, é apresentado o cenário utilizado, as métricas selecionadas e os parâmetros selecionados para gerar os resultados.

4.1. Configurações dos experimentos

Para realizar a simulação foi utilizado o Simulador de Mobilidade Urbana (SUMO, do inglês *Simulation of Urban Mobility*)[2] versão 0.32.0. Foi considerado a Interface de Controle de Tráfego[3] (TraCI, do inglês Traffic Control Interface) que utiliza a linguagem de programação Python[4]. Como forma de popular a simulação, foi utilizado o *trace* de mobilidade da cidade de Luxemburgo na Europa, o *Luxembourg SUMO Traffic* (LuST) [Codeca et al. 2015] que contém os registros de um período de 24 horas e abrange uma área de 155.95 km^2. É possível observar na Figura 3 a topologia complexa da cidade de Luxemburgo que ajuda a trazer uma maior fidelidade a aplicação da simulação referida, sendo a mesma aplicada em um cenário real.

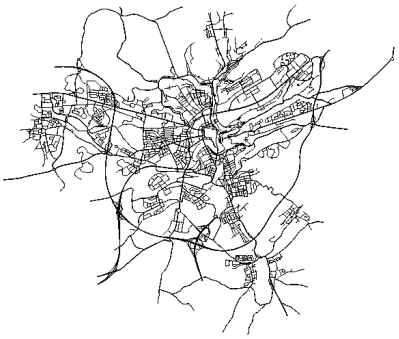

Figura 3. Cenário LuST - Imagem retirada do software SUMO.

O intervalo de clusterização definido foi de 60 segundos devido a dinamicidade da rede e mudança recorrente das VCs, provando-se mais eficiente devido a testes prévios realizados. A ocorrência das tarefas foi definida seguindo a Distribuição de Poisson com uma média de tarefas λ = 25. Como forma de definir um cenário mais desafiador para o algoritmo e seus concorrentes, foi escolhido o período de horário das 11:00 às 12:00 da manhã, sendo considerado o período com menor quantidade de carros no LuST. Foi atribuído um peso específico ao conjunto de tarefas T, sendo $peso = [1, 25, 50, 100, 200, 300]$ com um ganho de alocação sendo $2 \times peso$. O número de recursos atribuídos a cada veículo foi aumentado gradativamente com as configurações 1, 2 e 3 recursos de forma separada para melhor avaliação na questão do compartilhamento com diferentes unidades de recursos [da Costa et al. 2020].

Como forma de comparar o ARCANE , foram utilizados outros algoritmos para solução de problemas de otimização, sendo eles: programação dinâmica (DP, do inglês Dynamic Programming) que escolhe soluções ótimas locais, duas versões do algoritmo Greedy, sendo elas uma versão que considera apenas uma VC e outra que considera todas as VCs disponíveis (Greedy-N). Também para comparação, foi utilizado o ARCANE com

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

duas diferentes funções de *benchmark*, a Função de Ackley (ARCANE-A) e a Função de Rosenbrock (ARCANE-R).

Todos os resultados são médias geradas de uma amostragem oriunda de 33 execuções de simulação para cada combinação de parâmetros, gerando intervalos de confiança de 95%. Para calcular os valores obtidos, foi realizado a média aritmética dos valores gerados através dos seis pesos das tarefas (1, 25, 50, 100, 200 e 300) em cada configuração de recursos (1, 2, 3). Como forma de avaliar os algoritmos e o ARCANE , foram utilizadas as métricas de atendimento de tarefas, recompensa (ganho) por alocação e utilização dos recursos. A seguir são apresentados os resultados obtidos.

4.2. Impacto dos resultados obtidos

A Figura 4 apresenta o número de tarefas completas que foram atendidas. De forma geral o ARCANE superou os outros algoritmos no quesito tarefas atendidas nas três configurações. Na configuração 1, o ARCANE superou os outros algoritmos comparados atingindo 20,57% e 17.49% de tarefas atendidas nas versões com a Função de Ackley (ARCANE-A) e de Rosenbrock (ARCANE-R) respectivamente. Na configuração 2, com dois recursos para cada veículo, os desempenhos seguiram a mesma ordem com o ARCANE-A conseguindo atingir 46,71% de recursos atendidos, o ARCANE-R 40,47% o Greedy-N com 34,23%, seguido pelo Greedy (2,70%) e DP (2,20%). Na configuração 3, a ordem de desempenho se manteve seguindo os seguintes valores (79,99%, 70,06%, 60,13%, 5,42% e 5,09%). O ARCANE atendeu mais tarefas devido ao fato da seleção de tarefas ser realizada de forma mais eficiente entre os recursos e utilizar uma rotina mais otimizada em relação aos outros algoritmos.

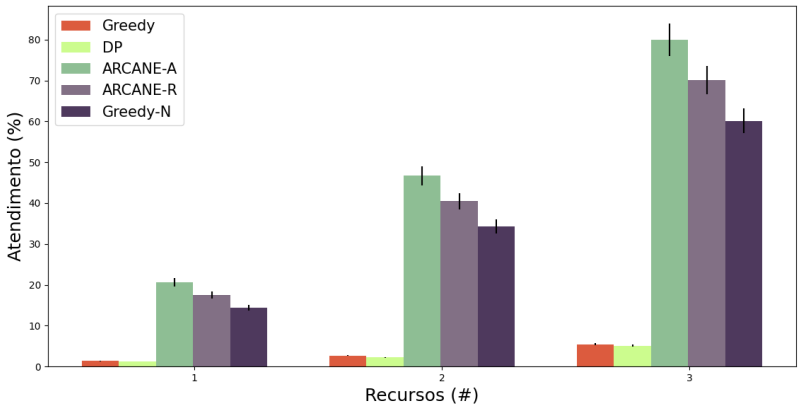

Figura 4. Desempenho do atendimento de tarefas.

Na Figura 5 é apresentado o gráfico com os valores de recompensa que os algoritmos obtiveram de acordo com as tarefas alocadas. O ARCANE apresentou um desempenho superior em relação aos outros algoritmos comparados atingindo 1,2, 2,96 e 5,11 (ARCANE-A) e 0,96, 2,53 e 4,38 (ARCANE-R) nas configurações 1, 2 e 3 respectivamente. Seguido pelos algoritmos Greedy-N (0,73, 2,10, 3,65), Greedy (0,5, 1, 2,33) e DP (0,43, 0,98, 2,10). Dessa forma, o ARCANE se sobressaiu aos outros algoritmos devido a utilizar tanto o cálculo da quantidade de processamento quanto o cálculo da recompensa para gerar o valor do *fitness* para alocar a melhor tarefa.

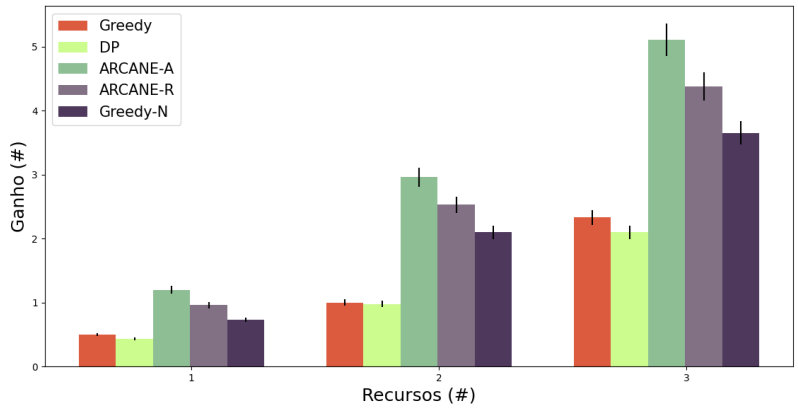

Figura 5. Desempenho da recompensa pela alocação de tarefas.

No quesito utilização dos recursos, os resultados podem ser vistos na Figura 6. De forma geral, os algoritmos ARCANE-A, ARCANE-R e Greedy-N apresentaram resultados bem acima dos outros dois comparados (Greedy e DP). Nas três configurações o ARCANE superou os outros algoritmos comparados na utilização de recursos devido ao fato de utilizar o cálculo do valor do *fitness* como forma de encontrar a melhor solução, melhorando sempre o valor do menor *fitness* dentro do laço principal do algoritmo.

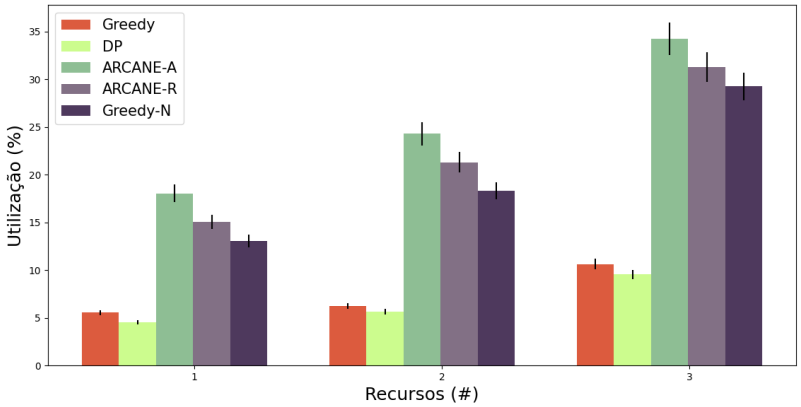

Figura 6. Desempenho da utilização de recursos.

5. Conclusão

O problema de alocação de tarefas em redes veiculares através das VCs estão se tornando cada vez mais exploradas. Diversos desafios são explorados devido ao fato da dinamicidade da rede. Além disso, a alta mobilidade dos veículos e a constante mudança de topologia faz com que o problema de alocação de recursos e tarefas se tornem ainda mais desafiador.

Diante desse desafio, este trabalho propôs o ARCANE para realizar a alocação de tarefas de forma eficiente nas VCs, aproveitando os recursos disponíveis da melhor forma. Para tanto, o ARCANE foi modelado com base em um algoritmo meta-heurístico

para alocar os recursos nas VCs. Quando comparado com outros algoritmos conhecidos na literatura tais como Greedy, Greedy-N e DP, o ARCANE mostrou mais eficiente nos quesitos atendimento de tarefas, recompensa pela alocação de tarefas e utilização dos recursos. Também é possível observar o impacto que a escolha da função de cálculo de *fitness* tem sobre os resultados de otimização no algoritmo meta-heurístico.

Como diretrizes para trabalhos futuros, espera-se modelar outros algoritmos meta-heurísticos para comparação no contexto das VCs, além de explorar o cenário de redes veiculares federada na alocação de tarefas. Para expandir a análise em um cenário genérico, outros *traces* de mobilidade serão avaliados de forma extensiva com diferentes volumes de tráfegos e padrões de movimento.

Referências

Assis, C. (2021). Billions poured into electric-vehicle companies, but much more will be needed before the auto industry changes. Acessado em: 16 de abril de 2021.

Codeca, L., Frank, R., and Engel, T. (2015). Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *VNC*, pages 1–8.

Correa, C., Ueyama, J., Meneguette, R. I., and Villas, L. A. (2014). Vanets: An exploratory evaluation in vehicular ad hoc network for urban environment. In *2014 IEEE 13th International Symposium on Network Computing and Applications*, pages 45–49.

da Costa, J. B. D., Meneguette, R. I., Rosário, D., and Villas, L. A. (2020). Combinatorial optimization-based task allocation mechanism for vehicular clouds. In *VTC2020-Spring*. IEEE.

Demir, M. S., Sait, S. M., and Uysal, M. (2018). Unified resource allocation and mobility management technique using particle swarm optimization for vlc networks. *IEEE Photonics Journal*, 10(6):1–9.

Faris, H., Aljarah, I., Mirjalili, S., Castillo, P. A., and Merelo, J. J. (2016). EvoloPy: An open-source nature-inspired optimization framework in python. In *Proceedings of the 8th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science and Technology Publications.

Feng, M., Guomin, L., and Wenrong, G. (2018). Heterogeneous network resource allocation optimization based on improved bat algorithm. In *2018 International Conference on Sensor Networks and Signal Processing (SNSP)*, pages 55–59.

Jayswal, A. K. (2020). Efficient task allocation for cloud using bat algorithm. In *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 186–190.

Katoch, S., Chauhan, S. S., and Kumar, V. (2020). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126.

Liang, J. J., Suganthan, P. N., and Deb, K. (2005). Novel composition test functions for numerical global optimization. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 68–75.

Liao, C., Wu, J., Du, J., and Zhao, L. (2017). Ant colony optimization inspired resource allocation for multiuser multicarrier systems. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6.

Lieira, D. D., Quessada, M. S., Cristiani, A. L., and Meneguette, R. I. (2021). Algorithm for 5g resource management optimization in edge computing. ieee latin america transactions. *IEEE Latin America Transactions*, 19(10).

Meneguette, R., De Grande, R., Ueyama, J., Filho, G. P. R., and Madeira, E. (2021). Vehicular edge computing: Architecture, resource management, security, and challenges. 55(1).

Meneguette, R. I. and Boukerche, A. (2017). A cooperative and adaptive resource scheduling for vehicular cloud. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 398–403.

Midya, S., Roy, A., Majumder, K., and Phadikar, S. (2018). Pso based optimized resource allocation in three tier cloud architecture for vanet. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 12–17.

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

Okwu, M. O. and Tartibu, L. K. (2021). *Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications*. Springer International Publishing.

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

Pereira, R. S., Lieira, D. D., Silva, M. A. C. d., Pimenta, A. H. M., da Costa, J. B. D., Rosario, D., Villas, L., and Meneguette, R. I. (2020). Reliable: Resource allocation mechanism for 5g network using mobile edge computing. *Sensors*, 20(19).

Rocha Filho, G. P., Meneguette, R. I., Torres Neto, J. R., Valejo, A., Weigang, L., Ueyama, J., Pessin, G., and Villas, L. A. (2020). Enhancing intelligence in traffic management systems to aid in vehicle traffic congestion problems in smart cities. *Ad Hoc Networks*, 107:102265.

Srivastava, S. and Sahana, S. K. (2019). Application of bat algorithm for transport network design problem. *Applied Computational Intelligence and Soft Computing*, 2019:9864090.

Vikhar, P. A. (2016). Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74. Springer Berlin Heidelberg.

Zhang, T., Grande, R. E. D., and Boukerche, A. (2015). Vehicular cloud. In *Proceedings of the 12th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks - PE-WASUN '15*. ACM Press.