Optimization of Task Allocation in Edge Computing to Industrial Internet with Simulated Annealing

Vitor Lux Barboza[1], **Janine Kniess**[1], **Rafael Stubs Parpinelli** [1]

[1]Pós-Graduação em Computação Aplicada
Universidade do Estado de Santa Catarina - Joinville, Brasil

vitor.barboza@edu.udesc.br, janine.kniess@udesc.br, rafael.parpinelli@udesc.br

Abstract. *The growth of applications in the context of the Industrial Internet (IIoT) has resulted in new requirements to meet critical tasks in terms of response time. Edge Computing offers an alternative to the use of the cloud in relation to the processing of application data, as it offers resources at the edge of the network, enabling the processing of industrial device tasks with low latency. In this work, we propose an Approach for Task Allocation of industrial vehicles, using edge nodes. The edge node simultaneously receives multiple tasks from different vehicles to process. As a result, there was a demand to define the best task processing order that meets the deadline requirements imposed by the applications. The solution was modeled based on the Simulated Annealing meta-heuristic to find the service order within the time limit. The results obtained in the iFogSim simulator demonstrate that the Task Allocation Approach was able to select the best task processing combination that obeys the required service time.*

Resumo. *Com o crescimento do número de aplicações no contexto da Internet Industrial (IIoT), surgem novos requisitos para atender as tarefas críticas com relação ao tempo de resposta. A Computação na Borda oferece uma alternativa em relação ao processamento dos dados na nuvem computacional, pois emprega recursos na borda da rede local, possibilitando o processamento das tarefas dos dispositivos industriais com baixa latência. Neste trabalho, propõem-se uma Abordagem para a Alocação de Tarefas de veículos industriais, utilizando-se de nós de borda (edge). O edge recebe simultaneamente múltiplas tarefas de diferentes veículos. Como consequência, adveio a demanda por definir a melhor ordem de processamento das tarefas que atenda aos requisitos de prazo impostos pelas aplicações. A solução foi modelada com base na meta-heurística Simulated Annealing para encontrar a melhor ordem para o atendimento dentro do tempo limite. Os resultados obtidos no simulador iFogSim, demonstram que a Abordagem para a Alocação de Tarefas pôde selecionar a melhor combinação de processamento que obedeça ao tempo de atendimento requerido.*

1. Introdução

Internet das coisas (IoT) é um conceito que visa incluir conectividade sem fio em dispositivos do dia a dia, a fim de permitir muitas formas de aplicativos inteligentes. A motivação para tais aplicações é a análise de uma enorme quantidade de dados coletados por dispositivos com ligação à Internet [de Figueiredo Marques and Kniess 2019]. Com a evolução

das tecnologias de comunicação sem fio na indústria, um volume maior de dispositivos passou a estar conectado à Internet. De acordo com [Qiu et al. 2020] o movimento de Internet das Coisas na indústria originou um segmento de dispositivos chamados Internet das Coisas Industriais, do inglês, *Industrial Internet of Things (IIoT)*. Exemplos típicos de aplicações de IIoT incluem fábricas inteligentes, manutenções remotas, vigilância industrial, monitoramento e sensoriamento das linhas de produção [Sisinni et al. 2018]. A Computação na Nuvem é uma tecnologia que provê recursos significantes para as aplicações do setor, como por exemplo, armazenamento de longo prazo.

Em seu trabalho [Sisinni et al. 2018] explica que dispositivos IIoT fazem parte de um ecossistema industrial modelado para atender serviços de produção e tem por principal objetivo melhorar a eficiência e inteligência da produção industrial. Tais dispositivos são essencialmente heterogêneos e semelhantes a IoT tradicional, também utilizam microcontroladores. Geralmente são dispositivos com poucos recursos computacionais, possuem finalidades pré-definidas e funcionalidades restritas para o que foram desenvolvidos.

No contexto de IIoT os dispositivos por vezes adquirem alta criticidade no ambiente em que estão inseridos, pois efetuam controle de processos e de fluxos fabris, onde paradas podem acarretar em perdas financeiras, ou acidentes. O tempo de resposta dos IIoT têm grande relevância neste ambiente, onde atrasos ou falhas de comunicação ocasionam inviabilidade na execução das aplicações. O envio dos dados dos dispositivos industriais de uma planta fabril, por exemplo, para a Nuvem pode aumentar a latência e inviabilizar o atendimento dentro do tempo limite para a execução da tarefa. A Computação na Borda, surge como uma alternativa ao uso da Nuvem, pois disponibiliza recursos computacionais na borda da rede, próximo dos dispositivos IIoT. A mesma tem sido usada em IIoT para processar os dados na borda da rede local e reduzir a latência das aplicações. Por exemplo, considere o cenário de aplicação IIoT apresentado na Figura 1.

Neste cenário, tem-se o Nível 1 composto pelos dispositivos industriais IIoT, como veículos conectados em rede, que executam tarefas diversas. Por exemplo, um dos modelos de veículo, ao chegar em uma máquina de produção que necessita abastecimento, identifica qual insumo está faltando para abastecê-la corretamente. Esta identificação de insumo faltante é feita por meio de uma câmera que tira fotos do compartimento de abastecimento da máquina de produção. Então, o veículo envia as imagens obtidas para o nó *edge* na borda que deve processar a tarefa e fornecer ao veículo o resultado do processamento, para que o mesmo possa dar sequência a sua atividade. Cada dispositivo IIoT tem limitações quanto aos recursos computacionais e por esta razão necessitam direcionar o processamento de suas tarefas para o nível acima deles.

No Nível 2, está a Computação na Borda composta de nós *edge* e os meios de comunicação para alcançá-los, como comutadores de redes sem fio e antenas. Considera-se neste trabalho que cada veículo se comunica exclusivamente com um único *edge*. No Nível 3, encontra-se a nuvem computacional. Sua principal função é prover armazenagem de longo termo dos dados já processados na borda. Diferentes modelos de veículos podem enviar múltiplas requisições para o nó *edge* simultaneamente. O *edge* deve receber e escalonar as tarefas recebidas de modo a atender aos requisitos de tempo especificados pelas aplicações. A Alocação de Tarefas [Qiu et al. 2020] é o meio pelo qual é possível distribuir as tarefas dos dispositivos da melhor forma e requer o uso de técnicas apropriadas para conciliar os múltiplos requisitos das aplicações, como distinção para tarefas mais

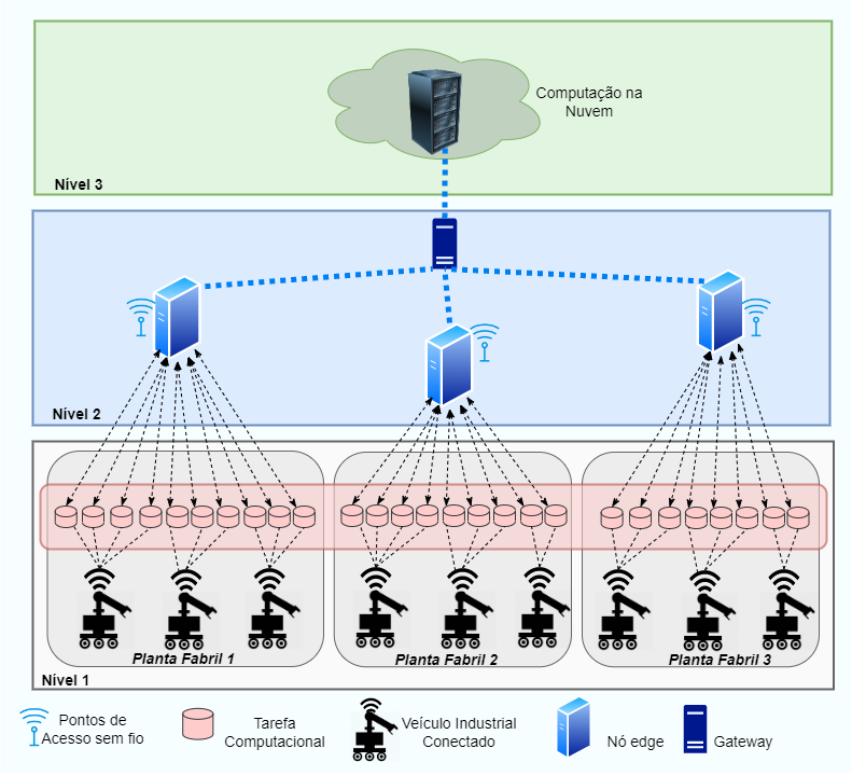

Figura 1. Cenário de Aplicação (Elaborado pelos autores).

ou menos urgentes, prazos de processamento e possíveis dependências entre as tarefas.

Neste artigo, apresenta-se uma Abordagem de Alocação de Tarefas de dispositivos IIoT móveis, utilizando-se da computação da borda (*edge*) para reduzir a latência da rede e atender ao requisito de Tempo de Resposta limite das tarefas. Na abordagem proposta, além do uso da Computação na Borda, foi desenvolvido um escalonador com base na meta-heurística de otimização *Simulated Annealing (SA)* [Kirkpatrick et al. 1983] para selecionar a melhor combinação otimizada que atenda ao requisito, limite de tempo das tarefas, no cenário com IIoT móveis.

O artigo está organizado da seguinte forma: na Seção 2 é apresentado o referencial teórico e os trabalhos relacionados. Na Seção 3 é descrita a Abordagem para a Alocação de Tarefas de IIoT móveis, detalhando-se a formulação do problema e a modelagem do algoritmo com o SA. Na Seção 4 são apresentados os resultados obtidos com ambiente de simulação iFogSim, [Gupta et al. 2017]. Por fim, na Seção 5 são apresentadas as conclusões do estudo e os trabalhos futuros.

2. Trabalhos Relacionados

Alguns estudos propõem soluções para o escalonamento de tarefas, apresentando diferentes estratégias e heurísticas na busca por soluções otimizadas ao problema. No trabalho de [He 2022] o autor propõe uma abordagem com uso de computação de borda, para prover a alocação de tarefas de dispositivos industriais. Em sua abordagem cada tarefa possui um prazo para completar o processamento. Tarefas sensíveis a atrasos são priorizadas pelo mecanismo para que recebam o processamento com antecedência. A meta-heurística selecionada pelo autor para definir as decisões do processo de alocação de tarefas é o *Genetic*

Algorithm (GA) [Davis 1991]. Utilizando o simulador CloudSim, os autores compararam a solução de escalonamento com dois algoritmos exatos, dentre eles o de busca gulosa. O estudo propõem cenários variando de 15 a 40 tarefas. Os resultados apresentam o GA atendendo satisfatoriamente a minimização do tempo de resposta e com resultados melhores que os outros dois algoritmos comparados.

No trabalho relacionado de [Xue et al. 2020] os autores discutem os desafios das aplicações industriais com alta demanda de recursos e dispositivos IIoT com baixa capacidade computacional e energética, os mesmos apresentam a computação na borda, como uma solução para aproximar os recursos necessários na execução de tais aplicações em detrimento da computação em nuvem. Os dispositivos IIoT são móveis e cada tarefa possui definido o seu tempo limite de conclusão. Os autores propõem um trabalho de priorização com base na urgência das tarefas para direcionar a ordem de processamento. Utilizou-se a programação dinâmica na estratégia de escalonamento, e para fins de comparação, implementaram outros algoritmos que atuam na alocação de tarefas, como *Prediction Based Sub-Task Offloading* [Wei et al. 2018] e *Offloading algorithm based on reinforcement learning* [Kiran et al. 2020]. Os autores simularam o algoritmo de alocação de tarefas no simulador EdgeCloudSim. As métricas avaliadas foram tempo de resposta, consumo de energia e custo de processamento. Os resultados mostraram superioridade quanto às métricas avaliadas em relação aos trabalhos comparados.

Em [You and Tang 2021], os autores propõem um cenário com dispositivos industriais e uso da computação de borda para processar as tarefas dos mesmos. Os autores propõem uma estratégia de otimização para solução do problema do escalonamento de alocação de tarefas, buscando minimizar o tempo de resposta do processamento das tarefas, seu consumo de energia e o custo total de processamento, porém as tarefas consideradas não possuem prazos de limite. A meta-heurística selecionada para resolver o problema de decisão é o *Particle Swarm Optimization (PSO)* [Kennedy and Eberhart 1995]. Neste trabalho foi proposta a utilização de mais de um *edge* para processamento das tarefas, e consideraram uma quantidade de 50 a 250 tarefas. Os resultados apresentam vantagem do algoritmo PSO frente aos demais usados como comparação e alcançando redução do tempo de resposta para as tarefas dos dispositivos.

Em [Matrouk 2023] o autor oferece um solução em computação de borda com capacidade de priorização na alocação de tarefas provenientes de dispositivos móveis. As tarefas são adicionadas a uma fila de processamento do servidor de borda, e aguardam até seguirem para o processamento de acordo com a otimização combinatória efetuada. No estudo, as tarefas possuem prazos definidos de conclusão, e registram o tempo que as mesmas permanecem em fila. O objetivo é reduzir a sobrecarga de processamento do nó de borda. O autor trabalha com o algoritmo bio-inspirado *First Fitness Animal Migration Optimization* para realizar as priorizações e as alocações de tarefas. Nas simulações, realizadas no simulador iFogSim, foram utilizadas de 50 a 300 tarefas e os resultados apresentados demonstram superioridade da solução proposta em comparação aos outros algoritmos avaliados pelos autores.

Em comparação aos trabalhos relacionados, a Abordagem Proposta para alocação de tarefas (descrita na Seção 3), compreende um conjunto de critérios que, (i) atende tarefas com um prazo definido de finalização; (ii) oferece um esquema de priorização de tarefas críticas em detrimento a outras não críticas; (iii) considera o tempo que a tarefa

estará aguardando atendimento em fila, quando há múltiplas tarefas na fila; (iv) atua sob dispositivos com mobilidade. Como solução otimizadora é utilizada a meta-heurística *Simulated Annealing (SA)* [Kirkpatrick et al. 1983]. Na estratégia de escalonamento deste presente trabalho, o SA desempenha o papel de apresentar para o *edge*, a decisão otimizada sobre a melhor ordem de execução das tarefas, de modo a atender aos pré-requisitos propostos pela aplicação na Função Objetivo, que é obtida a partir do cálculo do Tempo de Resposta de cada tarefa. Com esta abordagem busca-se encontrar o melhor tempo de resposta que atenda aos requisitos de tempo limite das tarefas das aplicações industriais. A Tabela 1 expõe as diferenças entre a Abordagem Proposta e os trabalhos relacionados.

Tabela 1. Comparação Trabalhos Relacionados

Autor/Ano	Prazos de Alocação	Esquema de Priorização	Tempo de Enfileiramento	Mobilidade	Heurística
(XUE; WU; YUE, 2020)	Sim	Sim	Sim	Sim	Algoritmo com Programação Dinâmica
(YOU; TANG, 2021)	Não	Não	Não	Sim	Algoritmo Particle Swarm Optimization
(HE, 2022)	Sim	Sim	Não	Não	Algoritmo Genético
(MATROUK, 2023)	Sim	Sim	Sim	Sim	Algoritmo First Fitness Animal Migration Optimization
(Abordagem Proposta, 2023)	Sim	Sim	Sim	Sim	Algoritmo Simulated Annealing

3. Abordagem para Alocação de Tarefas Industriais em *Edge*

Conforme representado na Figura 1, diferentes veículos podem enviar requisições ao nó *edge* situado na planta fabril. Ressalta-se, que na modelagem da alocação de tarefas apresentada, considera-se apenas um único *edge* na planta, e que o mesmo possui recursos computacionais suficientes para o funcionamento do escalonador.

3.1. Formulação do Problema da Alocação de Tarefas

A formulação do problema considera o cálculo do tempo de resposta de cada tarefa enviada pelos veículos, que deve ser minimizado e atender aos prazos propostos. Considera-se que existem *V* veículos industriais, $V = \{1, 2, \ldots, v\}$, o índice do veículo $v \in V$. Cada veículo envia múltiplas tarefas a serem executadas no respectivo *edge*. As tarefas enviadas por um veículo específico são $M = \{1, 2, \ldots, i\}$, o índice da tarefa $i \in M$.

Cada tarefa corresponde a uma tupla, que contém quatro atributos, representados por $T_{v,i} = (D_i, C_i, T_i^{limit}, v)$, o primeiro, D_i indica o volume de dados da tarefa em bits, este valor varia de tarefa para tarefa, o segundo, C_i é a quantidade de ciclos de CPU para processar 1 bit de dados da tarefa, já T_i^{limit} denota o prazo limite para execução completa, em milissegundos, que a tarefa pode tolerar, e *v* o índice do veículo que originou a tarefa.

Cada *edge*, em que j é o índice do *edge*, possui uma fila de entrada de tarefas, a quantidade de possibilidades diferentes de alocação de tarefas em um *edge* é de $M\ fatorial\ (M!)$. A representação do problema é tratada como uma permutação de inteiros, na qual cada ordem combinatória de execução possui um custo associado, o tempo de resposta, e que será levado em consideração pelo otimizador no cálculo juntamente com o prazo limite de cada tarefa.

Na abordagem proposta, o processo de alocação inclui os seguintes estágios:

- Transmissão da mensagem: Para calcular o tempo de transmissão para o *edge*, deve obter-se a taxa de transmissão da rede $R_{v,j}$. Segundo [Yang et al. 2022] e [You and Tang 2021], divide-se o volume de dados da tarefa (tamanho do pacote), pela taxa de transmissão da rede:

$$RT^{j}_{trans,v,i} = \frac{D_i}{R_{v,j}} \tag{1}$$

- Tempo de fila: Quando os dados chegam no *edge* e o mesmo possui recursos computacionais livres, a tarefa será executada imediatamente, porém, considera-se neste estudo que pode haver um tempo de enfileiramento de tarefas no *edge*, denominado $RT^{j}_{fila,i}$, que será registrado pelo *edge* e considerado no cálculo Tempo de Resposta.
- Processamento: é o tempo que o *edge* leva para completar a tarefa $RT^{j}_{exec,i}$. Calcula-se quantos ciclos de CPU são necessários para processar toda a tarefa, $C_{i,j}$. Para tal, multiplica-se o tamanho total da tarefa, D_i, pelo número de ciclos de CPU para processar 1 (um) bit, C_i, este número pode variar de tarefa para tarefa. Portanto:

$$C_{i,j} = D_i \times C_i \tag{2}$$

  Assim, encontra-se o número total de ciclos de CPU necessários para processar todos os bits da tarefa. Em seguida, calcula-se o tempo necessário para o *edge* executar todos esses ciclos, dividindo $C_{i,j}$, pela frequência do processador do nó *edge*, f_j. Como resultado, tem-se o tempo necessário para processar toda a tarefa, $RT^{j}_{exec,i}$, em milissegundos. Este cálculo é expresso por:

$$RT^{j}_{exec,i} = \frac{C_{i,j}}{f_j} \tag{3}$$

O tempo de resposta total para completar a tarefa *i* do veículo *v* no *edge j*, inclui o tempo de transmissão da mensagem de requisição ao *edge j*, o tempo em fila do pacote, e o tempo de processamento no *edge*, sendo representado por:

$$RT^{j}_{v,i} = RT^{j}_{trans,v,i} + RT^{j}_{exec,i} + RT^{j}_{fila,i} \tag{4}$$

Uma vez que cada tarefa possui um prazo máximo de execução T^{limit}_i, o cálculo da otimização combinatória que busca encontrar a solução otimizada de determinado conjunto de tarefas, deve considerar o prazo estipulado de cada tarefa. Portanto, finalizar a tarefa dentro do tempo limite constitui uma restrição na modelagem do problema, que deve ser considerada na função objetivo. A restrição é aplicada ao verificar se o valor de $RT^{j}_{v,i}$, na Equação 4 é menor ou igual ao valor de T^{limit}_i:

$$RT^{j}_{v,i} \leq T^{limit}_i \tag{5}$$

Deste modo, para atender a esta restrição, uma penalidade é adicionada a função objetivo para penalizar a ocorrência da infração e conduzir o algoritmo de otimização à soluções

otimizadas que atendam à restrição. A Equação 6 representa a função penalidade, sendo g o coeficiente de penalidade. Quanto maior o atraso, maior será a penalidade aplicada.

$$penalti(x) = g \times \sum_{v=1}^{V} \sum_{i=1}^{M} (RT_{v,i}^{j} - T_i^{limit}) \tag{6}$$

O coeficiente de penalidade $g \in [0, 1]$, sendo que o valor 0 equivale a nenhuma penalidade e 1 equivale à penalidade máxima. Para normalizar a função penalidade perante a função objetivo, definiu-se que o pior caso corresponde a $RT_{v,i}^{j} \geq 2 \times T_i^{limit}$, ou seja, qualquer valor de tempo de resposta maior que o dobro do tempo limite deve receber a penalidade máxima, $g = 1$. Para qualquer valor entre, $T_i^{limit} < RT_{v,i}^{j} \leq 2 \times T_i^{limit}$, a penalidade é aplicada como, $0 < g < 1$, sendo g proporcional ao quanto este valor se aproxima da penalidade máxima. Quando $RT_{v,i}^{j} \leq T_i^{limit}$, nenhuma penalidade é aplicada.

Com base na formulação do problema apresentada, a função objetivo, que busca minimizar o tempo de resposta da alocação de tarefas é expressa por:

$$min \sum_{v=1}^{V} \sum_{i=1}^{M} RT_{v,i}^{j} \; sujeito\, a: \; RT_{v,i}^{j} \leq T_i^{limit} \tag{7}$$

A função $f(x)$ com adição da função penalidade é expressa na Equação 8:

$$f(x) = \sum_{v=1}^{V} \sum_{i=1}^{M} RT_{i,v}^{j} + g \times \sum_{v=1}^{V} \sum_{i=1}^{M} (RT_{v,i}^{j} - T_i^{limit}) \tag{8}$$

3.2. Algoritmo Escalonador de Tarefas IIoT com Simulated Annealing

O *Simulated Annealing* é um algoritmo inspirado em fenômenos físicos inerentes à têmpera de metais, no qual a temperatura aplicada sobre o metal é um fator determinante no processo, agitando as moléculas, quando o metal é aquecido, ou estabilizando as moléculas, quando este é resfriado. Seguindo esta inspiração o algoritmo SA inicia com a temperatura alta, oferecendo uma exploração do campo de busca, com buscas por ótimos globais, na sequência conforme a temperatura vai reduzindo, o algoritmo é direcionado a busca de refinamento local. O tempo de resfriamento do metal é percebido na forma de iterações do algoritmo e o resfriamento da temperatura ocorre com o passar das iterações. Portanto, como descreve [Yuan et al. 2022] a definição do parâmetro de resfriamento da temperatura deve ser observado atentamente, pois influencia sobre o SA se aproximar e alcançar o ótimo global, ou não.

O Algoritmo 1 representa o funcionamento do *Simulated Annealing (SA)* na alocação de tarefas em um nó *edge*. Os dados de entrada são dispostos na forma de uma matriz, denominada *M1*, que corresponde aos tempo de resposta de cada tarefa obtidos a partir da Equação 8. Também são inseridos dois parâmetros nativos ao SA: (i) o número de iterações N e; (ii) o coeficiente de resfriamento α, usado na equação de resfriamento.

No Algoritmo 1, na linha 8, chama-se a função de geração da solução inicial S. Na linha 9 é calculado o tempo de resposta total. As linhas, 10 e 11 atribuem a S^* a sequência inicial obtida e o melhor tempo de resposta. Na linha 12 é iniciada a execução das iterações até que o número total de iterações N seja alcançado. Na linha 13, a função

de geração de soluções vizinhas faz um *swap* entre as posições do vetor da solução inicial, salvando a nova sequência obtida em outro vetor. Quando uma nova solução vizinha é gerada, o tempo total de resposta desta sequência é calculado (linha 14).

Na linha 18, verifica-se se a solução vizinha é melhor que a solução atual, caso sim, a nova solução é armazenada. Em seguida, é avaliado se a solução atual é a melhor solução encontrada até o momento, caso positivo, o resultado é salvo. Se a solução vizinha não é melhor que a atual, o SA subtrai S de S' e gera o Δ dos tempos de resposta (linha 21). Por fim, gera um valor de r aleatoriamente entre 0 e 1, e na linha 23 faz uma conferência se r é menor que a constante matemática e elevada ao $\frac{-\Delta}{T}$. Para fins de exemplificação, considere que o valor de $T_i^{limit} = 100$ms, e o valor obtido para $RT_{v,i}$ foi 180ms, que corresponde a 80% a mais do limite, então, assume-se que $g = 0{,}8$.

A temperatura T influencia nesta decisão, e em certas iterações, levará o SA a assumir uma sequência com o pior resultado, a fim de ampliar a busca global. A dinâmica de funcionamento do SA, também determina que em algumas iterações a pior solução vizinha seja descartada. Por fim, calcula-se a equação de resfriamento que leva em consideração o número da iteração atual e o N para atribuir a temperatura. A saída de dados oferece o resultado da combinação otimizada com a ordem de processamento das tarefas, e o tempo total que será levado para o atendimento de todas as tarefas do conjunto. No Algoritmo 1, também são detalhados os procedimentos do SA para a geração da solução inicial, a geração de soluções vizinhas e cálculo do tempo de resposta de uma determinada sequência.

A solução inicial é gerada de forma aleatória (linhas 29 a 31), ao realizar a leitura da matriz *M1*, e de acordo com o tamanho da mesma determinar a quantidade de tarefas que uma sequência de decisão deve conter. Cada tarefa selecionada aleatoriamente é inserida em um vetor que compõe a solução inicial.

Para a geração das soluções vizinhas à solução S atual do SA, realiza-se um *swap* dos componentes do vetor da solução S de modo aleatório (linhas 40 e 41). Cada nova sequência gerada a partir do *swap* é gravada em um novo vetor S' que é retornado ao SA para continuidade da busca. Por fim, o procedimento *Calculate_RT*, encarrega-se de realizar o cálculo do tempo de resposta total de uma determinada solução S. Para cada componente do vetor S, há um valor de tempo de resposta, que é somado individualmente. O valor total obtido é retornado ao SA (linha 52).

4. Experimentos Computacionais

Experimentos foram conduzidos utilizando a plataforma de simulação de código aberto iFogSim, [Gupta et al. 2017], em sua versão 2. A implementação da Abordagem Proposta para a alocação de tarefas no iFogSim foi realizada na linguagem de programação Java.

O iFogSim proporciona que os dispositivos com mobilidade transitem por diversos pontos de acesso sem fio durante o trajeto de deslocamento. O simulador EdgeCloudSim [Sonmez et al. 2018], por exemplo, permite mobilidade, mas esta fica restrita a um único ponto de acesso sem fio por *edge*, o que inviabilizaria a dinâmica proposta no cenário de aplicação. O computador utilizado para as simulações possui um processador Intel Core i7 de 1.8GHz, com 16GB de RAM e sistema operacional Windows 10. A métrica avaliada é o tempo de resposta obtido do melhor escalonamento das tarefas no *edge*.

Algorithm 1 Task Allocation with SA

```
1:  procedure TASK_ALLOCATION
2:      Input: M1[ ], N, α
3:      Output: Best task allocation and Total response time.
4:      N ← N;                                              ▷ Numbers of iterations
5:      α ← α;
6:      T ← 1;
7:      Iter ← 1;
8:      S ← GENERATE_INITIAL_SOLUTION(M1[]);                ▷ Initial solution
9:      RT_total ← CALCULATE_RT(S′);                        ▷ Total Time Response
10:     S* ← S;
11:     RT_best ← RT_total;
12:     while (Iter < N) do
13:         S′ ← GENERATE_NEIGHBOR_SOLUTION(S);             ▷ Neighbor solution
14:         RT_neighbor ← Calculate_RT(S′);
15:         if (RT_neighbor < RT_total) then
16:             S ← S′;
17:             RT_total ← RT_neighbor;
18:             if (RT_total < RT_best) then
19:                 S* ← S;
20:                 RT_best ← RT_total;
21:                 Δ = RT_neighbor − RT_total;
22:                 r ← generate r ∈ [0, 1];
23:                 if (r < e^{−Δ/T}) then
24:                     S ← S′;
25:                     RT_total ← RT_neighbor;
26:                     T = (1 − (Iter/N))^α;
27:                     Iter + +;
28:                 end if
29:             end if
30:         end if
31:     end while
32:     return (S*, RT_best);
33: end procedure
34: procedure GENERATE_INITIAL_SOLUTION
35:     S ← new Array[ ];
36:     i = 1;
37:     while (le_size(S) < le_size(M1)) do
38:         task ← int_random_de(le_size(M1));
39:         if (S[i] ⊅ le_value(task, M1)) then
40:             S[i] ← le_value(task, M1);
41:             i++;
42:         end if
43:     end while
44:     return S;
45: end procedure
46: procedure GENERATE_NEIGHBOR_SOLUTION
47:     S′ ← new Array[ ];
48:     x1 ← int_random_de(le_size(S));
49:     x2 ← int_random_de(le_size(S));
50:     if (x1 == x2) then
51: GENERATE_NEIGHBOR_SOLUTION(S);
52:     else
53:         S′[x1] ← S[x2];
54:         S′[x2] ← S[x1];
55:     end if
56:     return S′;
57: end procedure
58: procedure CALCULATE_RT
59:     RT_total ← 0;
60:     for (i = 1; i ≤ le_tamanho(S); i + +) do
61:         RT_total = RT_total + le_valor(S[i]);
62:         return RT_total;
63:     end for
64: end procedure
```

4.1. Ambiente de Configuração

Nesta Abordagem Proposta considera-se o uso de apenas um nó *edge*. Dois cenários distintos foram avaliados com a Abordagem Proposta. Cada cenário é composto por

quatro veículos. A diferença entre os cenários, refere-se ao número de tarefas propostas ao escalonador pelos veículos. No primeiro cenário, quatro veículos, demandam a execução de quatro tarefas, totalizando dezesseis tarefas, $M = 16$. No segundo cenário, foram mantidos os quatro veículos, mas aumentou-se o número de tarefas para cinco por veículo, totalizando vinte tarefas $M = 20$.

O tamanho do pacote das tarefas D_i foi ajustado em 8Kbits (8.000 bits) e 16Mbits (16.000.000 bits). A quantidade de ciclos de CPU para processar 1 bit da tarefa C_i, foi definida como, 10 e 100. O primeiro cenário de testes possui 12 tarefas de 8Kbits e 4 de 16Mbits. As tarefas de 8Kbits utilizam 100 ciclos por bit, e as tarefas de 16Mbits utilizam 10 ciclos por bit. O segundo cenário possui 12 tarefas de 8Kbits e 8 de 16Mbits. Da mesma forma as tarefas de 8Kbits usam 100 ciclos por bit, e as tarefas de 16Mbits, 10 ciclos por bit. Os parâmetros utilizados nas simulações foram adaptados a partir dos parâmetros utilizados nos trabalhos relacionados de [Matrouk 2023] e [You and Tang 2021].

A taxa de transmissão da rede $R_{v,j}$ é de 100 Mbps e a tecnologia de Camada de Enlace é o IEEE 802.11. A frequência do processador do *edge*, f_j é de 8GHz. Foram executadas 30 repetições do Algoritmo 1 para cada cenário. Em ambos os cenários, utilizou-se os mesmos valores da variáveis: D_i, C_i, $R_{v,j}$ e f_j. Nos dois cenários de simulação foram avaliadas três diferentes equações de resfriamento do *Simulated Annealing*. Este teste tem o objetivo de identificar a curva de resfriamento mais aderente ao problema e conduzir o SA na busca de resultados mais otimizados. Para analisar a variação das equações de resfriamento, adicionou-se ao Algoritmo 1 as três equações, Equação 9, Equação 10 e Equação 11, detalhadas a seguir:

$$T = T_{inicial} \times \alpha \tag{9}$$

$$T = \frac{(T_{inicial} - T_{final})}{2} \times (1 + \cos(\frac{Iter \times \pi}{N}) + T_{final} \tag{10}$$

$$T = (1 - (Iter/N))^{\alpha} \tag{11}$$

Com relação aos parâmetros de funcionamento do SA, nos dois cenários, o valor de *N* foi definido em 5000 iterações. Na Equação 9 e Equação 10, utilizou-se a temperatura $T_{inicial} = 1000$ graus e $T_{final} = 0,01$ graus. A Equação 11 não requer a definição da temperatura inicial e final, sendo controlado pela própria equação de resfriamento, os valores permanecem entre 0 e 1. Quanto ao coeficiente de resfriamento, o mesmo atua de forma distinta nas equações que o utilizam. Na Equação 9 o coeficiente atua como uma taxa de redução direta da temperatura, o valor foi fixado em $\alpha = 0,99$. Na Equação 11 o coeficiente atua como um exponencial que afeta a curvatura de redução da temperatura, o valor definido foi $\alpha = 3$, a Equação 10 não requer o coeficiente de resfriamento.

4.2. Resultados e Analises

A matriz com a composição dos cálculos de tempos de resposta efetuados pela função $f(x)$, considera a ordem de chegada das tarefas, ou seja, antes da otimização acontecer. Para o Cenário 1 a mesma é apresentada na Figura 2. Na Figura 3, apresenta-se matriz

com a composição dos cálculos de tempos de resposta efetuados pela função $f(x)$ do Cenário 2. Durante a etapa de otimização, enquanto o SA está efetuando a busca por uma solução otimizada entre as soluções vizinhas, os tempos de fila são recalculados observando-se o tempo adicional que determinada tarefa levaria nas soluções vizinhas avaliadas, conforme as linhas 60 e 61 do Algoritmo 1.

Tarefa	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	29	82	46	68	52	72	42	51	55	29	74	23	72	46	16
2	29	0	55	46	42	43	43	23	23	31	41	51	11	52	21	24
3	82	55	0	68	46	55	23	43	41	29	79	21	64	31	51	48
4	46	46	68	0	82	15	72	31	62	42	21	51	51	43	64	32
5	68	42	46	82	0	74	23	52	21	46	82	58	46	65	23	64
6	52	43	55	15	74	0	61	23	55	31	33	37	51	29	59	56
7	72	43	23	72	23	61	0	42	23	31	77	37	51	46	33	20
8	42	23	43	31	52	23	42	0	33	15	37	33	33	31	37	28
9	51	23	41	62	21	55	23	33	0	29	62	46	29	51	11	36
10	55	31	29	42	46	31	31	15	29	0	51	21	41	23	37	40
11	29	41	79	21	82	33	77	37	62	51	0	65	42	59	61	44
12	74	51	21	51	58	37	37	33	46	21	65	0	61	11	55	52
13	23	11	64	51	46	51	51	33	29	41	42	61	0	62	23	60
14	72	52	31	43	65	29	46	31	51	23	59	11	62	0	59	12
15	46	21	51	64	23	59	33	37	11	37	61	55	23	59	0	16
16	16	24	48	32	64	56	20	28	36	40	44	52	60	12	16	0

Figura 2. Matriz 1 da função $f(x)$ - 1º Cenário, $M = 16$.

Tarefa	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	29	82	46	68	52	72	42	51	55	29	74	23	72	46	16	142	174	185	164
2	29	0	55	46	42	43	43	23	23	31	41	51	11	52	21	24	146	178	186	165
3	82	55	0	68	46	55	23	43	41	29	79	21	64	31	51	48	101	133	142	120
4	46	46	68	0	82	15	72	31	62	42	21	51	51	43	64	32	104	138	143	123
5	68	42	46	82	0	74	23	52	21	46	82	58	46	65	23	64	111	143	140	124
6	52	43	55	15	74	0	61	23	55	31	33	37	51	29	59	56	97	129	130	106
7	72	43	23	72	23	61	0	42	23	31	77	37	51	46	33	20	91	123	126	106
8	42	23	43	31	52	23	42	0	33	15	37	33	33	31	37	28	85	117	124	105
9	51	23	41	62	21	55	23	33	0	29	62	46	29	51	11	36	86	118	128	110
10	55	31	29	42	46	31	31	15	29	0	51	21	41	23	37	40	75	107	118	104
11	29	41	79	21	82	33	77	37	62	51	0	65	42	59	61	44	51	83	93	86
12	74	51	21	51	58	37	37	33	46	21	65	0	61	11	55	52	59	84	101	97
13	23	11	64	51	46	51	51	33	29	41	42	61	0	62	23	60	29	54	72	71
14	72	52	31	43	65	29	46	31	51	23	59	11	62	0	59	12	53	46	69	93
15	46	21	51	64	23	59	33	37	11	37	61	55	23	59	0	16	48	35	58	82
16	16	24	48	32	64	56	20	28	36	40	44	52	60	12	16	0	21	26	58	62
17	142	146	101	104	111	97	91	85	86	75	51	59	29	53	48	21	0	31	43	42
18	174	178	133	138	143	129	123	117	118	107	83	84	54	46	35	26	31	0	26	45
19	185	186	142	143	140	130	126	124	128	118	93	101	72	69	58	58	43	26	0	22
20	164	165	120	123	124	106	106	105	110	104	86	97	71	93	82	62	42	45	22	0

Figura 3. Matriz da função $f(x)$ - 2º Cenário, $M = 20$

O gráfico da Figura 4 apresenta a variação do tempo de resposta calculado pela *f(x)* em milissegundos para o primeiro cenário de simulação (Figura 2) e aplicado às três equações de resfriamento do SA. A Tabela 2 sumariza os valores obtidos na simulação.

O gráfico da Figura 5 apresenta a variação do tempo de resposta para o segundo cenário de simulação (Figura 3). Os valores encontrados nesta simulação foram sumarizados na Tabela 3. Visualiza-se nas Figuras 4 e 5 uma vantagem da Equação 11 perante as demais, obtendo os valores de tempo de resposta mais baixos entre as três equações em ambos os cenários. Esse comportamento, deve-se ao fato da equação de resfriamento conduzir a proporção de buscas globais ou locais no *Simulated Annealing*.

Os resultados das Figuras 4 e 5 foram confirmados através dos gráficos de convergência das Figuras 6 e 7, onde verifica-se que na Equação 11, a queda da temperatura não acontece de forma abrupta, como a Equação 9, e não muito lenta, como a Equação 10.

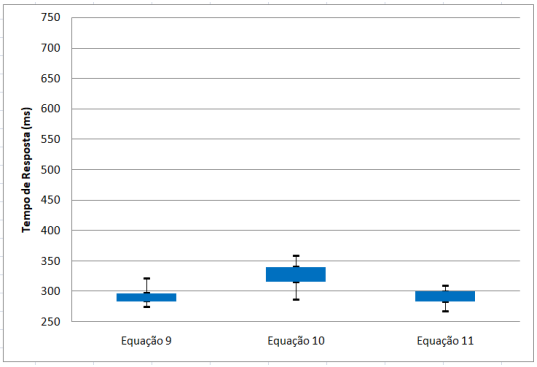

Figura 4. Tempo de Resposta: Equações de Resfriamento - 1º Cenário, $M = 16$.

Tabela 2. Resultados: Equações de Resfriamento - 1º Cenário, $M = 16$.

	Valor Mínimo	Primeiro Quartil	Terceiro Quartil	Valor Máximo	Mediana	Média	Desvio Padrão
Equação 9	275,0	284,0	297,0	321,0	292,5	292,9	11,9
Equação 10	287,0	315,8	340,3	358,0	329	327,0	18,8
Equação 11	267,0	283,0	299,8	309,0	293,5	291,4	11,7

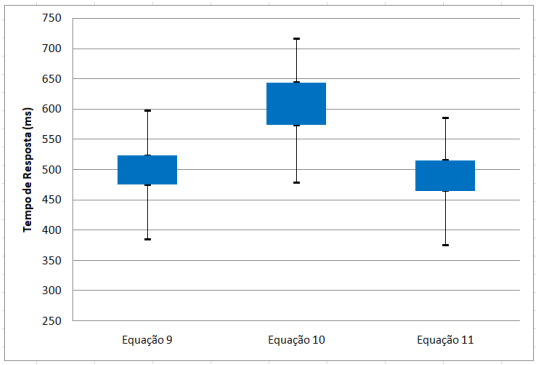

Figura 5. Tempo de Resposta: Equações de Resfriamento - 2º Cenário, $M = 20$.

Tabela 3. Resultados: Equações de Resfriamento - 2º Cenário, $M = 20$.

	Valor Mínimo	Primeiro Quartil	Terceiro Quartil	Valor Máximo	Mediana	Média	Desvio Padrão
Equação 9	403,0	449,3	476,0	499,0	462	463,2	20,5
Equação 10	447,0	485,8	572,0	628,0	542,5	535,9	51,7
Equação 11	400,0	437,3	463,8	485,0	452	449,6	20,6

Nos experimentos realizados, verificou-se que o tempo de resposta para o atendimento das tarefas (tempo de resposta máximo) no Cenário 1 ($M = 16$) para cada equação de resfriamento não deve ultrapassar a: Equação 9 = 321ms, Equação 10 = 358ms e Equação 11 = 309ms.

Dentre as 30 repetições do primeiro cenário com dezesseis tarefas, o menor valor de tempo de resposta obtido pela Equação 9 foi de 275 ms e o valor da média foi 292,9 ms. O menor valor obtido pela Equação 10 foi de 287 ms e a média foi 327 ms. O menor valor obtido pela Equação 11 foi de 267 ms e a média foi 291,4 ms.

No segundo cenário composto de vinte tarefas, os tempos máximos não devem ultrapassar a: Equação 9 = 499ms, Equação 10 = 628ms e Equação 11 = 485ms. O

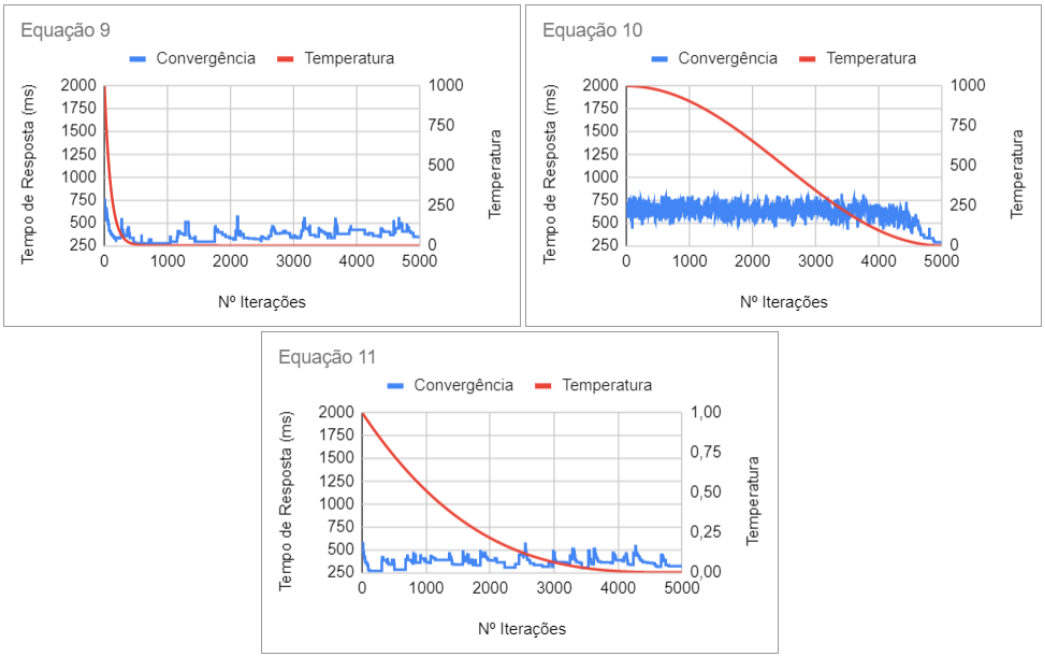

Figura 6. ***Gráficos de Convergência e Temperatura do SA - 1º Cenário***

menor valor de tempo de resposta obtido pela Equação 9 foi de 403 ms e a média foi 463 ms. O menor valor obtido pela Equação 10 foi de 447 ms e a média foi 535 ms. O menor valor obtido pela Equação 11 foi de 400 ms e a média foi 449 ms. A Figura 7 apresenta os gráficos de convergência e temperatura de cada equação no segundo cenário.

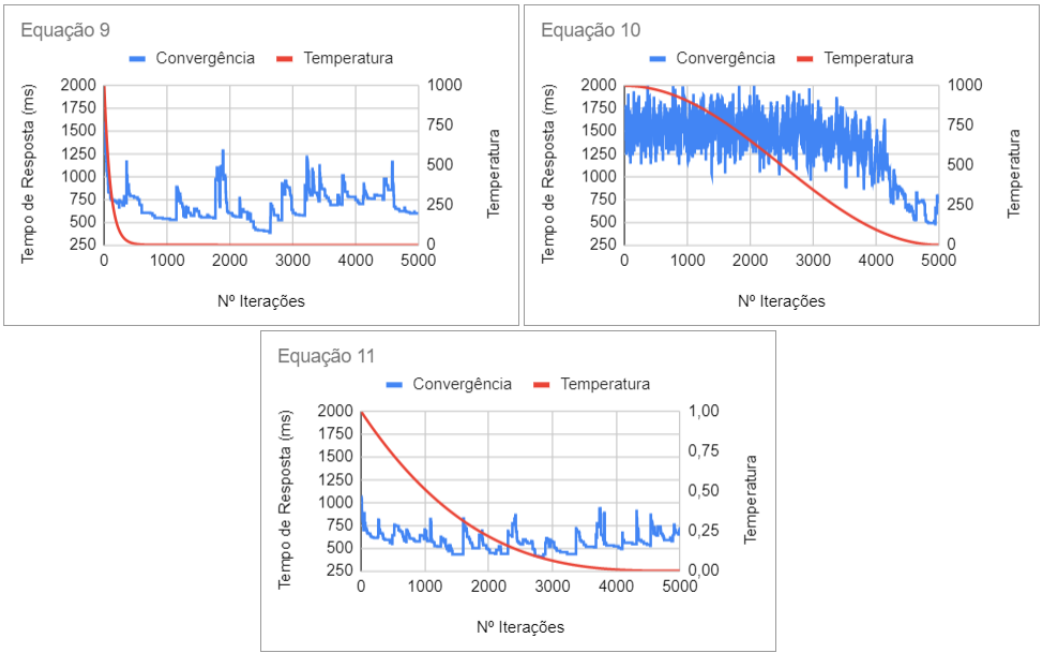

Figura 7. ***Gráficos de Convergência e Temperatura do SA - 2º Cenário***

Com base nos resultados, verificou-se que a abordagem desenvolvida para a Alocação de Tarefas com o SA, no cenário 1 ($M = 16$), encontrou a melhor sequência de

execução das tarefas: 3 -7 -5 -9 -15 -2 -13 -1 -16 -14 -12 -10 -8 -6 -4 -11. Esta sequência possibilitou a execução das tarefas com o menor tempo de resposta para o cenário 1, sendo 267 ms.

O melhor tempo de resposta encontrado no segundo cenário foi 400 ms e a melhor sequência de execução das vinte tarefas é: 9 -2 -13 -1 -11 -4 -6 -8 -10 -14 -12 -3 -7 -5 -15 -16 -17 -18 -19 -20. Em ambos os cenários, os resultados são considerados válidos perante a Função Objetivo. Outra métrica analisada foi o tempo de processamento da solução para a Alocação de Tarefas no *edge*. Neste experimento, utilizou-se a mesma máquina dos experimentos apresentados, um processador Intel Core i7 de 1.8GHz, com 16 GB de memória RAM, em sistema operacional Windows 10.

No primeiro cenário ($M = 16$), o tempo médio de execução do escalonador foi de 5 milissegundos. Como o menor tempo de resposta obtido foi de 267 ms com a Equação 11, somando-se esse tempo ao tempo de processamento, tem-se 273ms. Este tempo ficou dentro do tempo máximo que é de 309ms com a Equação 11.

No segundo cenário ($M = 20$), o tempo médio de processamento foi de 7 milissegundos. O menor valor de tempo de resposta encontrado pela Equação 11 foi de 400 ms. O tempo limite encontrado com a Equação 11 foi de 485ms. Somando-se o tempo de processamento e o tempo de resposta, tem-se 407ms, que atende ao máximo requerido.

5. Conclusões e Trabalhos Futuros

Neste artigo, apresentou-se a especificação, a implementação e a avaliação de desempenho de uma abordagem para escalonamento de tarefas de múltiplos dispositivos industriais utilizando um nó *edge*, que atende à uma planta fabril, sendo as tarefas com limite restritos para a sua finalização.

Alguns experimentos envolvendo a equação de resfriamento do SA foram efetuados, e observou-se que este parâmetro possui relevância significativa no comportamento do algoritmo, direcionando-o em tendências de mais iterações em buscas estocásticas ou mais iterações em buscas de refinamento, de forma que esta equação influencia nos resultados de otimização da Função Objetivo. Foram avaliadas três equações de resfriamento, selecionando-se a que obteve melhores resultados para compôr a solução proposta.

Os resultados obtidos demonstram que a meta-heurística de otimização *Simulated Annealing* proporciona a seleção de uma decisão otimizada para o escalonamento de tarefas dos dispositivos industriais para um nó *edge*. Conclui-se que a minimização do tempo de resposta na alocação e escalonamento de tarefas pôde ser atendida de forma adequada pela Abordagem Proposta utilizando o SA na composição da solução.

Como trabalhos futuros propõem-se: (i) utilizar diferentes instâncias que representem a diversidade do ambiente real da indústria e, (ii) implementar e realizar comparações com soluções da literatura que se assemelham a Abordagem Proposta.

Referências

Davis, L. (1991). Handbook of genetic algorithms.

de Figueiredo Marques, V. and Kniess, J. (2019). Mobility aware rpl (marpl): Mobility to rpl on neighbor variability. In Miani, R., Camargos, L., Zarpelão, B., Rosas, E., and

Pasquini, R., editors, *Green, Pervasive, and Cloud Computing*, pages 59–73, Cham. Springer International Publishing.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*.

He, J. (2022). Optimization of edge delay sensitive task scheduling based on genetic algorithm. In *2022 International Conference on Algorithms, Data Mining, and Information Technology (ADMIT)*, pages 155–159.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, pages 1942–1948 vol.4.

Kiran, N., Pan, C., Wang, S., and Yin, C. (2020). Joint resource allocation and computation offloading in mobile edge computing for sdn based wireless networks. *Journal of Communications and Networks*, 22(1):1–11.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Matrouk, K. e. (2023). Mobility aware-task scheduling and virtual fog for offloading in iot-fog-cloud environment. *Wireless Personal Communications*, 130:801–836.

Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M., and Wu, D. O. (2020). Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys and Tutorials*, 22(4):2462–2488.

Sisinni, E., Saifullah, A., Han, S., Jennehag, U., and Gidlund, M. (2018). Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734.

Sonmez, C., Ozgovde, A., and Ersoy, C. (2018). Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493. e3493 ett.3493.

Wei, F., Chen, S., and Zou, W. (2018). A greedy algorithm for task offloading in mobile edge computing system. *China Communications*, 15(11):149–157.

Xue, Y., Wu, X., and Yue, J. (2020). An offloading algorithm of dense-tasks for mobile edge computing. icWCSN 2020, page 35–40, New York, NY, USA. Association for Computing Machinery.

Yang, B., Pang, Z., Wang, S., Mo, F., and Gao, Y. (2022). A coupling optimization method of production scheduling and computation offloading for intelligent workshops with cloud-edge-terminal architecture. *Journal of Manufacturing Systems*, 65:421–438.

You, Q. and Tang, B. (2021). Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. *Journal of Cloud Computing*, 10:1–11.

Yuan, H., Hu, Q., Wang, M., Bi, J., and Zhou, M. (2022). Cost-minimized user association and partial offloading for dependent tasks in hybrid cloud–edge systems. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*.