

UMA ABORDAGEM GRASP PARA O PROJETO DE REDE MULTITAREFA DE SENSORES SEM FIO COM TOPOLOGIA DE FLORESTA DOMINANTE CLUSTERIZADA

Lucas Rian de Sousa Pereira & Rafael Castro de Andrade

Universidade Federal do Ceará

Departamento de Estatística e Matemática Aplicada

Campus do Pici S/N - bloco 910 - Fortaleza - Ceará, 60.440-900, Brasil

lucasrian@alu.ufc.br, rafael.andrade@ufc.br

Andréa Cynthia Santos Duhamel

Université Le Havre Normandie

Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes

ISEL, 25 Rue Philippe Lebon, 76600, Le Havre, France

andrea-cynthia.duhamel@univ-lehavre.fr

RESUMO

Redes Multitarefa de Sensores Sem Fio (MWSNs) vêm ganhando destaque por sua capacidade de monitorar simultaneamente diferentes aplicações, como em cidades inteligentes. Um dos principais desafios no planejamento dessas redes é a limitação energética dos sensores, geralmente alimentados por baterias. Este artigo propõe um novo procedimento GRASP voltado à otimização do consumo de energia em MWSNs. O desempenho do algoritmo é comparado ao do melhor modelo matemático da literatura, considerando instâncias de pequeno e médio porte. Nos testes com instâncias pequenas, o GRASP proposto encontrou a solução ótima em 26 das 30 instâncias avaliadas. Já em instâncias médias, obteve desempenho superior ao modelo matemático, encontrando soluções melhores em 10 das 14 instâncias analisadas, onde o modelo falha em obter soluções viáveis. Destaca-se ainda que a estrutura de vizinhança adotada na busca local contribuiu para uma melhoria média de 55% em relação às soluções iniciais.

PALAVRAS CHAVE. Meta-heurística GRASP, Otimização Combinatória, Rede Multitarefa de Sensores Sem Fio.

Meta-heurística, Otimização Combinatória

ABSTRACT

Multitask Wireless Sensor Networks (MWSNs) have gained attention for their ability to simultaneously monitor multiple applications, such as in smart cities. One of the main challenges in designing these networks is the efficient management of energy consumption, due to the limited battery life of the sensors. This paper proposes a new GRASP procedure aimed at optimizing energy usage in MWSNs. The algorithm's performance is compared with the best mathematical model from the literature on small- and medium-sized instances. For small instances, the proposed GRASP reached the optimal solution in 26 out of 30 cases. In medium-sized instances, it significantly outperformed the mathematical model, achieving better solutions in 10 out of 14 instances, whereas the model failed to find feasible solutions for all of them. Additionally, the neighborhood structure used in the local search led to an average improvement of 55% over the initial solutions.

KEYWORDS. GRASP Metaheuristic, Combinatorial Optimization, Multitask Wireless Sensor Network.

Metaheuristics, Combinatorial Optimization

1. Introdução

Redes de sensores sem fio (WSNs, do inglês *Wireless Sensor Networks*) consistem em um conjunto de dispositivos (sensores) de comunicação sem fio distribuídos em uma determinada área com o objetivo de coletar informações sobre o ambiente. Tradicionalmente, WSNs são projetadas para atender a aplicações específicas e independentes, como, por exemplo, a medição dos níveis de gases tóxicos no ar [Kadri et al., 2013] e a identificação de vagas livres em estacionamento [Tang et al., 2006]. No entanto, observa-se uma tendência de evolução das WSNs em direção a arquiteturas mais versáteis, capazes de suportar múltiplas aplicações com diferentes requisitos dentro de uma mesma infraestrutura física [Islam et al., 2012]. Essa disposição é denominada Rede Multitarefa de Sensores Sem Fio (MWSN, do inglês *Multitask Wireless Sensor Networks*). Recentemente, [Çavdar et al., 2022] destacaram exemplos do emprego de MWSNs em cidades inteligentes, onde uma única rede pode viabilizar simultaneamente o monitoramento do tráfego, a gestão de resíduos, a avaliação da qualidade do ar e a vigilância urbana.

Devido à capacidade limitada de bateria dos sensores, a eficiência energética é um aspecto importante no planejamento dessas redes. Arquiteturas baseadas em cluster são amplamente utilizadas na construção de WSNs devido ao seu baixo consumo energético [Vlajic e Xia, 2006]. Elas consistem na formação de uma estrutura de organização lógica denominada cluster, que é composta por um sensor mestre (cabeça do cluster) e demais sensores associados ao cluster, também chamados de sensores dominados, situados no raio de alcance desse sensor mestre.

Sensores dominados coletam e transmitem dados para sensores mestres que, por sua vez, coordenam seus clusters eliminando informações repetidas e repassando-as para a unidade central de processamento, responsável pela coleta e processamento de toda a informação sensoreada da rede. Dois clusters vizinhos se comunicam através de um sensor comum ao seu raio de ação, denominado sensor ponte. Por serem embarcados com atividades custosas de eliminação de redundância, sensores mestres devem formar um conjunto independente, ou seja, um sensor mestre não pertence ao raio de ação de outro mestre e, consequentemente, não há comunicação direta entre eles. No entanto, para garantir a comunicação entre todos os sensores da rede, o conjunto de sensores mestres deve formar um conjunto dominante, ou seja, todo sensor da rede ou faz parte desse conjunto ou, caso contrário, deve estar dentro do raio de ação de pelo menos um sensor mestre.

A organização de uma MWSN em k sub-redes, cada uma associada a uma aplicação específica a ser monitorada, estruturada com uma topologia clusterizada, é denominada na literatura ([Araujo et al., 2024; de Andrade et al., 2023]) de k -floresta clusterizada dominante (DCkF, do inglês *Dominating Clustered k -Forest*). Os nós dominados seguem um protocolo de comunicação que lhes permite permanecer em estado de baixo consumo energético (dormência) até que seja necessário realizar uma nova leitura de dados. Com isso, o consumo de energia das sub-redes é influenciado principalmente pelo peso das conexões entre sensores mestres e pontes, que é medido em função das distâncias entre eles. O objetivo do problema DCkF é minimizar o consumo energético da sub-rede com maior gasto energético.

Podemos encontrar algumas variantes do problema de projeto de redes de sensores sem fio na literatura relacionadas ao DCkF. Em [Santos et al., 2012], o projeto de WSN's em topologia clusterizada é formulado como o problema do Conjunto Independente Dominante com Restrições de Conectividade (IDSC). Os autores propõem um modelo de Programação Linear Inteira Mista (PLIM) e um conjunto de heurísticas para a resolução do problema, considerando como critérios de otimização minimizar o número médio de saltos dos sensores à unidade central de processamento da rede, ou a quantidade de clusters formados. Em Santos et al. [2016], os autores expandem o trabalho considerando o projeto de MWSN's, apresentando um conjunto de heurísticas para o problema com os mesmos critérios de otimização. O peso das conexões não é considerado nesses

trabalhos. Já em [Adasme e de Andrade, 2023], é tratado o caso particular do problema DCkF com $k = 1$, sendo propostos dois modelos de PLIM. A abordagem é generalizada em [de Andrade et al., 2023], onde os autores propõem um modelo de PLIM aplicável a valores arbitrários de k . Em [Araujo et al., 2024], é apresentado um modelo PLIM com uso de fluxo em redes que apresentou melhores resultados do que os reportados em [de Andrade et al., 2023] para instâncias de pequeno porte do problema com poucas dezenas de sensores. Sabe-se, para o caso particular $k = 1$ [Adasme e de Andrade, 2023], que o DCkF é NP-difícil, estando diretamente relacionado ao problema do Conjunto Dominante Conexo de Cardinalidade Mínima [Simonetti et al., 2011; Gendron et al., 2014]. Ambos os modelos em [Araujo et al., 2024; de Andrade et al., 2023] se mostraram pouco ou nada efetivos para instâncias com 40 ou mais sensores e poucas aplicações monitoradas. Isso evidencia a dificuldade intrínseca do problema, mesmo em sua forma mais simples.

Diante disso, é necessário utilizar procedimentos heurísticos capazes de gerar soluções de boa qualidade em tempo viável, especialmente para instâncias de maior porte. Ademais, destaca-se a elevada dificuldade de preservar a estrutura clusterizada durante os movimentos de busca local, de modo a manter a viabilidade da solução. Considerando essa lacuna, desenvolvemos um procedimento baseado na meta-heurística GRASP para o problema DCkF. Seus parâmetros foram calibrados e experimentos computacionais foram realizados em instâncias de pequeno e médio porte. Apresentamos resultados comparativos com o modelo de [Araujo et al., 2024], os quais indicam que o GRASP proposto é capaz de obter soluções de alta qualidade, como mostramos adiante.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2, apresentamos a caracterização matemática do problema DCkF. Em seguida, na Seção 3, descrevemos o procedimento GRASP proposto. Os experimentos computacionais são discutidos na Seção 4. Por fim, na Seção 5, apresentamos conclusões e direções para trabalhos futuros.

2. Definição Formal do Problema

Seja $G = (V, E)$ um grafo conexo representando uma WSN, onde V corresponde ao conjunto de sensores da rede e $uv \in E$ indica que os sensores u e v podem se comunicar. Seja $c_{uv} > 0$ a distância entre os sensores u e v . Dado um número de aplicações k , o problema DCkF consiste em encontrar um conjunto (floresta) $F = \{T_1, T_2, \dots, T_k\}$ de k árvores disjuntas clusterizadas de G e um conjunto de sensores mestres $M \subset V(F)$, onde $V(F) = \bigcup_{T \in F} V(T)$, tal que:

- Se $u \in V(F)$ e $u \notin M$, então u é nó ponte, i.e., $|\{v \in M : uv \in \bigcup_{T \in F} E(T)\}| > 1$.
- Se $u \notin V(F)$, então u é nó dominado, i.e., existe $v \in M$ tal que $uv \in E$;
- Para todos $u, v \in M$, $uv \notin E$.

O objetivo é *minimizar* o peso da árvore mais pesada em F , em que o peso c_T de uma árvore T é dado pela soma das arestas dessa árvore, i.e., $c_T = \sum_{uv \in T} c_{uv}$.

A Figura 1 mostra o exemplo de uma estrutura DCkF de uma rede com $k = 2$ com pesos todos unitários. Nós em preto representam sensores mestres, em cinza pontes e os demais são nós dominados. Em uma das árvores, identificamos dois clusters cujos mestres são o nó **j** e o nó **b**, com o nó **a** fazendo o papel de ponte. Na outra, temos apenas um cluster cujo nó mestre é **h**. Dizemos que o nó **j** domina os nós **g**, **i** e **d**, enquanto o nó **d** é dominado pelos nós **j** e **b**. As conexões que têm como uma das extremidades um nó dominado na floresta estão tracejadas, indicando que elas tem custo desprezível no cálculo do custo energético de cada árvore por conta dos sensores dominados permanecerem em estado de dormência, gastando muito pouca energia.

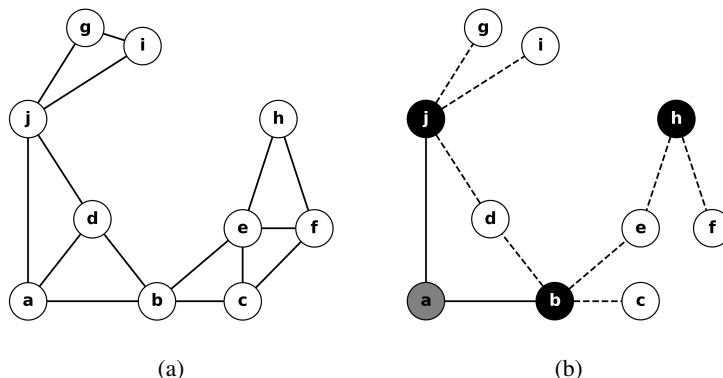


Figura 1: Exemplo de uma floresta clusterizada dominante com $k = 2$ em (b) para o grafo em (a).

3. Procedimento GRASP para o Problema DCKF

Propomos um procedimento baseado na meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*), originalmente introduzida por [Feo e Resende, 1995]. Essa abordagem iterativa é composta por duas fases principais: construção e busca local. Na etapa de construção, gera-se uma solução viável por meio de um processo guloso e randomizado, guiado por uma função de avaliação adaptativa que determina uma Lista Restrita de Candidatos (LRC) — um subconjunto dos candidatos mais promissores, cujo tamanho é controlado por um parâmetro que equilibra exploração e intensificação. A solução inicial construída é, então, aprimorada por uma busca local, que explora sua vizinhança com o objetivo de atingir um ótimo local. O procedimento é repetido por um número fixo de iterações e a melhor solução entre todas é mantida como resultado final.

O Algoritmo 1 descreve o esquema geral da meta-heurística GRASP adotada, enquanto o Algoritmo 2 detalha o procedimento de construção e seleção de elementos da LRC, que é feito através de um viés definido por uma função de probabilidade. O valor de cada solução é calculado pela função `Valor`, que retorna $+\infty$ se a solução for vazia, ou o peso da árvore de peso máximo caso contrário. As Tabelas 1 e 2 apresentam, respectivamente, as notações utilizadas e as entradas e parâmetros da meta-heurística.

3.1. Representação de uma Solução

Representamos uma solução por meio da tupla (F, M) , onde F corresponde à floresta formada pelas árvores da solução e M ao conjunto de nós identificados como mestres. Cada árvore $T_i \in F$ é descrita pelo par (V_i, E_i) , em que V_i representa o conjunto de vértices pertencentes à árvore e E_i o conjunto de arestas que a compõem. Por exemplo, a solução ilustrada na Figura 1 é expressa como $(\{T_1, T_2\}, \{j, b, h\})$. Nesse caso, a floresta F é formada por duas árvores: $T_1 = (\{a, j, b\}, \{aj, ab\})$, contendo três vértices e duas arestas, e $T_2 = (\{h\}, \emptyset)$, composta por um único vértice e nenhuma aresta. O conjunto $M = \{j, b, h\}$ indica que os vértices j , b e h atuam como mestres na solução.

3.2. Construção de uma Solução Inicial

O Algoritmo 3 descreve a fase de construção. Inicialmente, a floresta F e o conjunto de mestres M são vazios, e o conjunto de nós disponíveis V' , que representa os vértices não pertencentes a nenhuma árvore nem dominados por algum mestre, é inicializado com o conjunto de vértices de G (linha 1). Um nó é selecionado aleatoriamente a partir de uma LRC baseada na quantidade de vizinhos disponíveis e adicionado ao conjunto de mestres (linhas 3–5). Uma árvore trivial contendo apenas esse nó é então incorporada à floresta F (linha 6), e o conjunto V' é atualizado (linha 7).

Tabela 1: Notações adotadas.

Notação	Descrição
$P(a)$	Probabilidade de a .
$N_G(u) = \{v \in V(G) : uv \in E(G)\}$	Vizinhança aberta de $u \in V$ em G .
$N_G[u] = N_G(u) \cup \{u\}$	Vizinhança fechada de $u \in V$ em G .
$M(T)$	Nós mestres da árvore T .
$P(T)$	Nós pontes da árvore T .
$D(T)$	Nós dominados por pelo menos um mestre de T .

Tabela 2: Entradas e parâmetros da meta-heurística.

Parâmetro	Descrição
G	Grafo ponderado conexo representando uma WSN.
k	Número de aplicações (árvores).
i_1	Número de iterações do GRASP.
i_2	Número de iterações da busca local.
α_1	Percentual dos melhores candidatos incluídos na LRC durante a construção.
α_2	Percentual dos melhores candidatos incluídos na LRC durante a busca local.
f	Função de avaliação de candidatos.
l	Limite de vizinhos explorados sem melhora no valor da solução.

Algoritmo 1: GRASP ($G, k, i_1, i_2, \alpha_1, \alpha_2, f, l$)

```

1  $S^* \leftarrow \emptyset$ ;
2 for  $i \leftarrow 1$  to  $i_1$  do
3    $S \leftarrow \text{Construir}(G, k, \alpha_1, f)$ ;
4    $S \leftarrow \text{BuscaLocal}(G, S, i_2, \alpha_2, f, l)$ ;
5   if  $\text{Valor}(S) < \text{Valor}(S^*)$  then
6      $S^* \leftarrow S$ ;
7   if  $\text{Valor}(S^*) = 0$  then
8     break;
9 return  $S^*$ ;

```

Algoritmo 2: Sortear (L, α, f)

```

1 Ordene  $L$  de forma crescente com base nos valores de  $f$ ;
2  $L' \leftarrow$  os primeiros  $\lceil \alpha \cdot |L| \rceil$  elementos de  $L$ ;
3  $a \leftarrow$  sorteio em  $L'$  com  $P(b) = \frac{f(b)}{\sum_{c \in L'} f(c)}$ ;
4 return  $a$ ;

```

Esse processo (linhas 3–7) é repetido até que V' esteja vazio ou que o número de árvores em F seja igual a k . Por fim, a floresta resultante, composta inicialmente por árvores triviais, é expandida.

A etapa de expansão é detalhada no Algoritmo 4. Uma árvore $T \in F$ é selecionada aleatoriamente (linhas 2–6). Em seguida, uma tripla (u, v, w) é escolhida aleatoriamente a partir de uma LRC, considerando uma função de avaliação f (linha 7). A tripla (u, v, w) representa uma possível conexão entre um nó $u \in P(T) \cup D(T)$ — isto é, um nó do tipo ponte ou dominado da árvore T — e um nó disponível $v \in V'$. O valor de w depende do tipo de u : se u é uma ponte, w recebe o valor $|V(G)| + 1$; se u é dominado, w corresponde a um dos mestres que o dominam em T . O elemento w é essencial para permitir a avaliação de todas as possíveis conexões. O vértice v

é então classificado como mestre, e os nós u e v são adicionados à árvore T (linhas 8 e 9). Se u for dominado, adicionamos a aresta wu a T (linhas 10 e 11). Em seguida, adicionamos a aresta uv em T (linha 12). Por fim, o conjunto de nós disponíveis V' é atualizado (linha 13). O procedimento (linhas 2–13) é repetido até que não restem nós disponíveis. Foram implementadas quatro funções de avaliação distintas nessa etapa, conforme descrito na Tabela 3.

Algoritmo 3: Construir (G, k, α, f)

```

1  $F \leftarrow \emptyset; M \leftarrow \emptyset; V' \leftarrow V(G);$ 
2 while  $V' \neq \emptyset$  do
3    $N'_G(v) \leftarrow N_G(v) \cap V', \quad \forall v \in V';$ 
4    $u \leftarrow \text{Sortear}(V', \alpha, f'), \text{ com } f'(v) = |N'_G(v)|;$ 
5    $M \leftarrow M \cup \{u\};$ 
6    $T \leftarrow (\{u\}, \emptyset); F \leftarrow F \cup \{T\};$ 
7    $V' \leftarrow V' \setminus N_G[u];$ 
8   if  $|F| = k$  then
9     break;
10   $S \leftarrow (F, M);$ 
11  Expandir  $(G, S, V', \alpha, f);$ 
12 return  $S;$ 
```

Algoritmo 4: Expandir $(G, S = (F, M), V', \alpha, f)$

```

1 while  $V' \neq \emptyset$  do
2    $\forall T_i \in F, E'_1(T_i) \leftarrow \{(a, b, |V(G)| + 1) : ab \in E(G), a \in P(T_i), b \in V'\};$ 
3    $\forall T_i \in F, E'_2(T_i) \leftarrow \{(a, b, c) : ca, ab \in E(G), a \in D(T_i), b \in V', c \in M(T_i)\};$ 
4    $\forall T_i, E'(T_i) \leftarrow E'_1(T_i) \cup E'_2(T_i)$ 
5    $F' \leftarrow \{T_i \in F : E'(T_i) \neq \emptyset\};$ 
6   Sorteie  $T \in F'$  com  $P(T_i) = \frac{(c_{T_i})^{-1}}{\sum_{T_j \in F'} (c_{T_j})^{-1}};$ 
7    $(u, v, w) \leftarrow \text{Sortear}(E'(T), \alpha, f);$ 
8    $M \leftarrow M \cup \{v\};$ 
9    $V(T) \leftarrow V(T) \cup \{u, v\};$ 
10  if  $w \neq V(G) + 1$  then
11     $E(T) \leftarrow E(T) \cup \{wu\};$ 
12     $E(T) \leftarrow E(T) \cup \{uv\};$ 
13     $V' \leftarrow V' \setminus N_G[v];$ 
```

Tabela 3: Funções de avaliação utilizadas na expansão.

Função	Descrição
$f_1(u, v, w) = V' \cap N_G(v) $	Prioriza vértices com maior número de vizinhos disponíveis.
$f_2(u, v, w) = \begin{cases} (c_{uv})^{-1}, & \text{se } w = V(G) + 1 \\ (c_{wu} + c_{uv})^{-1}, & \text{caso contrário} \end{cases}$	Favorece conexões de menor custo.
$f_3(u, v, w) = \frac{f_1(u, v, w)}{f_2(u, v, w)}$	Equilibra cobertura e custo.
$f_4(u, v, w) = f_i(u, v, w), \text{ com } i \sim \text{Uniforme Discreta}(1, 3)$	Escolhe aleatoriamente uma das três funções anteriores.

3.3. Busca Local

Para a fase de busca local do GRASP, descrita no Algoritmo 5, propomos uma adaptação do algoritmo *Hill Climbing*. Trata-se de um método iterativo clássico que busca soluções melhores na vizinhança da solução atual [Michalewicz e Fogel, 2004]. Atribuímos o valor da solução inicial construída S a S^* e inicializamos o contador j com 0 (linha 1). Em seguida, realizamos um pré-processamento em S , obtendo a lista de candidatos L e o conjunto de nós disponíveis V' (linha 2). Caso L esteja vazia, a busca é encerrada (linhas 3 e 4). A partir desse ponto, inicia-se um laço (linhas 5–16) que modifica a solução S para alcançar um vizinho S' (linha 6), comparando-o com a melhor solução conhecida S^* . Quando uma solução melhor é encontrada, realiza-se seu pré-processamento, atualizando L e V' , e o contador j é reiniciado (linhas 7–10). A cada modificação sem melhoria, incrementa-se j em 1 (linhas 11 e 12). O laço é interrompido caso L esteja vazia (linhas 13 e 14), j atinja o limite de vizinhanças exploradas sem melhoria (linha 15) ou seja atingido o número máximo de iterações. Os algoritmos descritos a seguir detalham a estrutura de vizinhança utilizada na busca local.

Algoritmo 5: BuscaLocal (G, S, i_2, α, f, l)

```

1   $S^* \leftarrow S; j \leftarrow 0;$ 
2   $(L, V') \leftarrow \text{Pré-Processar}(G, S);$ 
3  if  $L = \emptyset$  then
4  |   return  $S^*$ 
5  for  $i \leftarrow 1$  to  $i_2$  do
6  |    $S' \leftarrow \text{Modificar}(G, S, L, V', \alpha, f);$ 
7  |   if  $\text{Valor}(S') < \text{Valor}(S^*)$  then
8  | |    $S^* \leftarrow S'; S \leftarrow S';$ 
9  | |    $(L, V') \leftarrow \text{Pré-Processar}(G, S);$ 
10 |    $j \leftarrow 0;$ 
11 |   if  $L = \emptyset$  then
12 | |   break
13 |   else
14 | |    $j \leftarrow j + 1;$ 
15 |   if  $j = l$  then
16 | |   break
17 return  $S^*$ 

```

O Algoritmo 6 realiza o pré-processamento da melhor solução em uma dada iteração da busca local. Inicialmente, selecionamos a árvore T de maior peso da floresta e a removemos da solução (linha 1). Em seguida, construímos o conjunto L contendo os nós que são pontes ou dominados e que são vizinhos exclusivamente dos mestres de T (linha 2). Um elemento deste conjunto será candidato a mestre, então essa restrição é importante para não violarmos a independência. Por fim, o conjunto de nós disponíveis é definido como L e o conjunto de nós mestres é atualizado, removendo os elementos pertencentes a T (linhas 3 e 4).

O Algoritmo 7 ilustra o processo para alcançar um vizinho de uma dada solução pré-processada. Inicialmente, S' recebe o valor da solução atual (linha 1). Em seguida, um nó é sorteado de uma LRC de L baseada no número de vizinhos disponíveis e classificado como mestre (linhas 2-4). A árvore trivial contendo esse nó é adicionada à solução e os nós disponíveis são atualizados (linhas 5-6). Por fim, a solução S' é expandida por meio do Algoritmo 4 (linha 7).

Algoritmo 6: Pré-Processar ($G, S = (F, M)$)

```

1  $T \leftarrow \arg \max_{T_i \in F} c_{T_i}; F \leftarrow F - T;$ 
2  $L \leftarrow \{v \in P(T) \cup D(T) : \{N_G(v) \cap M\} \subset M(T)\};$ 
3  $V' \leftarrow M(T) \cup L;$ 
4  $M \leftarrow M \setminus M(T);$ 
5 return ( $L, V'$ );

```

Algoritmo 7: Modificar (G, S, L, V', α, f)

```

1  $S' = (F', M') \leftarrow S;$ 
2  $N'_G(v) \leftarrow N_G(v) \cap V', \forall v \in V';$ 
3  $u \leftarrow \text{Sortear}(L, \alpha, f'), \text{ com } f'(v) = |N'_G(v)|;$ 
4  $M' \leftarrow M' \cup \{u\};$ 
5  $T \leftarrow \{\{u\}, \emptyset\}; F' \leftarrow F' \cup \{T\};$ 
6  $V' \leftarrow V' \setminus N_G[u];$ 
7 Expandir ( $G, S', V', \alpha, f$ );
8 return  $S'$ ;

```

4. Experimentos Computacionais

Os algoritmos foram implementados e testados no Google Colaboratory (Colab), plataforma baseada em notebooks hospedados na nuvem, que permite a execução de código Python diretamente no navegador. Utilizamos a versão gratuita que oferece uma CPU Intel Xeon (2 núcleos) com frequência de 2.20 GHz e 12 GB de memória RAM.

As implementações foram realizadas com o auxílio da biblioteca NetworkX [Hagberg et al., 2008], amplamente utilizada para modelagem, análise e manipulação de grafos em Python. Além disso, o modelo $(F)^+$ proposto em [Araujo et al., 2024] foi implementado no mesmo ambiente, utilizando a biblioteca MIP [Santos et al., 2020]. Para a resolução do modelo, foi adotado o solver Gurobi [Gur, 2024] em sua configuração padrão com um tempo limite de 600 segundos.

4.1. Instâncias

Grafos geométricos aleatórios (*Random Geometric Graphs* – RGGs) são amplamente empregados na modelagem e no projeto de redes de sensores sem fio, devido à sua capacidade de representar topologias espaciais realistas [Penrose, 2003]. Esses grafos são construídos posicionando-se os nós aleatoriamente em um espaço métrico, seguindo uma distribuição de probabilidade definida, e conectando dois nós sempre que a distância euclidiana entre eles for inferior a um raio de comunicação r . Os grafos utilizados nos experimentos são RGGs gerados em um quadrado de 100m^2 com $r = 25$. Embora a conectividade global não seja garantida, consideramos apenas casos em que o grafo resultante é conexo. Para esses grafos, consideramos valores de $k \in \{1, 2, 3, 4, 5\}$.

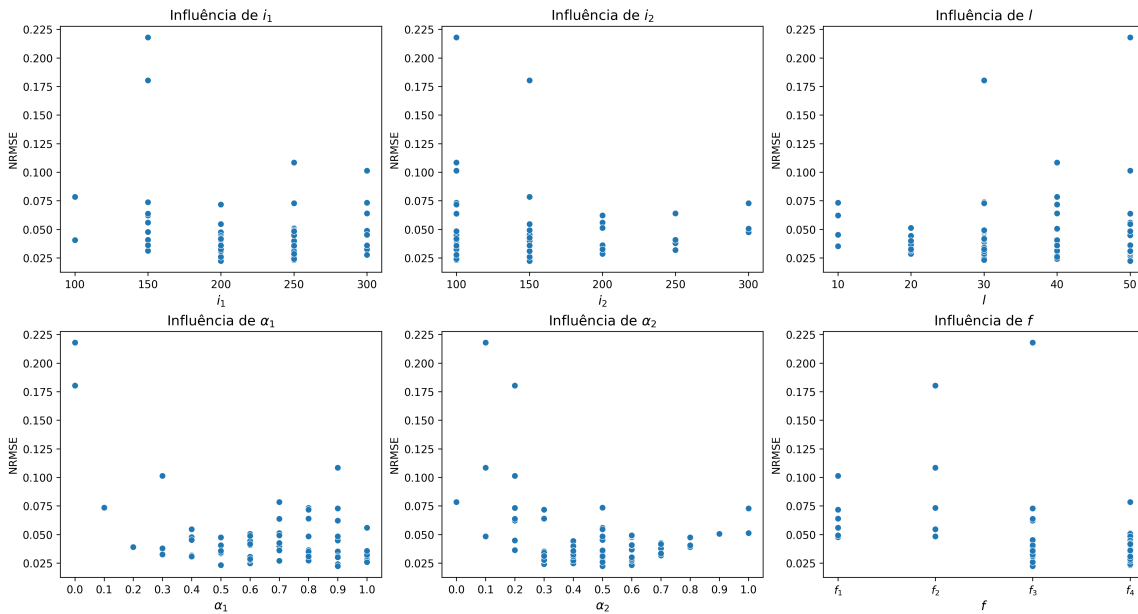
4.2. Calibração dos Parâmetros

Realizamos a calibração dos parâmetros da meta-heurística utilizando a biblioteca Optuna [Akiba et al., 2019]. Como métrica de avaliação, adotamos o erro quadrático médio normalizado (NRMSE). Utilizamos nesse processo apenas as instâncias que foram resolvidas otimamente pelo modelo $(F)^+$. Seja v_i o valor ótimo obtido pelo modelo $(F)^+$ para a instância $i \in \{1, 2, \dots, m\}$, e $\hat{v}_i$ o valor correspondente obtido pela meta-heurística. Seja $\bar{v}$ a média dos valores ótimos v_i . O NRMSE é calculado como $\frac{1}{\bar{v}} \sqrt{\frac{1}{m} \sum_{i=1}^m (v_i - \hat{v}_i)^2}$.

A Figura 2 apresenta os gráficos de influência individual de cada parâmetro sobre o desempenho da meta-heurística, gerados com a função `plot_slice` da biblioteca Optuna. No eixo x , são exibidos os valores testados para cada parâmetro durante a calibração, enquanto o eixo y mostra os valores do NRMSE obtidos nas respectivas execuções. Observamos que a quantidade

de iterações, bem como o limite de vizinhanças infrutíferas, não se configuram como fatores decisivos na obtenção de soluções de melhor qualidade. Em contrapartida, os parâmetros α_1 e α_2 demonstram maior influência: valores de α_1 iguais ou superiores a 0.5 estão associados a bons resultados, enquanto α_2 apresenta melhor desempenho quando próximo de 0.5. No que tange às funções de avaliação, aquelas associadas a f_3 e f_4 foram as que produziram os melhores resultados. A configuração que resultou no menor valor de NRMSE e, portanto, será adotada nas análises subsequentes, foi: $i_1 = 200$, $i_2 = 150$, $l = 50$, $\alpha_1 = 0.9$, $\alpha_2 = 0.5$ e $f = f_3$.

Figura 2: Influência dos parâmetros na meta-heurística.



4.3. Resultados

As Tabelas 4 e 5 apresentam os resultados obtidos a partir de 10 execuções da heurística para instâncias de pequeno e médio porte. Para garantir uma comparação justa com o modelo de referência, foi imposto ao Algoritmo 1 um limite de tempo de 600 segundos. As colunas n , id e k indicam, respectivamente, o número de sensores, o identificador do grafo e o número de árvores. As colunas $(F)^+$ e GRASP apresentam os resultados do modelo e do GRASP proposto, respectivamente. Em $(F)^+$, as colunas sol , $tempo$ e gap representam, respectivamente, o valor da solução, o tempo de execução (em segundos) e o gap do modelo. Já em GRASP, as colunas sol , $tempo$, gap e $melhora$ referem-se à melhor solução obtida nas 10 execuções, seu tempo de obtenção (em segundos), o gap em relação ao modelo e o percentual de melhoria no valor da solução inicial proporcionado pela busca local. Por fim, $\overline{sol}$ e $\overline{gap}$ indicam, respectivamente, a média do valor da solução e do gap ao longo das 10 execuções do GRASP. Casos em que o modelo não obteve uma solução viável dentro do limite de tempo são representados por “—”.

Nas instâncias de pequeno porte, observamos que o GRASP proposto igualou a solução do modelo $(F)^+$ em 26 das 30 instâncias testadas. Em 2 instâncias em que o modelo não obteve solução ótima, a heurística obteve melhores soluções. Notamos gaps positivos em relação ao modelo para a melhor solução obtida nas 10 execuções apenas para $k = 1$, em 2 instâncias, sendo o maior igual a 3,36%. Nos casos em que a heurística obteve tempo de execução menor, o que ocorreu em 26 das 30 instâncias, o tempo médio de execução foi 85% menor que o do modelo. Nas instâncias

de médio porte, observamos que, para aquelas com 200 sensores, o modelo $(F)^+$ nem sempre conseguiu obter uma solução dentro do tempo limite estabelecido. Entre as 14 instâncias em que o modelo encontrou soluções viáveis, o GRASP proposto apresentou melhores resultados em todos os casos com $k > 1$ (10 instâncias), com tempos de execução, em média, 94% menores que os do modelo. Considerando todas as instâncias, a busca local contribuiu com uma melhora percentual média de 55% no valor da solução inicial, o que mostra a eficácia da estrutura de vizinhança adotada. Nas instâncias com 200 nós e $k = 1$, o tempo de execução aumentou significativamente, chegando ao limite estipulado. Isso possivelmente se deve à estrutura de vizinhança utilizada, que nesse caso equivale a reconstruir uma nova solução do zero. Assim, pode ser interessante investigar se a busca local é realmente necessária para $k = 1$, talvez seja mais eficaz apenas aumentar o número de iterações sem aplicá-la — ou, alternativamente, explorar uma estrutura de vizinhança mais adequada para essa configuração.

Tabela 4: Resultados para instâncias de pequeno porte.

n	id	k	$(F)^+$			GRASP					
			sol	tempo	gap	sol	tempo	melhora	gap	$\overline{sol}$	$\overline{gap}$
30	1	1	180.19	2.08	0.00%	180.19	10.34	32.50%	0.00%	180.19	0.00%
		2	76.66	8.98	0.00%	76.66	6.00	48.34%	0.00%	76.66	0.00%
		3	38.09	15.12	0.00%	38.09	4.40	62.35%	0.00%	38.09	0.00%
		4	32.37	14.98	0.00%	32.37	5.94	26.31%	0.00%	32.37	0.00%
		5	0.00	0.14	0.00%	0.00	0.07	100.00%	0.00%	0.00	0.00%
	2	1	199.27	1.31	0.00%	199.27	13.64	23.19%	0.00%	199.27	0.00%
		2	95.66	65.42	0.00%	95.66	7.37	35.07%	0.00%	95.66	0.00%
		3	56.28	408.35	0.00%	56.28	6.05	49.37%	0.00%	56.28	0.00%
		4	29.80	1.45	0.00%	29.80	5.46	69.53%	0.00%	29.80	0.00%
		5	27.88	0.86	0.00%	27.88	5.50	65.78%	0.00%	27.88	0.00%
40	3	1	170.53	51.15	0.00%	170.53	21.88	38.72%	0.00%	174.42	2.28%
		2	85.50	600.01	35.00%	85.50	10.58	55.42%	0.00%	85.50	0.00%
		3	31.90	192.57	0.00%	31.90	6.38	55.14%	0.00%	42.88	34.41%
		4	28.64	112.56	0.00%	28.64	5.06	26.51%	0.00%	28.64	0.00%
		5	0.00	0.20	0.00%	0.00	0.05	100.00%	0.00%	0.00	0.00%
	4	1	201.81	145.72	0.00%	209.14	34.33	42.97%	3.63%	215.87	6.96%
		2	82.98	600.01	17.00%	82.98	16.63	61.81%	0.00%	85.70	3.28%
		3	57.30	600.01	41.00%	57.30	10.72	48.49%	0.00%	57.36	0.11%
		4	29.52	449.32	0.00%	29.52	9.18	77.84%	0.00%	29.52	0.00%
		5	27.16	279.20	0.00%	27.16	6.78	70.28%	0.00%	27.44	1.04%
50	5	1	247.96	377.36	0.37%	250.42	34.69	29.63%	0.99%	252.37	1.78%
		2	116.70	600.00	37.00%	113.58	15.42	43.92%	-2.67%	113.58	-2.67%
		3	62.12	600.01	41.00%	62.12	11.57	43.66%	0.00%	62.12	0.00%
		4	37.33	600.03	40.00%	37.33	10.02	65.36%	0.00%	37.33	0.00%
		5	29.17	600.01	11.00%	29.17	6.33	27.38%	0.00%	29.17	0.00%
	6	1	246.13	92.72	0.00%	246.13	37.70	32.24%	0.00%	255.73	3.90%
		2	118.53	600.02	30.00%	118.53	16.73	32.41%	0.00%	119.70	0.99%
		3	75.59	600.01	41.00%	69.55	11.55	43.61%	-7.99%	72.65	-3.89%
		4	54.10	600.01	54.00%	54.10	7.82	48.32%	0.00%	54.10	0.00%
		5	32.18	600.01	39.00%	32.18	8.09	57.25%	0.00%	32.73	1.71%

5. Conclusões

Neste trabalho, propomos uma abordagem baseada na meta-heurística GRASP para otimizar o consumo de energia em Redes Multitarefa de Sensores Sem Fio com topologia de floresta dominante clusterizada. Os resultados experimentais indicam que, para instâncias de pequeno porte, a heurística alcançou a solução obtida pelo modelo em 26 das 30 instâncias, com tempo de execução, em média, 85% menor. Para as instâncias de médio porte, o GRASP produziu melhores soluções

em todos os casos com $k > 1$, com tempo, em média, 94% inferior. A busca local mostrou-se muito eficiente, gerando melhoria média de 55% na qualidade das soluções, destacando a eficácia da estrutura de vizinhança. Como direções futuras, pretendemos investigar alternativas para tornar o caso $k = 1$ mais eficiente, explorar novas estratégias de busca local e integrar a heurística ao modelo matemático como valor de *cutoff*, bem como aplicar o procedimento GRASP para melhorar soluções viáveis obtidas em nós da árvore de busca durante a resolução do modelo.

Tabela 5: Resultados para instâncias de médio porte.

n	id	k	$(F)^+$			GRASP					
			sol	tempo	gap	sol	tempo	melhora	gap	$\overline{sol}$	$\overline{gap}$
100	7	1	224.10	600.02	46.99%	245.21	157.35	47.96%	9.42%	255.75	14.12%
		2	109.83	600.01	62.11%	108.66	41.39	50.15%	-1.07%	114.95	4.66%
		3	68.34	600.01	70.12%	60.57	24.99	52.14%	-11.37%	61.91	-9.41%
		4	53.42	600.01	78.98%	35.19	18.42	73.77%	-34.13%	36.23	-32.18%
		5	30.12	600.04	73.74%	27.43	14.15	74.66%	-8.93%	29.06	-3.51%
	8	1	220.89	600.01	50.17%	238.25	136.46	44.88%	7.86%	249.47	12.94%
		2	109.14	600.02	62.27%	100.86	44.58	48.80%	-7.59%	106.45	-2.46%
		3	68.91	600.01	69.25%	60.26	24.44	65.80%	-12.55%	61.98	-10.06%
		4	53.31	600.01	77.57%	34.17	19.80	76.30%	-35.90%	35.20	-33.98%
		5	29.86	600.02	72.71%	26.56	14.09	70.28%	-11.05%	27.83	-6.80%
200	9	1	231.94	600.03	78.64%	256.32	602.68	27.72%	10.51%	267.34	15.26%
		2	-	600.01	-	111.38	166.16	36.40%	-	120.26	-
		3	-	600.02	-	61.27	78.95	64.54%	-	62.82	-
		4	51.14	600.03	88.50%	34.37	55.00	59.83%	-32.79%	35.78	-30.03%
		5	-	600.02	-	25.89	40.91	69.27%	-	26.70	-
	10	1	241.64	600.03	76.13%	257.47	600.14	46.51%	6.55%	270.64	12.00%
		2	-	600.02	-	109.25	166.77	57.21%	-	118.67	-
		3	-	600.01	-	60.74	81.81	70.51%	-	63.18	-
		4	82.19	600.03	92.23%	37.87	52.66	69.84%	-53.92%	46.01	-44.02%
		5	-	600.01	-	25.96	38.66	68.40%	-	26.56	-

Agradecimentos

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) (Código de Financiamento 001), Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) (Proc. PGP-0192-00040.01.00/22, SPU No: 10583149/2022) e Conselho Nacional de Desenvolvimento Científico e Tecnológico (Proc. CNPq 303988/2021-5, 306492/2025-3). Agradecemos aos revisores pelos valiosos comentários e sugestões.

Referências

- Adasme, P. e de Andrade, R. C. (2023). Minimum weight clustered dominating tree problem. *European Journal of Operational Research*, 306(2):535–548.
- Akiba, T., Sano, S., Yanase, T., Ohta, T., e Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, p. 2623–2631. ACM.
- Araujo, T. G., Andrade, R. C., e Santos, A. C. (2024). A flow-based formulation for the multi-sink clustered wsn. In *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, p. 2235–2240.

- de Andrade, R. C., de Sousa, G. H., Duhamel, C. D., Santos, A. C., e Adasme, P. (2023). Modelo matemático para rede clusterizada de sensores sem fio para atividades compartilhadas de monitoramento. In *Anais do LV SBPO*, p. 1–12, São José dos Campos.
- Feo, T. e Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.
- Gendron, B., Lucena, A., Cunha, A., e Simonetti, L. (2014). Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing*, 26:645–657.
- Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, 2024.
- Hagberg, A. A., Schult, D. A., e Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conference (SciPy)*, p. 11–15.
- Islam, M., PhD, Hassan, M., Lee, G.-W., e Huh, E.-n. (2012). A survey on virtualization of wireless sensor networks. *Sensors*, 12:2175–2207.
- Kadri, A., Yaacoub, E., Mushtaha, M., e Abu-Dayya, A. (2013). Wireless sensor network for real-time air pollution monitoring. In *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, p. 1–5.
- Michalewicz, Z. e Fogel, D. B. (2004). *How to Solve It: Modern Heuristics*. Springer, Berlin, Heidelberg, 2 edition. ISBN 978-3-662-07281-8.
- Penrose, M. (2003). *Random Geometric Graphs*. Oxford studies in probability. Oxford University Press.
- Santos, A. C., Duhamel, C., Belisário, L., e Guedes, L. (2012). Strategies for designing energy-efficient clusters-based WSN topologies. *Journal of Heuristics*, 18:657–675.
- Santos, A. C., Duhamel, C., e Silva Belisário, L. (2016). Heuristics for designing multi-sink clustered wsn topologies. *Engineering Applications of Artificial Intelligence*, 50:20–31.
- Santos, F., Toffolo, T., e de Aragão, M. V. (2020). A python toolbox for modeling and solving mixed-integer linear programs. *Optimization Online*.
- Simonetti, L., Salles da Cunha, A., e Lucena, A. (2011). The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In Pahl, J., Reinert, T., e Voß, S., editors, *Network Optimization*, p. 162–169, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN 978-3-642-21527-8.
- Tang, V. W., Zheng, Y., e Cao, J. (2006). An intelligent car park management system based on wireless sensor networks. In *2006 First International Symposium on Pervasive Computing and Applications*, p. 65–70.
- Vlajic, N. e Xia, D. (2006). Wireless sensor networks: to cluster or not to cluster? In *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, p. 9 pp.–268.
- Çavdar, M. C., Korpeoglu, I., e Özgür Ulusoy (2022). Application placement with shared monitoring points in multi-purpose iot wireless sensor networks. *Computer Networks*, 217:109302.