

METAHEURÍSTICA GRASP APLICADA AO PROBLEMA p -HUB DE MÁXIMA COBERTURA

Warley Ferreira da Cunha ^{1,2}, Sérgio Ricardo de Souza ², Elisangela Martins de Sá ²

¹ Universidade Estadual de Montes Claros

Campus Universitário Prof. Darcy Ribeiro, Av. Prof. Rui Braga, s/n - Vila Mauriceia,
CEP: 39401-089, Montes Claros, MG, Brasil

² Centro Federal de Educação Tecnológica de Minas Gerais

Av. Amazonas, 7675, CEP: 30510-000, Belo Horizonte, MG, Brasil

warley.cunha@unimontes.br, sergio@cefetmg.br, elisangelamartins@cefetmg.br

RESUMO

Este trabalho apresenta um algoritmo baseado na metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) para o problema p -hub de máxima cobertura, não capacitado, com alocação única. O objetivo do problema é determinar a melhor localização para p -hubs e a atribuição de cada nó não hub a um único hub, de modo que a demanda total entre pares de nós dentro de uma determinada distância de cobertura seja maximizada. Testes computacionais usando instâncias da literatura foram realizados. Os resultados foram satisfatórios, com soluções de boa qualidade e tempos computacionais inferiores aos do *solver* CPLEX. Além disso, os resultados obtidos também foram comparados com resultados de referência da literatura, evidenciando um *gap* médio de 0,1% em relação aos valores da função objetivo e tempos médios de CPU inferiores em 100% das instâncias analisadas.

PALAVRAS CHAVE. Problema de máxima cobertura, Localização de p -hubs, Alocação única não capacitada. *GRASP*.

Tópicos: MH – Metaheurísticas, L&T – Logística e Transportes.

ABSTRACT

This work presents an algorithm based on the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic for the uncapacitated single-allocation maximum-coverage p -hub problem. The objective of the problem is to determine the best location for p -hubs and the assignment of each non-hub node to a single hub, so that the total demand between pairs of nodes within a given coverage distance is maximized. Computational tests performed using instances from the literature show satisfactory results, with good-quality solutions and runtimes lower than those of the CPLEX solver. Furthermore, the results obtained were also compared with reference results from the literature, evidencing an average gap of 0.1% to the objective function values and lower average CPU times in 100% of the instances analyzed..

KEYWORDS. Maximum coverage problem, p -hub location, Uncapacitated single allocation, *GRASP*.

Paper topics: MH - Metaheuristics, L&T – Logistics and Transport.

1. Introdução

Arquiteturas do tipo *hub-and-spoke* (HS) são amplamente utilizadas em projetos de redes de grande escala, como as adotadas em sistemas de telecomunicações, companhias aéreas de transporte de passageiros e cargas, serviços postais, dentre outras. Essas estruturas de rede, caracterizadas por fluxos entre diversos pares origem-destino (O–D), alcançam eficiência ao rotear o tráfego por meio de nós *hubs*. Esses *hubs* funcionam como instalações intermediárias para consolidação/desconsolidação, comutação e transbordo de fluxos, viabilizando a centralização das operações logísticas em detrimento de conexões diretas entre pares O–D. Dessa forma, a configuração HS possibilita a interligação de um grande número de nós com um número reduzido de arcos entre os *hubs*, o que resulta em uma diminuição significativa da infraestrutura necessária e dos custos operacionais [O’Kelly e Miller, 1994].

Desde as contribuições iniciais de O’Kelly [1986a,b, 1987], diversos trabalhos têm sido desenvolvidos sobre diferentes variações dos problemas HS. Esses estudos consideram aspectos como a presença ou ausência de restrições de capacidade nos *hubs*, a alocação única ou múltipla dos nós não-*hubs* aos *hubs*, a aplicação de fatores de desconto relacionados ao fluxo e a definição do número de *hubs* a serem instalados — seja esse valor previamente estabelecido ou determinado como parte da solução do problema.

Este trabalho trata de um caso particular do problema de redes HS, denominado Problema *p*-Hub de Máxima Cobertura Não Capacitado com Alocação Única (*Uncapacitated Single Allocation p-Hub Maximal Covering Problem* – USApHMCP). Classificado como NP-difícil [Kara e Tansel, 2003], o USApHMCP é mais complexo de resolver que outros problemas de localização de *hubs* [Silva e Cunha, 2017].

O objetivo é localizar *p hubs* e alocar cada nó não-*hub* a exatamente um *hub*, de modo a maximizar a demanda total entre pares de nós cuja distância não ultrapasse um limite pré-estabelecido. Assume-se que os arcos e os *hubs* não possuem restrições de capacidade; cada nó não *hub* é atribuído a um único *hub*; e, por fim, a rede de *hubs* é completa (ou seja, todos os pares de *hubs* estão conectados), contendo no caminho O–D no máximo dois *hubs*, visto que conexões diretas entre nós não *hubs* não são permitidas.

As primeiras formulações matemáticas para os problemas *p-hub* de máxima cobertura com alocação única (USApHMCP) e múltipla (UMApHMCP), propostas por Campbell [1994], assumiram o critério de cobertura binária. Hwang e Lee [2012] também estudaram o USApHMCP com o critério de cobertura binária e propuseram um novo modelo matemático, além de duas heurísticas para tratar o problema, uma baseada em distância e outra em volume de demanda, como alternativas à solução da formulação exata, que é inviável para instâncias de grandes dimensões. Peker e Kara [2015] propuseram novos modelos matemáticos para os problemas USApHMCP e UMapHMCP, que podem ser aplicados tanto nos casos de cobertura binária quanto de cobertura parcial por meio de uma função de decaimento.

Janković et al. [2017] propõem heurísticas eficientes baseadas na Busca em Vizinhança Variável (*Variable Neighborhood Search* – VNS) com o objetivo de resolver os problemas USApHMC e UMapHMCP, considerando critérios de cobertura binária e parcial. As heurísticas, classificadas como GVNS (*General VNS*) e BVNS (*Basic VNS*), mostraram desempenho superior às abordagens existentes, inclusive para instâncias de grande porte, fornecendo soluções de alta qualidade em tempos computacionais reduzidos.

Silva e Cunha [2017] propõem uma metaheurística de Busca Tabu (TS) para o USApHMCP. Os resultados computacionais demonstram a eficácia da heurística, proporcionando soluções melhores em tempos de processamento mais curtos quando comparada

aos resultados previamente publicados.

Diante disso, o objetivo deste trabalho é aplicar a metaheurística *Greedy Randomized Adaptive Search Procedures* (GRASP) com busca local em vizinhança variável para resolver o USApHMCP. Os resultados dos experimentos computacionais com o algoritmo proposto, utilizando as instâncias da literatura *Australian Post* (AP), serão comparados aos obtidos pelo *solver* CPLEX, bem como aos resultados de Silva e Cunha [2017].

O artigo está estruturado da seguinte forma. A Seção 2 apresenta a caracterização e o modelo matemático para o USApHMCP. A Seção 3 detalha o algoritmo proposto, enquanto os experimentos computacionais e seus resultados são detalhados na Seção 4. Considerações finais e sugestões para aprimoramento da abordagem adotada são discutidas na Seção 5.

2. Formulação do Problema e Modelo Matemático

Diversas formulações matemáticas para o problema de alocação de p -hubs podem ser encontradas na literatura. Especificamente, neste trabalho, adota-se a formulação proposta por Peker e Kara [2015].

Seja N o conjunto de nós, cuja cardinalidade será denotada por $n = |N|$. Um determinado número p de nós é selecionado como *hubs*, cujo conjunto é dado por H , isto é, $p = |H|$, $H \subseteq N$. Sejam c_{ij} e W_{ij} o custo unitário de transporte e o fluxo do nó de origem i até o nó destino j , respectivamente. Assume-se que um nó não *hub* pode usar apenas um nó *hub* para se comunicar com os outros nós. O custo de transporte do nó $i \in N$, atribuído ao *hub* $k \in H$, para o nó $j \in N$, atribuído ao *hub* $m \in H$, é dado por $\gamma c_{ik} + \alpha c_{km} + \delta c_{mj}$, sendo γ , α , δ taxas unitárias para coleta (origem-*hub*), transferência (*hub*-*hub*) e distribuição (*hub*-destino), respectivamente. O parâmetro α está associado à economia de escala de transporte *hub*-*hub*, portanto, espera-se $\alpha \leq \gamma$ e $\alpha \leq \delta$. O par (i, j) é coberto pelos *hubs* k e m se o custo de i para j via *hubs* k e m não excede o valor previamente definido β_{ij} (custo de transporte que indica o nível de cobertura para o par O-D). Veja a Figura 1.

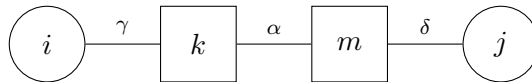


Figura 1: Trecho de uma rede *hub-and-spoke* para o par O-D. Assume-se que $\alpha \leq \gamma = \delta$

Seja a_{ij}^{km} um parâmetro do modelo tal que $a_{ij}^{km} = 1$ se $\gamma c_{ik} + \alpha c_{km} + \delta c_{mj} \leq \beta_{ij}$, e 0, caso contrário. Por fim, considere o parâmetro auxiliar $\lambda_{ij} = \max_{km} \{a_{ij}^{km}\}$, $\forall i, j \in N$. Defina, então, a variável binária de decisão x_{ij} , de modo que $x_{ij} = 1$ se o nó i for atribuído ao *hub* j , e 0, caso contrário. Além disso, $x_{ii} = 1$ se o nó i for *hub*, e 0, caso contrário. A variável de decisão não negativa z_{ij} representa a fração do fluxo roteado do nó de origem i para o nó de destino j que é coberto.

O USApHMCP pode ser formulado, portanto, como:

$$\max \sum_{i \in N} \sum_{j \in N} W_{ij} z_{ij} \quad (1)$$

$$\text{sujeito a: } \sum_{k \in N} x_{kk} = p \quad (2)$$

$$\sum_{k \in N} x_{ik} = 1, \forall i \in N \quad (3)$$

$$x_{ik} \leq x_{kk}, \forall i, k \in N \quad (4)$$

$$z_{ij} \leq \sum_{k \in N} a_{ij}^{km} x_{ik} + \lambda_{ij}(1 - x_{jm}), \forall i, j \in N, m \in N \quad (5)$$

$$x_{ij} \in \{0, 1\}, \forall i \in N, j \in M \quad (6)$$

$$z_{ij} \geq 0, \quad \forall i, j \in N. \quad (7)$$

A função objetivo (1) visa maximizar a demanda total coberta entre todos os pares de origem e destino. As restrições (2) garantem que exatamente p hubs sejam instalados, enquanto as restrições (3) impõem que cada nó i seja atribuído a exatamente um hub selecionado. As restrições (4) garantem que nenhum nó seja atribuído a um hub que não esteja aberto naquela localização. As restrições (5) determinam o limitante superior para a fração de fluxo entre os nós i e j que é coberta. Vale ressaltar que, devido às restrições (3), que garantem que cada nó j seja atribuído a exatamente um hub $m' \in N$ (ou seja, $x_{jm'} = 1$), as restrições (5) se reduzem a $z_{ij} \leq \sum_{k \in N} a_{ij}^{km} x_{ik} + \lambda_{ij}$ quando $x_{jm} = 0 \quad \forall m \neq m'$, ou $z_{ij} \leq \sum_{k \in N} a_{ij}^{km} x_{ik}$, para $x_{jm'} = 1$. Finalmente, as restrições (6) e (7) são as restrições de domínio das variáveis.

3. Descrição do Algoritmo Proposto

Esta seção descreve em detalhes o algoritmo utilizado para a resolução do problema abordado, abrangendo a representação e a construção da solução, a função de avaliação, a busca local, as estruturas de vizinhança e a metaheurística GRASP implementada.

3.1. Representação da Solução

Uma solução para o USApHMCP é representada por dois vetores de inteiros (s, h) . O vetor $s = (s_1, \dots, s_n)$ possui dimensão n , em que s_i indica o hub ao qual o nó i está alocado. O vetor h , de dimensão p , contém os hubs selecionados, em que $h = (h_1, \dots, h_p)$. A Figura 2 representa a topologia de uma rede hub-and-spoke com hubs nos nós $h = (7, 8, 9)$ e alocação dos demais nós conforme o vetor de alocação s .

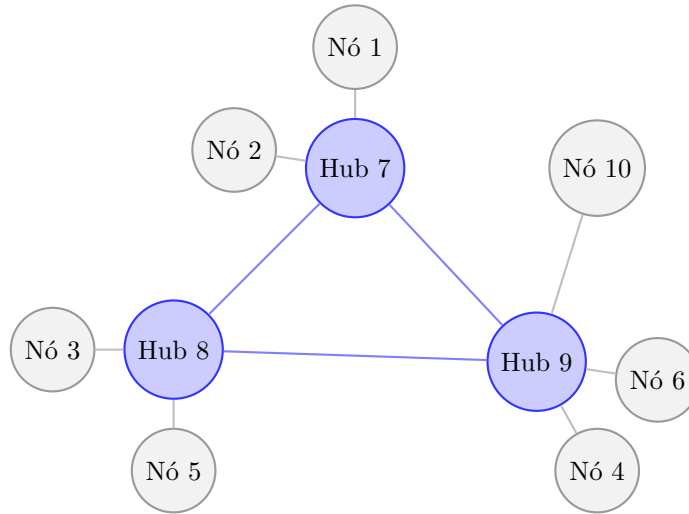


Figura 2: Representação de uma rede HS com hubs em $h = (7, 8, 9)$ e alocação de nós dada por $s = (7, 7, 8, 9, 8, 9, 7, 8, 9, 9)$.

3.2. Construção da Solução Inicial

No contexto do USApHMCP, a construção da solução inicial seguirá uma abordagem gulosa, selecionando como hubs os p nós com maior fluxo, definido pela soma das demandas originadas e destinadas ao nó. Em seguida, realiza-se uma alocação inicial gulosa, associando cada nó não hub ao hub mais próximo.

3.3. Função de avaliação

A função de avaliação avalia a qualidade de uma solução. Dada uma solução (s, h) , a função de avaliação utilizada é:

$$f(s, h) = \sum_{i \in N} \sum_{j \in N} W_{ij} a_{ij}^{s_i s_j}. \quad (8)$$

3.4. Estrutura de Vizinhança e Busca Local

A exploração do espaço de soluções do problema consiste em gerar uma nova solução a partir da solução atual (s, h) , aplicando um conjunto de movimentos. Estes movimentos geram as seguintes estruturas de vizinhança.

- (i) N_1 : gerada por **movimento de realocação**, que consiste em remover a alocação de um nó não *hub* a um *hub* e alocá-lo a outro *hub* aberto. Com este movimento, o número de vizinhos de (s, h) gerados é igual a $(n - p) \cdot (p - 1)$;
- (ii) N_2 : gerada por **movimento de substituição de *hub***, consiste em remover um *hub* aberto e escolher outro nó não *hub* para desempenhar a função de *hub*. Mantendo o número de *hubs* (p) fixo, o número de vizinhos gerados a partir desse movimento é $p(n - p)$;
- (iii) N_3 : gerada pelo **movimento de substituição de *hub* com busca local de realocação**, combina as características das estruturas N_2 e N_1 , permitindo gerar novas soluções a partir de três etapas: (i) fechar um *hub* atualmente aberto e abrir outro que estava fechado; (ii) alocação gulosa na solução parcial obtida na etapa anterior; (iii) aplicar uma busca local utilizando a vizinhança N_1 na solução gerada após a etapa de alocação. Embora o número de vizinhos gerados a partir deste movimento é o mesmo que em N_2 , a complexidade desta estrutura é maior que as outras, devido à alocação gulosa e à busca local.

A estratégia de busca local adotada neste trabalho baseia-se na abordagem *Variable Neighborhood Descent* (VND), mais especificamente em sua variação denominada VND Cíclico. Essa técnica explora diferentes estruturas de vizinhança de maneira sistemática e sequencial, sem reiniciar a ordem de exploração mesmo quando uma melhoria é encontrada. No VND Cíclico, parte-se de uma solução inicial (s, h) , composta pela alocação dos nós (s) e pela seleção dos *hubs* (h) , bem como de um valor inicial da função objetivo Fo . Um conjunto fixo de vizinhanças $\mathcal{N} = \{N_1, N_2, N_3\}$ é utilizado, sendo cada vizinhança aplicada sequencialmente à solução corrente.

A cada iteração, para cada vizinhança $N_j \in \mathcal{N}$, uma nova solução (s', h') é gerada por meio da aplicação do movimento correspondente. Caso essa nova solução apresente uma melhora na função objetivo ($f(s', h') > Fo$), a solução atual e o valor da função objetivo são atualizados. Esse processo é repetido por um número fixo de iterações, denotado por $k_{\max}$, conforme descrito no Algoritmo 1.

A principal característica do VND Cíclico é a continuidade na varredura das vizinhanças, mesmo após uma melhoria. Essa abordagem permite uma diversificação controlada do processo de busca, favorecendo a exploração do espaço de soluções sem comprometer a intensificação.

Algoritmo 1 Algoritmo de Busca Local - VND Cíclico

Entrada: Solução inicial (s, h) , valor da função objetivo Fo , número máximo de iterações $k_{\max}$, vizinhanças $\mathcal{N} = \{N_1, N_2, N_3\}$

```

1: para  $k = 1$  to  $k_{\max}$  faça
2:   para cada vizinhança  $N_j \in \mathcal{N}$  faça
3:      $(s', h') \leftarrow$  aplicar a vizinhança  $N_j$  sobre  $(s, h)$ ;
4:     se  $f(s', h') > f(s, h)$  então
5:        $(s, h) \leftarrow (s', h')$ ; ▷ Atualiza a solução
6:        $Fo \leftarrow f(s', h')$ ; ▷ Atualiza a  $F_O$ 
7:     fim se
8:   fim para
9: fim para

```

Saída: Solução final (s, h) e valor Fo

3.5. Metaheurística *Greedy Randomized Adaptive Search Procedures* (GRASP)

A metaheurística GRASP, proposta por Feo e Resende [1995], é um processo iterativo para problemas combinatórios, em que cada iteração consiste basicamente em duas fases: *Construção* e *Busca Local*.

A fase de construção da GRASP implementada neste trabalho adota uma estratégia parcialmente gulosa, controlada por um parâmetro $\alpha \in [0, 1]$, que regula o equilíbrio entre decisões totalmente gulosas, quando $\alpha = 0$, e escolhas completamente aleatórias, quando $\alpha = 1$. O objetivo dessa fase é selecionar um conjunto inicial de p hubs com base na contribuição de cada nó ao fluxo da rede, e posteriormente alocar cada nó ao hub mais próximo, com base na menor distância, formando, assim, a solução inicial (s, h) .

Para isso, inicialmente calcula-se, para cada nó $i \in N$, a soma dos fluxos com origem (O_i) e com destino (D_i), ou seja, $O_i = \sum_{j \in N} W_{ij}$ e $D_i = \sum_{j \in N} W_{ji}$. Os nós são então ordenados em ordem decrescente de $(O_i + D_i)$ e os valores $g_{\max}$ e $g_{\min}$ correspondem ao maior e ao menor valor dessa soma. Define-se um limite de corte $g = g_{\max} - \alpha(g_{\max} - g_{\min})$, e apenas os nós que satisfazem $(O_i + D_i) \geq g$ compõem a Lista de Candidatos Restrita (LCR).

Um nó é então selecionado aleatoriamente da LCR e adicionado à lista de hubs h . Esse processo se repete até que p hubs distintos tenham sido escolhidos. Após essa seleção, cada nó da rede é alocado ao hub mais próximo, considerando o menor custo de transporte, resultando no vetor de alocação s . A solução (s, h) obtida é então utilizada como ponto de partida para a fase de busca local da GRASP.

O Algoritmo 2 formaliza o processo de construção da solução inicial, seguindo a abordagem parcialmente gulosa proposta pela metaheurística GRASP.

O Algoritmo 3 representa o pseudocódigo do procedimento de busca local. Essa etapa explora as três estruturas de vizinhança apresentadas anteriormente. Após cada modificação, os nós não definidos como hubs são atribuídos ao hub mais próximo, com base no menor custo.

O pseudocódigo apresentado no Algoritmo 4 descreve o procedimento da metaheurística GRASP utilizada neste trabalho. O algoritmo recebe como entrada dois parâmetros principais: o valor de α , que regula o equilíbrio entre a escolha gulosa e aleatória na fase de construção, e $GRASP_{\max}$, que define o número máximo de iterações da metaheurística. Nos experimentos realizados, conduzimos testes preliminares com $\alpha = 0,4, 0,6, 0,8$ e 1 ,

Algoritmo 2 Construção($g(\cdot), \alpha$)

```

1:  $h \leftarrow \emptyset$ ;
2: Inicialize o conjunto  $LC$  de candidatos;
3: enquanto  $LC \neq \emptyset$  faça
4:    $g(t_{\min}) = \min\{g(t) \mid t \in LC\}$ ;
5:    $g(t_{\max}) = \max\{g(t) \mid t \in LC\}$ ;
6:    $LCR = \{t \in LC \mid g(t) \geq g(t_{\max}) - \alpha(g(t_{\max}) - g(t_{\min}))\}$ ;
7:   Selecione aleatoriamente um elemento  $t \in LCR$ ;
8:    $h \leftarrow h \cup \{t\}$ ;
9:    $LC \leftarrow LC \setminus \{t\}$ ;                                ▷ Atualiza o conjunto  $LC$  de candidatos
10: fim enquanto;
11: Construa  $s$  com base nos hubs  $h$ ;                            ▷ Atribuições dos nós via menor distância
12: Retorne  $(s, h)$ ;
13: fim Construção;

```

Algoritmo 3 BuscaLocal ($f(\cdot), N(\cdot), (s, h)$)

```

1:  $V \leftarrow \{(s', h') \in N(s, h) \mid f(s', h') > f(s, h)\}$ ;    ▷ Gera vizinhos com melhoria
2: enquanto  $|V| > 0$  faça
3:   Selecione  $(s', h') \in V$ ;                                    ▷ Escolhe um vizinho com melhoria
4:    $(s, h) \leftarrow (s', h')$ ;                                    ▷ Atualiza a solução atual
5:    $V \leftarrow \{(s', h') \in N(s, h) \mid f(s', h') > f(s, h)\}$ ;    ▷ Nova vizinhança
6: fim enquanto;
7: retorne  $(s, h)$ ;                                              ▷ Retorna a solução localmente ótima
8: fim BuscaLocal;

```

além de $GRASP_{\max} = 3$ e 30. Com base no desempenho obtido, os valores foram definidos empiricamente como $\alpha = 0,8$ e $GRASP_{\max} = 3$, de modo a equilibrar a qualidade das soluções e o tempo de execução computacional.

Por fim, o Algoritmo 5 sintetiza todo o procedimento empregado na resolução do problema USApHMCP.

4. Experimentos Computacionais

Esta seção apresenta os resultados dos experimentos computacionais. Ela está organizada na seguinte forma. A Subseção 4.1 descreve os conjuntos de instâncias utilizados nos experimentos realizados; a Subseção 4.2 apresenta as tabelas com os resultados obtidos nos experimentos computacionais.

4.1. Instâncias da Literatura

Para a determinação dos resultados computacionais, foi utilizado o conjunto de dados da literatura *Australian Post* (AP). Este conjunto de dados, disponibilizado na *OR Library*, que tem até 200 nós, fornece o número de nós n , o número de hubs p , as coordenadas cartesianas (x_i, y_i) dos nós, as demandas w_{ij} de cada nó e as taxas unitárias para coleta γ (origem-hub), transferência α (hub-hub) e distribuição δ (hub-destino). A notação APn_p refere-se a uma instância desse conjunto de dados AP, composta por n nós e p hubs. Neste trabalho, foi adotado $\gamma = \delta = 1$ e $\alpha = 0,75$.

Dado que as instâncias AP não disponibilizam explicitamente os valores de β_{ij} , neste trabalho considera-se, para tais parâmetros, os resultados da função objetivo obtidos

Algoritmo 4 GRASP ($f(\cdot)$, $g(\cdot)$, $N(\cdot)$, $GRASP_{\max}$, α , (s, h))

```

1:  $f^* \leftarrow -\infty$ ;
2: para Iter = 1 até  $GRASP_{\max}$  faça
3:    $(s, h) \leftarrow \text{Construção}(g(\cdot), \alpha)$ ;
4:    $(s, h) \leftarrow \text{BuscaLocal}(f(\cdot), N(\cdot), (s, h))$ ;
5:   se  $f(s, h) > f^*$  então
6:      $(s^*, h^*) \leftarrow (s, h)$ ;
7:      $f^* \leftarrow f(s, h)$ ;
8:   fim se
9: fim para
10:  $(s, h) \leftarrow (s^*, h^*)$ ;
11: retorne  $(s, h)$ ;
12: fim GRASP;

```

Algoritmo 5 Procedimento aplicado ao USApHMCP

Entrada: Instâncias APs , α , β_{ij}

- 1: Para cada nó $i \in N$, calcule o fluxo total com origem (O_i) e com destino (D_i) no nó i :

$$O_i = \sum_{j \in N} W_{ij} \text{ e } D_i = \sum_{j \in N} W_{ji}.$$
 - 2: Ordene os nós $i \in N$ em ordem decrescente de $(O_i + D_i)$.
 - 3: Selecione os p primeiros nós com maior fluxo de entrada e saída como *hubs*.
 - 4: Gere uma solução inicial usando alocação por menor distância e calcule o valor de função objetivo.
 - 5: Execute a metaheurística GRASP – Algoritmo 4.
-

na resolução do problema p -hub centro de alocação única não capacitada (*Uncapacitated Single Allocation p-Hub Center Problem* - USApHCP), valor de referência amplamente adotado na literatura. Os valores correspondentes foram fornecidos por Peker e Kara [2015], para instâncias com $n \geq 25$, e por Meyer et al. [2009], para instâncias com $n < 25$. Ressalta-se que os mesmos valores de β_{ij} foram utilizados em Silva e Cunha [2017], artigo adotado como base comparativa na seção de resultados computacionais.

Todos os experimentos computacionais deste trabalho foram realizados em um *notebook* com sistema operacional *Windows* 11, processador 11ª Geração do Intel(R) Core(TM) i7 – 1165G7 2.80 GHz e 16 GB de memória RAM. O algoritmo GRASP foi implementado na linguagem C++ e, para o cálculo das soluções exatas, foi utilizado o solver CPLEX 22.1.1, configurado com uma *thread*, limite de tempo igual a 86400 segundos e demais configurações padrão, meio da biblioteca *Concert Technology* usando a linguagem C++.

4.2. Resultados dos Experimentos

A Tabela 1 mostra os resultados dos experimentos computacionais realizados com as instâncias AP contendo até 200 nós. A primeira parte da tabela apresenta os valores ótimos (OptSol) da função objetivo (Fo) e os respectivos tempos de execução em segundos (CPUt), conforme reportado pelo *solver* CPLEX. Já na coluna TS, encontram-se os resultados obtidos pela heurística Busca Tabu (*Tabu Search* - TS), proposta por Silva e Cunha [2017]. Os valores exibidos correspondem ao *gap* percentual em relação ao ótimo e ao tempo de execução, em segundos. Por fim, na última coluna da tabela, são apresentados os resultados da metaheurística proposta neste trabalho. Nessa coluna, são informados o *gap*

médio ($\overline{gap}$), o gap referente ao melhor valor de Fo encontrado (gap^*), o menor tempo para a obtenção desse melhor valor ($CPUt^*$) e o tempo médio de execução ($\overline{CPUt}$), calculado com base nas 30 execuções do algoritmo GRASP.

Para o cálculo do gap médio, em valores percentuais, tem-se:

$$\overline{gap}(\%) = \frac{\text{valor médio} - \text{OptSol}}{\text{OptSol}} \times 100$$

e, para calcular o gap^* , utiliza-se o melhor valor de Fo encontrado em vez do valor médio.

Como os experimentos computacionais deste trabalho foram executados em uma máquina distinta daquela utilizada por Silva e Cunha [2017], os tempos de execução foram normalizados com base nos valores do CPU Mark, conforme disponibilizado pela ferramenta PassMark <https://www.cpubenchmark.net/singleCompare.php>. Os valores de desempenho das CPUs correspondentes são 24,338 para o equipamento utilizado por Silva e Cunha [2017] e 10,110 para o utilizado neste estudo. Assim, os tempos computacionais apresentados na Tabela 1 foram ajustados mediante a aplicação do fator de correção $24.338/10.110 = 2,4073$ aos tempos originalmente reportados por Silva e Cunha [2017].

Com base nos dados apresentados na Tabela 1, constata-se que a meta-heurística GRASP implementada demonstrou desempenho altamente competitivo em relação ao algoritmo de Busca Tabu (TS), proposto por Silva e Cunha [2017]. Para as instâncias com até 100 nós, observa-se que a GRASP foi capaz de obter soluções ótimas em diversas execuções, apresentando valores nulos de gap ($gap^* = 0$), evidenciados em negrito na tabela. Em alguns casos, inclusive, o gap médio ($\overline{gap}$) também foi igual a zero, o que reforça a consistência e a robustez da abordagem proposta.

Ressalta-se que as maiores instâncias (com 100 e 200 nós) não puderam ser resolvidas pelo solver CPLEX, uma vez que sua execução foi automaticamente interrompida pelo sistema operacional devido à elevada demanda por memória. Durante a execução dessas instâncias, observou-se a finalização abrupta dos processos, ocasionada pela insuficiência de recursos computacionais, o que resultou no encerramento forçado da execução. Dessa forma, na Tabela 1 utilizamos o símbolo “–” para indicar ausência dos valores de OptSol e CPUt. No entanto, para o cálculo do gap dessas instâncias, considerou-se o valor de cobertura máxima igual a 3978,92. Tal limitação evidencia o elevado custo computacional exigido pelo modelo exato em instâncias com grande número de nós, reforçando, assim, a necessidade da utilização de métodos heurísticos para lidar de forma eficiente com problemas dessa magnitude.

5. Considerações Finais

Os resultados dos experimentos computacionais, com a aplicação da metaheurística GRASP ao problema USApHMCP, utilizando as instâncias APs, evidenciam um desempenho notável na obtenção de soluções ótimas ou de soluções com valores de função objetivo bastante próximos do ótimo, com tempos computacionais substancialmente inferiores aos obtidos pelo *solver* CPLEX.

Além disso, a metaheurística GRASP proposta demonstrou-se altamente competitiva em relação à metaheurística de Busca Tabu (TS), presente na literatura, destacando-se nos seguintes aspectos: (i) qualidade das soluções – em aproximadamente 40,0% das instâncias analisadas (11 de 28), a heurística GRASP obteve soluções ótimas ou com valores de gap percentual extremamente reduzidos; (ii) estabilidade – os baixos valores de gap médio, bem como os casos em que o gap médio coincidiu com o gap^* , indicam que a heurística GRASP apresenta desempenho consistente ao longo de múltiplas execuções, sem depender

Tabela 1: Resultados do *solver* CPLEX e metaheurística GRASP em comparação com o algoritmo de Busca Tabu (TS) proposto por Silva e Cunha [2017] usando as instâncias AP.

Instância AP_{n_p}	β_{ij}	CPLEX		TS		GRASP			
		OptSol	CPUt	gap	CPUt	$\overline{gap}$	gap*	CPUt*	$\overline{CPUt}$
AP10 ₂	40383	3978,92	0,01	0	0,036	0	0	0,001	0,001
AP10 ₃	34772	3937,3	0,02	0	0,120	0	0	0,001	0,002
AP10 ₄	32574	3954,53	0,02	0	0,235	0,85	0	0,002	0,002
AP10 ₅	32531	3954,53	0,01	0	0,317	0	0	0,001	0,001
AP20 ₂	45954	3973,21	0,38	0	0,469	0,12	0,12	0,006	0,008
AP20 ₃	43400	3973,2	0,46	0	1,442	0,46	0,43	0,009	0,013
AP20 ₄	38607	3974,27	0,11	0	2,910	0,55	0	0,018	0,022
AP20 ₅	37868	3973,2	0,16	0	4,838	0,10	0	0,021	0,031
AP25 ₂	53207	3976,57	0,86	0	1,112	0,04	0,04	0,008	0,012
AP25 ₃	46608	3972,51	1,79	0	3,543	0,29	0,27	0,017	0,020
AP25 ₄	45552	3976,68	0,66	0	7,089	0	0	0,038	0,054
AP25 ₅	45552	3976,68	0,61	0	12,169	0	0	0,049	0,067
AP40 ₂	61683	3978,92	65,31	0	7,915	0,02	0,02	0,046	0,055
AP40 ₃	58193	3978,92	1,99	0	24,391	0,03	0,02	0,102	0,128
AP40 ₄	52265	3977,28	9,14	0	50,127	0,01	0	0,181	0,229
AP40 ₅	49741	3977,97	10,44	0	85,751	0,06	0,02	0,293	0,302
AP50 ₂	65523	3978,69	2407,17	0	19,015	0	0	0,097	0,120
AP50 ₃	60132	3978,42	450,38	0	58,153	0,01	0,01	0,227	0,269
AP50 ₄	52906	3978,92	4,07	0	118,083	0,08	0,06	0,452	0,469
AP50 ₅	50708	3978,92	3,64	0	199,489	0,07	0,06	0,714	0,750
AP100 ₂	65915	–	–	0	19,015	0,03	0	1,303	1,337
AP100 ₃	60659	–	–	0	58,153	0,02	0,01	3,799	3,918
AP100 ₄	56125	–	–	0	118,083	0,02	0,01	7,617	7,790
AP100 ₅	54243	–	–	0	199,489	0,02	0,01	12,652	13,479
AP200 ₂	68232	–	–	0	6617,721	0,01	0,01	20,892	22,561
AP200 ₃	64237	–	–	0	18482,797	0,01	0,01	61,471	72,181
AP200 ₄	59999	–	–	0	38488,199	0,01	0,01	136,020	174,364
AP200 ₅	58562	–	–	0	64351,212	0,01	0,01	219,706	230,159
Média	–	–	–	0	4604,710	0,10	0,04	16,634	18,87

de soluções pontuais ou esporádicas; (iii) Eficiência computacional – em todas as instâncias analisadas, a GRASP apresentou tempos médios de execução significativamente inferiores aos da TS. Na instância AP200₅, por exemplo, o tempo da TS foi aproximadamente 280 vezes superior ao da GRASP, evidenciando uma economia computacional da ordem de centenas de vezes. De forma geral, o tempo médio reportado por Silva e Cunha [2017] é de 4604,710 segundos, o que mostra que, em média, a heurística proposta executa 244 vezes mais rápido do que a TS proposta por aqueles autores. Além disso, a heurística proposta se destaca em escalabilidade, uma vez que nas instâncias de maior porte (com 100 e 200 nós), onde o CPLEX não alcançou soluções viáveis dentro do tempo limite, a GRASP manteve um bom desempenho, tanto em relação à qualidade das soluções quanto ao tempo de execução.

Como trabalhos futuros, sugere-se a utilização de versões adaptativas da metaheurística GRASP, como a *Reactive* GRASP, e a hibridização com outras metaheurísticas, uma vez que os resultados obtidos indicam que, apesar do bom desempenho da GRASP, há casos em que a exploração adaptativa de parâmetros (como no *Reactive* GRASP) e a combinação de estratégias complementares podem acelerar a convergência, evitar a parada prematura

em ótimos locais e aprimorar soluções para instâncias mais complexas. Além disso, análises estatísticas e testes em instâncias maiores podem contribuir para validar e expandir os resultados obtidos.

Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, processo 409785/2023-7), à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG, processo APQ-02473-24), ao Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) e a Universidade Estadual de Montes Claros (UNIMONTES) pelo apoio ao desenvolvimento deste trabalho.

Referências

- Alumur, S. e Kara, B. Y. (2008). Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21.
- Alumur, S. A., Campbell, J. F., Contreras, I., Kara, B. Y., Marianov, V., e O’Kelly, M. E. (2021). Perspectives on modeling hub location problems. *European Journal of Operational Research*, 291(1):1–17.
- Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 2(72):387–405.
- Farahani, R. Z., Hekmatfar, M., B., A. A., e Nikbakhsh, E. (2013). Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & Industrial Engineering*, 64:1096–1109.
- Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133.
- Gendreau, M. e Potvin, J.-Y., editors (2010). *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, 2 edition.
- Hsieh, S.-Y. e Kao, S.-S. (2019). A survey of hub location problems. *Journal of Interconnection Networks*, 19(01):1940005.
- Hwang, Y. H. e Lee, Y. H. (2012). Uncapacitated single allocation p-hub maximal covering problem. *Computers & Industrial Engineering*, 63(2):382–389.
- Janković, O., Mišković, S., Stanimirović, Z., e Todosijević, R. (2017). Novel formulations and vns-based heuristics for single and multiple allocation p-hub maximal covering problems. *Annals of Operations Research*, 259:191–216.
- Janković, O. e Stanimirović, Z. (2017). A general variable neighborhood search for solving the uncapacitated r-allocation p-hub maximal covering problem. *Electronic Notes in Discrete Mathematics*, 58:23–30.
- Kara, B. Y. e Tansel, B. C. (2003). The single-assignment hub covering problem: Models and linearizations. *The Journal of the Operational Research Society*, 54(1):59–64.

- Karimi, H. e Bashiri, M. (2011). Hub covering location problems with different coverage types. *Scientia Iranica*, 18(6):1571–1578.
- Meyer, T., Ernst, A. T., e Krishnamoorthy, M. (2009). A 2-phase algorithm for solving the single allocation p-hub center problem. *Computers & Operations Research*, 36:3143–3151.
- O’Kelly, M. (1986a). Activity levels at hub facilities in interacting networks. *Geographical Analysis*, 18(4):343–356.
- O’Kelly, M. E. (1986b). The location of interacting hub facilities. *Transportation Science*, 20(2):92–106.
- O’Kelly, M. E. (1987). A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404.
- O’Kelly, M. E. e Miller, H. J. (1994). The hub network design problem: A review and synthesis. *Journal of Transport Geography*, 2(1):31–40.
- Peker, M. e Kara, B. Y. (2015). The p-hub maximal covering problem and extensions for gradual decay functions. *Omega*, 54:158–172.
- Silva, M. R. e Cunha, C. B. (2017). A tabu search heuristic for the uncapacitated single allocation p-hub maximal covering problem. *European Journal of Operational Research*, 262:954–965.
- Yaman, H. (2011). Allocation strategies in hub networks. *European Journal of Operational Research*, 211(3):442–451.