LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

ALGORITMO *ADAPTIVE LARGE NEIGHBORHOOD SEARCH* APLICADO AO PROBLEMA *p-HUB* CENTRO NÃO CAPACITADO COM ALOCAÇÃO MÚLTIPLA

Alexandre Sidney Ferreira, Elisangela Martins de Sá, Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas, 7675, CEP: 30510-000, Belo Horizonte, MG, Brasil
alexandre.sidney@gmail.com, elisangelamartins@cefetmg.br, sergio@cefetmg.br

RESUMO

Este artigo apresenta um algoritmo baseado na meta-heurística *Adaptive Large Neighborhood Search* para solucionar o problema *p-hub* centro não capacitado com alocação múltipla. O objetivo deste problema é selecionar p nós para se tornarem *hubs* e alocar nós não *hubs* aos *hubs*, de modo que o custo máximo de atendimento das demandas, considerando os custos associados a todos os pares origem-destino, seja minimizado. Este problema pode ser aplicado no planejamento de um sistema de distribuição envolvendo produtos perecíveis ou produtos sensíveis ao fator tempo, como cargas urgentes. Experimentos computacionais foram realizados para comparar o tempo de execução e do valor de função objetivo do algoritmo implementado com os resultados disponíveis na literatura. As avaliações indicam que o algoritmo é capaz de encontrar boas soluções em um tempo computacional adequado.

PALAVRAS CHAVE. Problemas de Localização de *Hubs*, *p-Hub* Centro, *Adaptive Large Neighborhood Search*, Meta-heurísticas.

Tópicos (Logística e Transportes. Meta-heurística.)

ABSTRACT

This article presents an algorithm based on the Adaptive Large Neighborhood Search metaheuristic to solve the uncapacitated multiple allocation p-hub center problem. The objective of this problem is to select p nodes to become hubs and allocate non-hub nodes to the hubs, so that the maximum cost of meeting the demands, considering the costs associated with all origin-destination pairs, is minimized. This problem can be applied to the planning of a distribution system involving perishable products or time-sensitive products, such as urgent cargo. Computational experiments are carried out to compare the execution time and objective function value of the proposed algorithm in relation to results found in the literature. The performed evaluations show that the algorithm provides good solutions in an adequate computational time.

KEYWORDS. Hub Location Problems, p-Hub Center, Adaptive Large Neighborhood Search, Metaheuristics.

Paper topics (Logistics and Transport. Metaheuristic)

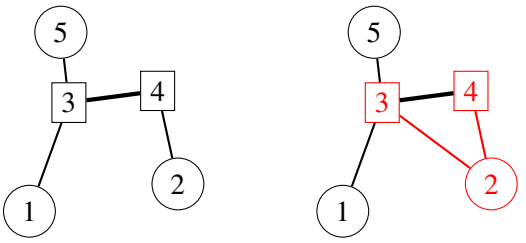
DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

1. Introdução

Em um sistema que envolve a distribuição de produtos perecíveis ou cargas urgentes, sensíveis ao fator tempo, o desafio fundamental enfrentado é a redução do tempo de transporte, além da redução de custo. O tempo de transporte está diretamente relacionado com a distância a ser percorrida entre a origem e o destino do produto. Neste contexto, a utilização de um sistema de transporte com topologia de rede eixo-raio pode se mostrar eficiente.

A topologia de rede eixo-raio é caracterizada pela presença de *hubs* e pode ser utilizada para reduzir os tempos de entrega e melhorar a eficiência operacional, resultando em menores custos de transporte. Por exemplo, ao transportar produtos da origem ao destino por meio de conexões entre *hubs*, a maior velocidade nessas rotas pode diminuir significativamente o tempo total de transporte. Além disso, demandas provenientes de várias origens para um ou mais destinos podem ser agrupadas e transportadas de forma conjunta usando a mesma conexão entre *hubs*, aumentando a eficiência operacional e reduzindo os custos de transporte.

Essa topologia pode ser classificada conforme o tipo de alocação entre nós não *hubs* e *hubs*, que pode ser única ou múltipla. Na alocação única, cada nó não *hub* é alocado a um único nó *hub*. Enquanto que na alocação múltipla, cada nó não *hub* pode ser alocado a um ou mais nós *hubs*. A Figura 1 ilustra uma rede eixo-raio para cada tipo de alocação.



(a) Alocação Única (b) Alocação Múltipla

Figura 1: Ilustração de redes eixo-raio com os dois tipos de alocação.

Este artigo aborda o problema *p-hub* centro não capacitado com múltiplas alocações (*uncapacitated multiple allocation p-hub center problem* – UMApHCP), uma extensão do problema pHCP proposto por Campbell [1994]. O UMApHCP visa definir p nós para se tornarem *hubs* e alocar nós não *hubs* a esses *hubs*, com o objetivo de minimizar o máximo custo de transporte entre todos os pares origem-destino, projetando assim uma rede do tipo eixo-raio. Neste problema, os nós *hubs* não possuem restrições de capacidade (não capacitado) e os nós não *hubs* podem ser alocados a mais de um *hub* (alocação múltipla).

Diversos estudos envolvendo o pHCP e suas variantes são encontrados na literatura. Campbell et al. [2007] propuseram o problema *p-hub* centro com alocação múltipla, apresentando suas características e uma formulação inicial. Ernst et al. [2009] desenvolveram duas novas formulações de otimização inteira e uma abordagem *branch-and-bound* para resolver o problema com alocação múltipla, além de demonstrarem que o problema é NP-Difícil.

Devido à complexidade do UMApHCP, vários estudos propuseram heurísticas para sua resolução. Brimberg et al. [2017] implementaram a meta-heurística Basic Variable Neighborhood Search (BVNS) para resolver instâncias do problema, comparando os resultados com modelos matemáticos do UMApHCP, e evidenciaram as limitações desses modelos e a superioridade da BVNS. Mišković e Stanimirović [2017] propuseram uma meta-heurística híbrida que combina Particle Swarm Optimization (PSO) com busca local. Silva et al. [2019b] desenvolveram uma abordagem baseada em Arquitetura Híbrida Multiagente (AMAM), enquanto Silva et al. [2019a] aplicaram

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

algoritmos genéticos com técnicas de aprendizado. Além disso, Brimberg et al. [2022] introduziram uma heurística baseada na meta-heurística General Variable Neighborhood Search (GVNS), e Ferreira et al. [2023] utilizaram a meta-heurística *Iterated Local Search* (ILS).

Neste trabalho, utilizamos a meta-heurística *Adaptive Large Neighborhood Search* (ALNS) para resolver instâncias do UMApHCP, devido à sua promissora utilização em problemas com vizinhança grandes e sua capacidade de adaptação.

O restante deste artigo está estruturado da seguinte forma: a Seção 2 apresenta o modelo matemático e a caracterização do problema. A Seção 3 descreve o procedimento de resolução implementado. Os experimentos computacionais e seus resultados são discutidos na Seção 4. Por fim, a Seção 5 apresenta as considerações finais e as direções futuras para pesquisa.

2. Caracterização do Problema

O UMApHCP consiste em definir p nós para se tornarem *hubs* e a alocação dos nós não *hubs* a esses *hubs* com o objetivo de minimizar o custo máximo de transporte entre todos os pares origem-destino. Neste problema, os nós *hubs* são considerados não capacitados, ou seja, não há restrições de capacidade para esses nós. Além disso, os nós não *hubs* podem ser alocados a mais de um *hub* (alocação múltipla). Cada nó não *hub* deve ser alocado a pelo menos um nó *hub*, não sendo permitido o transporte direto entre nós que não são *hubs*. Por fim, exatamente p *hubs* devem ser instalados.

O problema pode ser descrito da seguinte forma. Seja $G = (N, A)$ um grafo, em que N é um conjunto com n nós e $A = \{(i, j) \mid i, j \in N\}$ representa o conjunto de pares de nós. Além disso, assume-se que todo nó $i \in N$ é um candidato a se tornar um *hub*. Para cada par de nós $(i, j) \in A$, d_{ij} representa a distância entre os nós i e j. O parâmetro α é o fator de desconto de transporte entre *hubs*, que representa a economia de escala obtida pela concentração de fluxo nos *hubs*. A aplicação desse fator é apresentada na Figura 2. Desta forma, o custo de transporte do nó i ao nó j passando pelos *hubs* $k \in N$ e $l \in N$ é dado por $d_{ik} + \alpha d_{kl} + d_{lj}$.

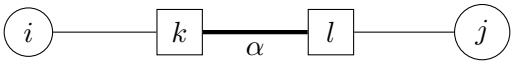


Figura 2: Aplicação do fator de desconto de transporte entre *hubs*.

A seguir é apresentada a formulação tradicional de quatro índices, para o UMApHCP, proposta por Ernst et al. [2009]. Esta formulação matemática usa as seguintes variáveis de decisão: (i) variável contínua s, que representa o valor da função objetivo; (ii) variável binária y_{ijkl}, que assume o valor 1 se e somente se o fluxo entre os nós i e j é realizado através do caminho $i \to k \to l \to j$ e o valor 0, caso contrário; (iii) variável binária z_k, que assume o valor 1 se e somente se o nó k é definido como *hub* e o valor 0, caso contrário. O modelo matemático é, então, dado por:

$$\min \; s \tag{1}$$

$$\text{sujeito a} \sum_{k \in N} z_k = p \tag{2}$$

$$\sum_{k \in N} \sum_{l \in N} y_{ijkl} = 1, \quad i, j \in N \tag{3}$$

$$\sum_{k \in N} y_{ijkl} \leq z_l, \quad i, j, l \in N \tag{4}$$

$$\sum_{l \in N} y_{ijkl} \leq z_k, \quad i, j, k \in N \tag{5}$$

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

$$s \geq \sum_{k \in N} \sum_{l \in N} (d_{ik} + \alpha d_{kl} + d_{lj}) y_{ijkl}, \quad i, j \in N \tag{6}$$

$$y_{ijkl} \in \{0,1\}, z_k \in \{0,1\}, \quad s \in \mathbb{R}_+, \quad i, j, k, l \in N. \tag{7}$$

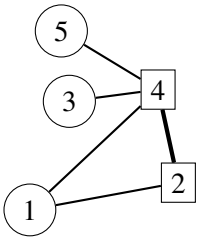
Neste modelo, a função objetivo (1), a ser minimizada, representa o limite superior do custo de atendimento, considerando todos os pares de demandas. A restrição (2) garante que o número de *hubs* selecionados seja exatamente igual a p. O conjunto de restrições (3) assegura que a demanda entre os pares (i, j) seja atendida apenas por um par de *hubs* (k, l) ou por um único *hub*, quando $k = l$. Os conjuntos de restrições (4) e (5) garantem que os nós sejam atribuídos apenas a nós *hubs*. O conjunto de restrições (6) determina que s seja o limitante superior de custo, dentre todos os pares de nós de demandas do grafo. O conjunto de restrições (7) determina o domínio de definição das variáveis de decisão.

3. Algoritmo Implementado

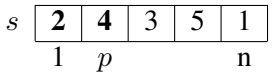
Esta seção aborta a implementação da meta-heurística ALNS para solucionar o UMApHCP, detalhando a representação da solução, a geração da solução inicial, a função de avaliação, o procedimento de busca local incorporado ao ALNS e o próprio algoritmo ALNS implementado.

3.1. Representação da Solução

Neste trabalho, uma solução para o UMApHCP é representada por um vetor de inteiros s de dimensão $n = |N|$, onde cada elemento do vetor representa um nó $i \in N$. Neste vetor, as posições de 1 até p correspondem aos nós determinados como *hubs*, e as posições de $p + 1$ até n correspondem aos nós não *hubs*. A Figura 3 ilustra uma solução para uma instância com $p = 2$ e $n = 5$, onde os nós 2 e 4 são designados como *hubs* e os nós 3, 5 e 1 nós não *hubs*.



(a) Rede.



(b) Representação da solução.

Figura 3: Rede eixo-raio representada pela solução.

3.2. Geração da Solução Inicial

Para construir uma solução inicial s para o UMApHCP, foi utilizada uma heurística construtiva do tipo gulosa. O critério guloso foi a soma das distâncias de cada nó i a todos os outros nós, denotada por sd_i, em que a forma de escolha foi selecionar para *hub* os p nós que estão mais distantes de todos os outros nós. Para ordenar os nós no vetor s, foi utilizado o método de ordenação *Quick Sort*. A implementação da heurística construtiva utilizada é detalhada no Algoritmo 1.

3.3. Função de Avaliação

Seja c_{ij} o menor custo de atendimento da demanda entre os nós i e j, considerando o caminho de custo mínimo na rede associada à solução s. A função de avaliação da solução s é dada por:

$$f(s) = \max_{\forall i,j \in N} \{c_{ij}\}. \tag{8}$$

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

Algoritmo 1: Solução Inicial

```
   Entrada: n, p, d
   Saída: s
1  para i ← 1 até n faça
2  |   sd_i ← 0;
3  |   para j ← 1 até n faça
4  |   |   sd_i ← sd_i + d_ij;
5  |   fim
6  fim
7  Quick Sort(s, sd);
8  Retorne s;
```

Esta função consiste em encontrar o maior custo, calculado dentre todas as demandas, considerando todos os pares origem-destino.

Para determinar o caminho com o menor custo dentre todas as demandas, a partir da configuração de *hubs* são definidas as alocações dos nós não *hubs* através de uma adaptação do algoritmo de caminho mínimo de Floyd-Warshall [Ahuja et al., 1993]. O método de caminho mínimo de Floyd-Warshall encontra o caminho mínimo entre todos os pares ordenados de um grafo. Uma adaptação no método foi realizada, pois no problema tratado o fluxo direto entre nós não *hub* não é permitido. A alteração no método permitiu também a redução da complexidade do algoritmo de $O(n^3)$ para $O(n^2p)$, reduzindo o tempo de processamento.

O pseudocódigo do algoritmo de caminho mínimo de Floyd-Warshall, adaptado para o problema em questão, é apresentado no Algoritmo 2. Para calcular o custo mínimo, é criada uma matriz c para armazenar os custos de todas as demandas origem-destino. O valor c_{ij} dessa matriz é inicialmente definido como infinito positivo caso não exista conexão direta entre os nós i e j; com o valor da distância, caso exista conexão e um dos nós não é *hub*; ou com o valor da distância multiplicado pelo fator de desconto α, se os dois nós são *hubs*.

O Algoritmo 2 recebe como entrada a solução s, as matrizes c e $pred$, que representam o custo e o caminho inicial entre cada par de nós i e j do grafo, o número de nós n e o número de *hubs* p. Este algoritmo retorna as matrizes c e $pred$ atualizadas, com os custos e os caminhos mínimos calculados para todas as demandas. Na linha 4, a avaliação é feita apenas na parte triangular superior das matrizes c e $pred$, com os valores replicados na parte triangular inferior nas linhas 5 e 6.

Algoritmo 2: Floyd-Warshall Adaptado

```
   Dados: s, n, p, c_{n×n}, pred_{n×n}
   Saída: c, pred
1  para k ← 1 até p faça
2  |   para i ← 1 até n e i ≠ s_k faça
3  |   |   para j ← i até n e j ≠ s_k faça
4  |   |   |   se c_ij > c_{i s_k} + c_{s_k j} então
5  |   |   |   |   c_ij ← c_{i s_k} + c_{s_k j}; c_ji ← c_ij;
6  |   |   |   |   pred_ij ← s_k; pred_ji ← pred_ij;
7  |   |   |   fim
8  |   |   fim
9  |   fim
10 fim
```

3.4. Procedimento de Busca Local

Para encontrar ótimos locais da solução, foi utilizado um método de busca local de descida com a estratégia de *Best Improvement*. Esta estratégia consiste em analisar toda a vizinhança da solução corrente, selecionando a melhor solução vizinha. Um movimento ocorrerá quando, após analisar todos os vizinhos, o melhor vizinho for uma solução melhor que a solução corrente. Caso

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

contrário, a solução corrente é um ótimo local, ou seja, todos os vizinhos são piores ou tem qualidade igual a ela.

O procedimento de busca local implementado utiliza uma estrutura de vizinhança baseada no movimento *Swap Hub*. Esse movimento consiste em trocar um *hub* por um nó que não é *hub*. Por exemplo, na Figura 4, uma solução vizinha de uma solução s é a solução s', onde o nó 3 deixa de ser *hub* e o nó 1 passa a ser o novo *hub*. A vizinhança associada a esse movimento inclui todas as soluções obtidas pela troca de um *hub* em uma posição i do vetor s, onde i varia de 1 até p, por nós localizados nas posições j, onde j varia de $p+1$ até n.

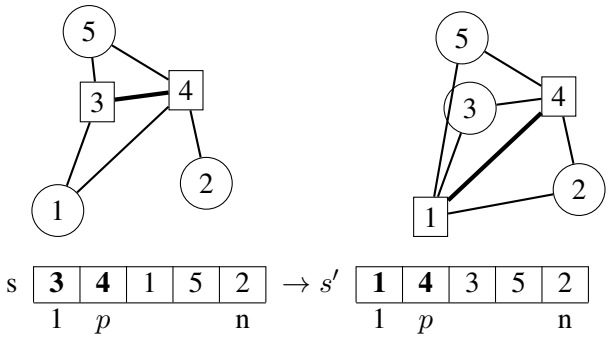


Figura 4: Representação do movimento *Swap Hub*.

O procedimento de busca local é descrito no Algoritmo 3. Esse algoritmo recebe como entrada uma solução inicial s, a função de avaliação, o número de nós n e o número de *hubs* p. Nas linhas 2 a 7, ele procura o melhor vizinho da solução atual. Na linha 8, verifica se esse vizinho representa uma melhoria em relação à solução corrente. Caso positivo, o algoritmo aplica recursivamente o procedimento à nova solução, buscando explorar sua vizinhança. Assim, o algoritmo garante que a solução final seja um ótimo local.

Algoritmo 3: Melhor Vizinho

Entrada: $s, n, p, f(.)$
Saída: s^*
1 $s^* \leftarrow s$;
2 **para** $i \leftarrow 1$ ***até*** p **faça**
3 **para** $j \leftarrow p+1$ ***até*** n **faça**
4 $s' \leftarrow$ Swap Hub(s, i, j);
5 **se** $f(s') < f(s^*)$ **então** $s^* \leftarrow s'$;
6 **fim**
7 **fim**
8 **se** $f(s^*) < f(s)$ **então** $s^* \leftarrow$ Melhor Vizinho$(s^*, n, p, f(.))$;
9 Retorne s^*;

3.5. Meta-heurística *Adaptive Large Neighborhood Search*

A meta-heurística *Adaptive Large Neighborhood Search* (ALNS) [Ropke e Pisinger, 2006] explora o espaço de soluções por meio de operadores que destroem e reparam a solução atual. Esses operadores atuam como diferentes tipos de vizinhanças, que são selecionadas com base na pontuação que recebem e acumulam a cada iteração do ALNS. Essa pontuação define pesos dinâmicos para cada operador, permitindo que o algoritmo se ajuste à instância e ao estado da busca [Gendreau et al., 2010]. Além disso, o ALNS permite a aceitação de uma nova solução que seja pior que a solução corrente, utilizando o critério de aceitação da meta-heurística *Simulated Annealing*

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

[Kirkpatrick et al., 1983]. Nesse caso, a aceitação de uma solução pior é baseada na probabilidade $e^{-\frac{\Delta}{T}}$, onde $\Delta = f(s') - f(s)$ (com $\Delta > 0$) e T é a temperatura corrente, que diminui ao longo da busca.

A implementação da meta-heurística ALNS para resolver o problema abordado é descrita no Algoritmo 4, baseada no trabalho de Coelho et al. [2012]. O algoritmo recebe como entrada os seguintes parâmetros: (i) solução inicial (s_0); (ii) número máximo de iterações sem melhoras (alnsMax); (iii) período de atualização dos pesos (φ); (iv) temperatura inicial (T_0); (v) a taxa de resfriamento da temperatura (κ); (vi) parâmetros ω^1, ω^2, ω^3 e ω^4, que atribuem pontuações conforme a qualidade da nova solução; (vii) taxa de decaimento dos pesos dos operadores (λ), que ajusta a sensibilidade dos pesos conforme o desempenho dos métodos de destruição e reparo (descritos na Subseção 3.5.1); e $(viii)$ o parâmetro que define o nível de destruição da solução ($nivel$). A saída do algoritmo é representada pela variável s^*, que corresponde à melhor solução encontrada.

O algoritmo começa atribuindo a solução inicial às variáveis s e s^*, que correspondem, respectivamente, à solução corrente e à melhor solução encontrada até o momento. A temperatura inicial T_0 é atribuída à variável T, que representará a temperatura corrente. Os pesos dos operadores de destruição e reparo são inicializados com o valor 1, e suas pontuações são definidas como 0. A busca inicia-se com a seleção aleatória de um operador de destruição d e um operador de reparo r dos conjuntos Ω^- e Ω^+, respectivamente, com base nas probabilidades ρ^- e ρ^+. Na linha 8, o operador de destruição selecionado é aplicado à solução corrente, conforme o nível de destruição estabelecido pelo parâmetro $nivel$, resultando na solução destruída s'. Em seguida, o operador de reparo é aplicado a s', gerando a nova solução s''.

Se a nova solução s'' for melhor que a solução corrente s, uma busca local é realizada para encontrar o ótimo local dessa nova solução, e o resultado é atribuído à s. Nesse caso, a variável $pontuacao$ é atualizada com o valor ω^2 (linha 12). Se a solução corrente s superar a melhor solução s^* encontrada até o momento, s^* é atualizada para s, $pontuacao$ recebe ω^1, e $melhorIter$ é ajustada para a iteração corrente $iter$. Caso a nova solução não seja superior a solução corrente, o critério de aceitação é verificado (linha 19). Se o critério permitir a aceitação de uma solução pior, s é atualizada para s'', e $pontuacao$ recebe o valor ω^3. Alternativamente, a busca local pode ser realizada apenas quando há uma melhoria na melhor solução até então, o que constitui uma variação do algoritmo.

Os pesos dos operadores de destruição e reparo utilizados são atualizados com base na pontuação recebida (linha 24). Se a iteração corrente for múltipla do período de atualização dos pesos, os pesos são ajustados de acordo com a pontuação (linhas 27 e 30). Por fim, a temperatura é atualizada, diminuindo seu valor conforme a taxa de resfriamento κ, reduzindo a aceitação de soluções piores ao longo das iterações (linha 33).

O algoritmo repete esse processo até que o critério de parada seja atendido, ou seja, enquanto a temperatura for maior que a temperatura mínima e o número máximo de iterações sem melhorias não for alcançado.

3.5.1. Operadores de Destruição e Reparo

Para diversificar a busca local, foram implementados operadores de destruição e reparo que selecionam aleatoriamente partes da solução. O operador de destruição aleatória (OD1) fecha um ou mais nós *hubs*, ou seja, seleciona aleatoriamente um ou mais nós nas posições 1 a p na solução corrente. Assim, a solução é parcialmente destruída conforme a quantidade de nós *hubs* que deixam de ser *hubs*, definida pelo parâmetro $nivel$. Por outro lado, o operador de reparo aleatório (OR1) abre um ou mais nós não *hubs*, selecionando aleatoriamente nós nas posições $p + 1$ a n na solução

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

Algoritmo 4: Algoritmo ALNS

Entrada: $s_0, n, p, f(.), \Omega^-, \Omega^+, alnsMax, \varphi, T_0, \kappa, \omega^1, \omega^2, \omega^3, \omega^4, \lambda, nivel$
Saída: s^*

$s \leftarrow s_0$; $s^* \leftarrow s$; $T \leftarrow T_0$; $iter \leftarrow 0$; $melhorIter \leftarrow iter$;
$\rho_d^- \leftarrow (1, ..., 1)$; $\rho_r^+ \leftarrow (1, ..., 1)$;
$\psi_d^- \leftarrow (0, ..., 0)$; $\psi_r^+ \leftarrow (0, ..., 0)$;
while $T > 0.01$ ***e*** $iter - melhorIter \leq alnsMax$ **do**
  $iter \leftarrow iter + 1$;
  $pontuacao \leftarrow \omega^4$;
  Selecionar $d \in \Omega^-$ e $r \in \Omega^+$ usando as probabilidades dadas por ρ^- e ρ^+;
  $s' \leftarrow d(s, nivel)$;
  $s'' \leftarrow r(s', nivel)$;
  se $f(s'') < f(s)$ **então**
    $s \leftarrow BuscaLocal(s'')$;
    $pontuacao \leftarrow \omega^2$;
    se $f(s) < f(s^*)$ **então**
      $s^* \leftarrow s$;
      $pontuacao \leftarrow \omega^1$;
      $melhorIter \leftarrow iter$;
    fim
  senão
    $\Delta \leftarrow f(s'') - f(s)$;
    se $\Delta > 0$ **então**
      Gerar um valor aleatório $x \in [0, 1]$;
      se $x < e^{-\frac{\Delta}{T}}$ **então**
        $s \leftarrow s''$;
        $pontuacao \leftarrow \omega^3$;
      fim
    fim
  fim
  $\psi_d^- \leftarrow \psi_d^- + pontuacao$; $\psi_r^+ \leftarrow \psi_r^+ + pontuacao$;
  se $iter \mod \varphi = 0$ **então**
    para *cada operador d com pontuação* > 0 **faça**
      $\rho_d^- \leftarrow \lambda \cdot \rho_d^- + (1 - \lambda) \cdot \psi_d^-$; $\psi_d^- \leftarrow 0$;
    fim
    para *cada operador r com pontuação* > 0 **faça**
      $\rho_r^+ \leftarrow \lambda \cdot \rho_r^+ + (1 - \lambda) \cdot \psi_r^+$; $\psi_r^+ \leftarrow 0$;
    fim
  fim
  $T \leftarrow \kappa \cdot T$
end
Retorne s^*;

destruída. A quantidade de nós aberto é determinada conforme a necessidade para reconstruir a solução.

Para intensificar a busca, foram desenvolvidos operadores de destruição e reparo com diferentes estratégias. Foram criados os seguintes operadores de destruição: (i) operador de destruição OD2, que fecha um ou mais *hubs* que têm a menor distância em relação a todos os outros nós; (ii) operador de destruição OD3, que fecha um ou mais *hubs* que têm a maior distância em relação a todos os outros nós; (iii) operador de destruição OD4, que fecha um ou mais nós *hubs* que, ao fecharem, pioram menos a solução; (iv) operador de destruição OD5, que fecha um ou mais *hubs* que, ao fechar, piora mais a solução; e (v) operador de destruição OD6, que fecha um ou mais *hubs* que estão no caminho de maior custo entre os pares origem e destino.

Além disso, foram criados dois operadores de reparo: (i) operador de reparo OR2, que substitui o *hub* fechado pelo nó não *hub* mais próximo; e (ii) operador de reparo OR3, que substitui o *hub* fechado pelo nó não *hub* mais distante.

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

4. Experimentos Computacionais

O algoritmo ALNS descrito foi implementado em C++ e executado em um computador com processador Intel i5-7500 CPU 3.4GHz, 8GB de RAM e sistema operacional Windows 10. Os testes com o modelo matemático do UMApHCP também foram realizados no mesmo ambiente, utilizando a API *Concert Technologies* da versão acadêmica do *solver* CPLEX 20.1, igualmente em C++.

4.1. Instâncias Utilizadas

Nos experimentos computacionais, foi utilizado o conjunto de instâncias AP (*Australian Post*), introduzido em Ernst e Krishnamoorthy [1996]. Esse grupo de instâncias é baseado em dados reais do serviço postal australiano. Foram utilizadas instâncias de 10 a 40 nós e 2 a 10 *hubs* para comparar os resultados do algoritmo implementado com as soluções obtidas pelo *solver* CPLEX. Para instâncias maiores, com 50 a 200 nós e 2 a 40 *hubs* foram realizadas comparações com os resultados de Brimberg et al. [2017], que utilizaram a meta-heurística *Basic Variable Neighborhood Search* (BVNS), e com os resultados de Ferreira et al. [2023], que empregaram a meta-heurística *Iterated Local Search* (ILS). Devido ao tempo limite definido de 7200 segundos, não foi possível obter soluções para as maiores instâncias no CPLEX.

4.2. Calibração dos Parâmetros

Para calibrar os parâmetros do algoritmo ALNS, foram realizados experimentos utilizando o *software* IRACE [López-Ibáñez et al., 2016]. Para garantir uma calibração imparcial, foram selecionadas instâncias variadas, abrangendo desde o grupo de 10 nós até o grupo de 200 nós, seguindo um padrão em que a primeira instância do grupo de 10 nós, a segunda do grupo de 20 nós, e assim sucessivamente, foram escolhidas até a sétima instância do grupo de 200 nós. Após configurar os parâmetros-alvo no IRACE, os resultados indicaram os seguintes valores: 60 para o número máximo de iterações sem melhora ($alnsMax$); 20.000 para a temperatura inicial (T_0); 0,99975 para a taxa de resfriamento da temperatura (κ); 0,70 para a taxa de decaimento dos pesos (λ); 1 para o nível de destruição da solução ($nivel$) e para o período de atualização dos pesos (φ). Para a definição da pontuação dos operadores, foram seguidas recomendações da literatura, estabelecendo $\omega^1 > \omega^2 > \omega^3 > \omega^4 \leq 0$, ajustados em 10, 5, 2 e 1 ponto, respectivamente. O fator $\alpha = 0,75$, correspondente a um desconto de 25% no custo de conexão entre *hubs*, também foi considerado.

4.3. Resultados computacionais

Para avaliar o desempenho do algoritmo implementado, cada instância foi executada 30 vezes, utilizando os parâmetros definidos pelo IRACE. O tempo de execução foi medido desde a construção inicial até a conclusão do ALNS. O Desvio Relativo Percentual (*Relative Percent Deviation* - RPD) entre o valor médio da função objetivo obtido pelo ALNS e o melhor valor conhecido, seja ele o valor ótimo ou o melhor resultado da literatura, foi calculado conforme descrito a seguir:

$$RPD(\%) = \frac{Valor\ médio - Melhor\ Valor}{Melhor\ Valor} \times 100. \tag{9}$$

Foram avaliadas duas configurações do algoritmo ALNS implementado. Na Configuração 1, a busca local foi aplicada antes de atualizar a melhor solução encontrada, ou seja, na linha 14 do Algoritmo 4. Nesta configuração, a busca local ocorre com menor frequência, pois a nova solução deve ser melhor tanto que a solução corrente quanto a melhor solução encontrada até o momento. Embora isso reduza a probabilidade de encontrar o ótimo global, também diminui o tempo de execução devido ao menor número de buscas locais realizadas.

Na Configuração 2, a busca local foi aplicada antes de atualizar a solução corrente, ou seja, na linha 11, conforme descrito no Algoritmo 4. Nesse cenário, a busca local é mais frequente, pois a nova solução precisa apenas ser melhor que a solução corrente. Além disso, durante a fase

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

Tabela 1: Resultados computacionais para as instâncias de 10 a 40 nós.

Instância		CPLEX		ILS		ALNS Configuração 1		ALNS Configuração 2	
| N |	| H |	Valor	Tempo(s)	RPD(%)	Tempo(s)	RPD(%)	Tempo(s)	RPD(%)	Tempo(s)
10	2	39.922,11	0,48	0,00	0,00	0,00	0,00	0,00	0,00
	3	32.713,93	0,37	0,00	0,01	0,27	0,00	0,00	0,00
	4	31.577,96	0,43	0,00	0,01	0,79	0,00	0,00	0,00
	5	30.371,32	0,43	0,00	0,01	0,01	0,00	0,00	0,00
20	2	45.954,15	7,10	0,00	0,02	0,00	0,00	0,00	0,01
	3	40.909,59	8,46	1,37	0,04	4,24	0,00	0,00	0,03
	4	38.320,25	15,26	0,00	0,04	2,63	0,01	0,00	0,05
	5	37.868,15	11,85	0,00	0,05	0,00	0,01	0,00	0,04
	10	37.868,15	6,25	0,00	0,10	0,00	0,01	0,00	0,07
25	2	51.533,30	37,37	0,00	0,03	0,91	0,00	0,00	0,03
	3	45.552,50	28,57	0,00	0,05	3,58	0,01	0,00	0,05
	4	45.552,50	54,12	0,00	0,07	0,00	0,01	0,00	0,05
	5	45.552,50	32,78	0,00	0,10	0,00	0,01	0,00	0,08
	10	45.552,50	25,64	0,00	0,27	0,00	0,01	0,00	0,12
40	2	61.140,80	1539,50	0,00	0,10	0,62	0,01	0,00	0,07
	3	56.309,88	1708,39	0,00	0,20	0,29	0,02	0,02	0,15
	4	51.279,14	5187,50	0,00	0,28	1,09	0,03	0,00	0,28
	5	49.741,20	4691,89	0,00	0,39	1,16	0,03	0,00	0,35
	10	49.741,20	1791,33	0,00	1,10	0,00	0,03	0,00	0,58
Média			797,25	0,07	0,15	0,82	0,01	0,00	0,10

Tabela 2: Resultados computacionais para as instâncias de 50 a 200 nós.

Instância		BVNS		ILS		ALNS Configuração 1		ALNS Configuração 2	
| N |	| H |	Valor	Tempo(s)	RPD(%)	Tempo(s)	RPD(%)	Tempo(s)	RPD(%)	Tempo(s)
50	2	61.179,03	0,03	0,00	0,19	0,05	0,02	0,00	0,13
	3	56.729,94	0,02	0,33	0,51	1,59	0,04	0,00	0,30
	4	52.905,77	0,04	2,25	0,67	0,83	0,12	0,00	0,51
	5	50.707,87	0,03	0,00	0,91	0,87	0,11	0,00	0,75
	10	50.707,87	0,05	0,00	2,31	0,00	0,21	0,00	1,33
100	2	63.197,10	0,09	0,12	1,89	0,21	0,20	0,02	1,11
	3	57.925,66	0,06	0,00	2,91	2,72	0,55	0,05	2,52
	4	54.704,51	0,18	1,52	6,02	3,55	0,73	0,00	5,19
	5	53.949,33	0,57	0,75	7,25	1,13	1,02	0,09	8,03
	10	51.860,03	0,56	0,00	20,63	0,00	1,47	0,00	15,09
	20	51.860,03	1,19	0,00	62,21	0,08	2,00	0,00	29,00
	30	51.860,03	2,16	0,00	112,25	0,00	0,85	0,00	43,29
	40	51.860,03	3,53	0,00	159,53	0,00	1,34	0,00	61,04
200	2	67.083,28	1,11	0,27	18,55	1,69	0,85	0,22	8,89
	3	62.945,55	2,20	1,26	28,48	2,49	2,04	0,30	20,15
	4	59.420,93	2,92	3,06	42,35	2,68	2,88	0,14	43,47
	5	57.419,32	2,47	2,97	57,68	1,98	6,32	0,13	67,31
	10	55.958,75	2,66	0,08	233,21	0,06	27,88	0,00	135,68
	20	55.958,75	8,45	0,00	570,55	0,00	39,70	0,00	271,89
	30	55.958,75	4,17	0,00	1.091,26	0,00	42,47	0,00	469,85
	40	55.958,75	11,48	0,00	1.799,11	0,02	38,86	0,00	689,92
Média			2,09	0,55	313,28	0,87	7,85	0,04	125,05

de alta temperatura, a solução corrente pode ser substituída por uma solução pior, o que aumenta a chance de encontrar soluções melhores. Embora isso aumente a probabilidade de alcançar o ótimo global, o tempo de execução é maior devido ao maior número de buscas locais realizadas.

A Tabela 1 apresenta os resultados do algoritmo ALNS para as instâncias de 10 a 40 nós, comparando-os com os obtidos pelo *solver* CPLEX e pelos resultados de Ferreira et al. [2023] usando a meta-heurística ILS. O ALNS demonstrou um bom desempenho, na Configuração 1, que apresentou baixo tempo de execução, embora com um RPD médio de $0,82\%$. Já na Configuração 2, apesar do aumento no tempo de execução, houve uma significativa melhoria no RPD médio em comparação tanto com a Configuração 1 quanto com o ILS. Resultados ótimos foram alcançados em quase todas as instâncias, exceto na instância de 40 nós e 3 *hubs*. Em síntese, ambas as configurações do ALNS encontraram soluções de alta qualidade com tempos de execução inferiores aos do solver CPLEX, sendo que a Configuração 2 demonstrou maior eficiência em instâncias maiores.

A Tabela 2 apresenta os resultados do algoritmo para as instâncias de 50 a 200 nós, comparando-os com os obtidos por Brimberg et al. [2017], que usaram a meta-heurística BVNS, e por Ferreira et al. [2023], que utilizaram a meta-heurística ILS. Como os experimentos computacionais de Brimberg et al. [2017] foram realizados em hardware diferente do utilizado neste trabalho, os tempos de execução foram ajustados por um fator de 1,15, conforme o *software PassMark*, disponível em `https://www.cpubenchmark.net`, que permite a comparação das CPUs. Portanto, os tempos de execução apresentados para o BVNS correspondem aos valores originais multiplicados pelo fator de ajuste. O algoritmo ALNS implementado demonstrou um bom desempenho, com tempos de execução baixos na Configuração 1, mesmo para as instâncias maior. Contudo, o RPD médio foi de 0,87%. Na Configuração 2, apesar do aumento no tempo de execução, houve uma melhora significativa no RPD médio, reduzido para 0,04%. Esse resultado supera os 0,87% da Configuração 1 e os 0,55% alcançados pelo ILS. Em resumo, a Configuração 2 do ALNS encontrou soluções de alta qualidade com tempos de execução inferiores aos do ILS, embora ainda superiores aos do BVNS.

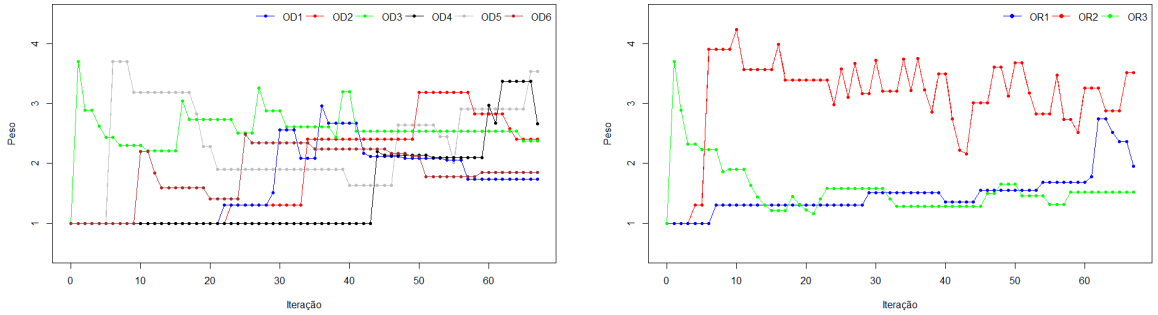
4.4. Pontuação dos Operadores

As Figuras 5(a) e 5(b) mostram as pontuações obtidas pelos operadores de destruição e reparo do algoritmo ALNS durante os experimentos com a instância de 200 nós e 4 *hubs*. Ambos os gráficos evidenciam a adaptabilidade dos operadores no ALNS, com diferentes operadores se destacando em momentos distintos ao longo das iterações.

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

O primeiro gráfico apresenta as pontuações acumuladas dos operadores de destruição ao longo das iterações. As cores das linhas representam os diferentes operadores de destruição (OD1 a OD6), enquanto o eixo vertical mostra a pontuação acumulada (Peso) por cada operador ao longo do tempo. Embora nenhum operador de destruição tenha se sobressaído consistentemente, cada um, em momentos distintos, contribuiu para melhorar o valor da função objetivo da solução corrente ou da melhor solução, conforme refletido nas pontuações.

O segundo gráfico ilustra as pontuações dos operadores de reparo durante as mesmas iterações. Cada linha colorida representa um dos operadores de reparo (OR1 a OR3), com o eixo vertical mostrando o "Peso"acumulado. O operador OR2 se destacou em termos de pontuação. No entanto, durante as iterações, todos os operadores, em determinados momentos, contribuíram para melhorar o valor da função objetivo, seja da solução corrente ou da melhor solução, ou tiveram suas soluções aceitas pelo critério do *Simulated Annealing*.



(a) Operadores de Destruição.

(b) Operadores de Reparo.

Figura 5: Gráfico do valor peso dos operadores de destruição e reparo durante as iterações.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou a implementação do algoritmo ALNS para resolver o problema UMApHCP. Os testes demostraram um desempenho muito bom para as instâncias com 10 a 40 nós, o algoritmo alcançou resultados ótimos com um tempo médio de 0,10s com a Configuração 2. Quando a solução ótima não foi atingida, o algoritmo obteve resultados com valores de função objetivo muito próximos da solução ótima, apresentando um desvio relativo percentual médio reduzido. Além disso, para as instâncias com 50 a 200 nós com a Configuração 2, o algoritmo ALNS superou o ILS tanto em termos de valor da função objetivo quanto em tempo de execução, embora os tempos do BVNS ainda tenham sido mais rápidos.

Como sugestões para trabalhos futuros, destacam-se: (i) a proposição de novos operadores de destruição e reparo; (ii) a exploração de diferentes métodos de busca local; (iii) a implementação de mecanismos para penalizar operadores com alto consumo de tempo; (iv) a realização de testes com outras instâncias disponíveis na literatura.

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 Apoio à Pesquisa do Estado de Minas Gerais (FAPEMIG, processo APQ-00417-21) e ao Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) pelo apoio ao desenvolvimento deste trabalho.

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br

Referências

Ahuja, R. K., Magnanti, T. L., e Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice-Hall, New Jersey, USA.

Brimberg, J., Mišković, S., Todosijević, R., e Uroševic, D. (2022). The uncapacitated r-allocation p-hub center problem. *International Transactions in Operational Research*, 29(2):854–878.

Brimberg, J., Mladenović, N., Todosijević, R., e Urošević, D. (2017). A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem. *Optimization Letters*, 11:313–327.

Campbell, A. M., Lowe, T. J., e Zhang, L. (2007). The p-hub center allocation problem. *European journal of operational research*, 176(2):819–835.

Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72(2):387–405.

Coelho, L. C., Cordeau, J.-F., e Laporte, G. (2012). The inventory-routing problem with transshipment. *Computers & Operations Research*, 39(11):2537–2548.

Ernst, A. T., Hamacher, H., Jiang, H., Krishnamoorthy, M., e Woeginger, G. (2009). Uncapacitated single and multiple allocation p-hub center problems. *Computers & Operations Research*, 36(7): 2230–2241.

Ernst, A. T. e Krishnamoorthy, M. (1996). Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location science*, 4(3):139–154.

Ferreira, A. S., de Sá, E. M., e de Souza, S. R. (2023). Meta-heurística ils aplicada ao problema p-hub centro não capacitado com alocação múltipla.

Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer.

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

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The IRACE package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Mišković, S. e Stanimirović, Z. (2017). A hybrid metaheuristic method for the deterministic and robust uncapacitated multiple allocation p-hub centre problem. *European Journal of Industrial Engineering*, 11(5):631–662.

Ropke, S. e Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Silva, J. F., Martins, F. C., Silva, M. A. L., e da Silva, S. R. d. (2019a). Algoritmo genético aplicado à solução do problema p-hub centro não capacitado de múltiplas alocações. In *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*, p. 856–867. SBC.

Silva, J. F., Silva, M. A. L., de Souza, S. R., e Souza, M. J. F. (2019b). Arquitetura híbrida multiagente aplicada ao problema phub centro não capacitado de múltiplas alocações. In *Anais do LI SimpÃ³sio Brasileiro de Pesquisa Operacional*, p. 108276. Galoá.

DOI: 10.59254/sbpo-2024-193599
https://proceedings.science/p/193599?lang=pt-br