São José dos Campos - SP
LV SBPO
LV Simpósio Brasileiro de Pesquisa Operacional
São José dos Campos, SP - 6 a 9 de novembro de 2023

META-HEURÍSTICA ILS APLICADA 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 *Iterated Local Search* (ILS), para solucionar o problema p-*Hub* centro não capacitado de múltipla alocação. O objetivo deste problema é determinar quais nós serão *Hubs* e a atribuição de cada um dos nós não *Hubs* a um ou mais nós hubs, de modo que o custo máximo de atendimento das demandas, dentre 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 são realizados para comparação do tempo de execução e do valor de função objetivo do algoritmo proposto em relação ao *solver* CPLEX e resultados encontrados na literatura. Os resultados mostram que o algoritmo é capaz de encontrar boas soluções em um tempo computacional adequado.

PALAVRAS CHAVE. Problemas de Localização de *Hubs*, p-*Hub* Centro, meta-heurística ILS.

Tópicos (Logística e Transportes. Meta-heurística.)

ABSTRACT

This article presents an algorithm, based on Iterated Local Search metaheuristic (ILS), to solve the uncapacitated multiple allocation p-hub center problem. The objective of this problem is to determine which nodes will be hubs and the assignment of each of the non-hub nodes to one or more hub nodes, so that the maximum cost of meeting demands, among the costs associated with all origin-destination pairs, be minimized. This problem can be applied in planning a distribution system involving perishable products or time-sensitive products such as urgent shipments. Computational experiments are performed to compare the runtimes and the objective function values of the proposed algorithm in relation to CPLEX solver and results found in the literature. The results show that the algorithm is able to find suitable solutions in adequate computational time.

KEYWORDS. Hub Location Problems, p-Hub Center, metaheuristic ILS.

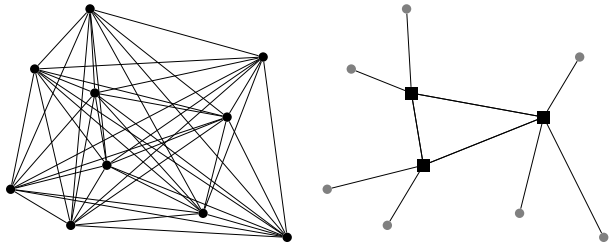
Paper topics (Logistics and Transport. Metaheuristic)

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

1. Introdução

Em sistemas com grande demanda de fluxo a ser roteado entre todos os pares de nós, uma rede com topologia ponto a ponto, com conexão direta entre todos os pares de nós, pode ser ineficiente e cara. Neste caso, uma rede com topologia eixo-raio pode ser mais eficiente e gerar economia, tanto no ponto de vista financeiro, quanto em termos de número de conexões. Um sistema com topologia eixo-raio se caracteriza pela existência de concentradores, também chamados de hubs, que servem para triagem, distribuição e transbordo de mercadorias, pessoas ou serviços e redes de comunicação. As redes eixo-raio foram inicialmente propostas em Goldman [1969].

A Figura 1(a) apresenta uma rede com conexão direta entre todos os pares de nós de demanda origem-destino. Já a Figura 1(b) apresenta uma rede com o mesmo conjunto de nós com uma topologia eixo-raio, em que os quadrados representam os nós hubs e os círculos representam os nós não hubs. Nesta segunda topologia, a conexão entre os nós é realizada por nós hubs e cada nó que não é hub conecta-se apenas a um nó hub.



(a) Ponto a Ponto (b) Eixo-Raio

Figura 1: Conexão entre pares de origem-destino [Silva et al., 2020].

Os problemas de localização de hubs consistem em determinar a localização dos hubs e alocar clientes aos hubs, projetando, desta forma, uma rede do tipo eixo-raio, a fim de otimizar um dado objetivo. Alumur e Kara [2008] classificam os problemas de localização de hubs em quatro grandes grupos: (i) *p-hub median problem (p-HMP)*, que consiste em instalar uma quantidade fixa de hubs que minimize o custo total de transporte; (ii) *p-hub location problem with fixed costs (HLPFC)*, que consiste em minimizar o custo total de instalação de hubs e de transporte; (iii)*p-hub center problem (p-HCP)*, que consiste em localizar um número fixo de hubs que minimize o máximo custo, considerando todos os pares de demandas do sistema; e (iv) *hub covering problems (HCP)*, que consiste em maximizar a cobertura das demandas. As alocações de nós de demanda a nós hubs podem ser (i) simples, em que os nós de demandas são alocados a um único hub, ou (ii) múltiplas, em que os nós de demandas são alocados a um ou mais hubs. O número de hubs pode ser uma quantidade fixa determinada *a priori* ou variável, determinado pelo modelo. Além disso, pode-se considerar também a capacidade nos hubs ou nos arcos que conectam os hubs.

O problema p-hub Centro (*p-hub Center Problem – p-HCP*), proposto por Campbell [1994], consiste em minimizar o máximo custo dentre os custos de atendimento das demandas para cada par origem-destino (critério $\min - \max$), em um sistema com demanda origem-destino entre todos os nós. Este problema aplica-se a um sistema de distribuição envolvendo produtos perecíveis ou produtos sensíveis ao fator tempo, como cargas urgentes. Três variantes do problema foram tratadas em Silva et al. [2020] de forma exata: (i) Problema p-hub Centro não Capacitado de Múltiplas Alocações (*Uncapacitated Multiple Allocation p-hub Center Problem – UMApHCP*) e (ii) duas formulações do Problema p-hub Centro Capacitado de Múltiplas Alocações (*Capacitated Multiple Allocation p-hub Center Problem – CMApHCP*).

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

Este artigo aborda o UMApHCP, que consiste em definir quais nós serão hubs e alocar clientes aos hubs, de forma a minimizar o máximo custo de transporte entre todos os pares origem-destino. Este problema assume que os nós não hubs não possuem restrições de alocação (múltipla alocação) e que os nós hubs não possuem restrições de capacidade (não capacitado). Estudos sobre aspectos computacionais e a complexidade do UMApHCP, apresentados por Kara e Tansel [2000], constatam que este problema é NP-Difícil. Então, diversos algoritmos usando heurísticas e meta-heurísticas já foram propostos. Ernst et al. [2009] desenvolvem um algoritmo heurístico baseado em *simulated annealing* (SA). Brimberg et al. [2017] apresentam uma implementação da meta-heurística *Basic Variable Neighborhood Search* (BVNS). Silva et al. [2019] solucionam o UMApHCP utilizando algoritmo genético e técnicas de aprendizado, que são aplicadas na escolha dos operadores de cruzamento do algoritmo.

O presente trabalho tem, como objetivo, aplicar a meta-heurística *Iterated Local Search* (ILS) [Lourenço et al., 2019] para resolver instâncias do UMApHCP. Esta meta-heurística foi escolhida devido a sua facilidade de implementação, aliada ao pequeno número de parâmetros a serem calibrados, além de seu sucesso na solução de problemas de características semelhantes, como em Oliveira et al. [2023].

O restante deste artigo está estruturado da seguinte forma. Na Seção 2 é apresentado o modelo matemático e a caracterização do problema. A seguir, na Seção 3, são apresentados a representação das soluções, a estrutura de vizinhança proposta e os algoritmos implementados. Os experimentos computacionais e seus resultados são descritos na Seção 4. Por fim, na Seção 5, são apresentadas as considerações finais e as direções futuras de pesquisa.

2. Caracterização do problema

A caracterização do modelo matemático para o UMApHCP é apresentada em Ernst et al. [2009]. Desta forma, o UMApHCP consiste em definir quais nós serão hubs e quais clientes serão alocados a eles, de forma que o máximo custo de transporte entre todos os pares origem-destino seja minimizado. Assume-se que todo nó deve ser alocado a pelo menos um hub, o transporte direto entre nós que não são hubs não é permitido, apenas p hubs devem ser instalados, os nós não hubs não possuem restrições de alocação (múltipla alocação) e os nós hubs não possuem restrições de capacidade (não capacitado).

O UMApHCP pode ser descrito da seguinte forma. Seja $G = (N, A)$ um grafo, em que N é um conjunto com n nós e $A = \{(i, j)|i, j \in N\}$ representa o conjunto de pares de nós. Assume-se que todo nó $i \in N$ é um candidato a se tornar um hub. Para cada par de nós $(i, j) \in A$, c_{ij} representa o custo de transporte na conexão entre o nó i e o nó j. Como o transporte direto entre nós não hub não é permitido, o custo de transporte de $i \in N$ até $j \in N$, passando pelos hubs $k \in N$ e $l \in N$, é dado por:

$$ct_{ij} = \gamma c_{ik} + \alpha c_{kl} + \beta c_{lj} \quad (1)$$

Nesta expressão, c_{ik} representa o custo de coleta entre o nó de origem i e o hub k; c_{kl} representa o custo de transferência entre o hub k e o hub l; e c_{lj} representa o custo de distribuição entre o hub l e o nó de destino j. Os parâmetros γ e β são fatores de custo no fluxo do nó de origem para o nó hub de coleta e do nó hub de distribuição para o nó de destino, respectivamente. Já α é o fator de desconto do fluxo entre hubs, que representa a economia de escala obtida pela concentração de fluxo nos hubs.

O modelo matemático do UMApHCP, proposto em Ernst et al. [2009], é descrito como segue. Seja p o parâmetro que representa o número de hubs a serem definidos no conjunto N; γ, α e β, respectivamente, os custos de coleta, transporte e distribuição; e c_{ij} o custo de transporte na conexão entre o nó i e o nó j. Define-se, então, z_l como uma variável de decisão binária, que

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

assume valor $z_l = 1$ se e somente se o nó l for escolhido para ser um hub, ou $z_l = 0$, caso contrário; e y_{ijkl} uma variável de decisão binária que assume o valor $y_{ijkl} = 1$ se e somente se, para o fluxo entre i e j, o nó i é alocado para o hub k e o nó j para o hub l, ou valor $y_{ijkl} = 0$, caso contrário. O modelo matemático, então, é dado por:

$$\min \quad t \tag{2}$$

$$\text{sujeito a} \quad \sum_{k \in N} z_k = p \tag{3}$$

$$\sum_{k \in N} \sum_{l \in N} y_{ijkl} = 1, \quad i, j \in N \tag{4}$$

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

$$\sum_{l \in N} y_{ijkl} \leq z_k, \quad i, j, k \in N \tag{6}$$

$$t \geq \sum_{k \in N} \sum_{l \in N} (\gamma c_{ik} + \alpha c_{kl} + \beta c_{lj}) y_{ijkl}, \quad i, j \in N \tag{7}$$

$$y_{ijkl}, z_k \in \{0, 1\}, \quad t \in \mathbb{R}_+, \quad i, j, k, l \in N \tag{8}$$

Neste modelo, a função objetivo (2), a ser minimizada, representa o limite superior do custo de atendimento dos pares de demandas. A Expressão (3) garante que o número de hubs selecionados seja exatamente igual a p. A Expressão (4) 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 $k = l$. As Expressões (5) e (6) garantem que os nós sejam atribuídos apenas a nós hubs. A Expressão (7) determina que t seja o limitante superior de custo, dentre todos os pares de nós de demandas do grafo. As Expressão (8) determinam o domínio de definição das variáveis de decisão.

3. Algoritmo proposto para solução

Esta seção apresenta a implementação realizada da meta-heurística *Iterated Local Search* para solucionar o UMApHCP.

3.1. Representação da Solução

Neste trabalho, uma solução para o UMApHCP é representada pelo vetor de inteiros s, de dimensão n, sendo n o número de nós, em que cada elemento deste vetor representa um nó $i \in N$. Neste vetor, as posições de 1 até p representam os nós determinados como hub e as posições de $p + 1$ até n representam os nós determinados como clientes. A Figura 2 representa uma solução para uma instância com p = 2 e n = 5, em que os nós 2 e 4 foram nomeados como hubs e os nós 3, 1 e 5 foram nomeados como nós não hubs.

s	**2**	**4**	3	1	5
	1	p			n

Figura 2: Representação de uma solução para p = 2 e n = 5

3.2. Geração de Solução inicial

Para construir uma solução inicial s para o UMApHCP foi utilizada uma heurística construtiva do tipo gulosa. O critério de escolha foi baseado na soma dos custos entre cada nó i a todos os outros nós, denotado por sc_i, em que foram avaliados dois critérios de escolha: (i) selecionar para hub os p nós que estão menos distantes de todos os outros nós; ou (ii) os que estão mais distantes de todos os outros nós. Para ordenar os nós no vetor s, foi utilizado o algoritmo de ordenação *Quick Sort*. O pseudo código da heurística construtiva utilizada é mostrado no Algoritmo 1.

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

Algoritmo 1: Geração de Solução Inicial

Entrada: s
Saída: s
para $i = 1 \ldots, n$ **faça**
  $sc_i \leftarrow 0$;
  para $j = 1 \ldots, n$ **faça**
    $sc_i \leftarrow sc_i + c_{ij}$;
  fim
fim
$s \leftarrow quickSort(s, sc_i, tipoOrdenacao)$;
Retorne s;

3.3. Função de avaliação

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

$$f(s) = \max_{\forall i,j \in N} \{custo_{i,j}\}. \tag{9}$$

Esta expressão 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, foi realizada uma adaptação do algoritmo de caminho mínimo de Floyd-Warshall. O Algoritmo 2 mostra esta adaptação.

Algoritmo 2: Algoritmo Floyd-Warshall Adaptado

Dados: s
Saída: $custo$
para $k = 1 \ldots, p$ **faça**
  para $i = 1 \ldots, n$ **faça**
    se *i é um hub* **então**
      $custo_{i,s[k]} \leftarrow \alpha \times d_{i,s_k}$;
      $custo_{s[k],i} \leftarrow \alpha \times d_{s_k,i}$;
    senão
      $custo_{i,s[k]} \leftarrow \gamma \times d_{i,s_k}$;
      $custo_{s[k],i} \leftarrow \beta \times d_{s_k,i}$;
    fim
  fim
fim
para $k = 1 \ldots, p$ **faça**
  para $i = 1 \ldots, n$ **faça**
    se $s[k] \neq i$ **então**
      para $j = 1 \ldots, n$ **faça**
        se $s[k] \neq j$ **então**
          se $custo_{i,j} > custo_{i,s[k]} + custo_{s[k],j}$ **então**
            $custo_{i,j} \leftarrow custo_{i,s[k]} + custo_{s[k],j}$;
          fim
        fim
      fim
    fim
  fim
fim
Retorne $custo$;

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

3.4. Estrutura da Vizinhança

O método de resolução proposto utiliza apenas uma estrutura de vizinhança para explorar o espaço de soluções, qual seja, a estrutura *Swap Hub*. A vizinhança obtida pelo movimento *Swap Hub* consiste em trocar um hub por um nó que não é hub. Por exemplo, uma solução vizinha de uma solução s, apresentada na Figura 3, é uma solução s'. Observa-se que o nó 2 deixou de ser hub e o nó 3 é o novo hub. A vizinhança associada a este movimento corresponde às soluções que podem ser obtidas pela troca de hubs em uma posição i do vetor s, em que i varia de 1 até p, por nós na posição j, em que j varia de $p+1$ até n.

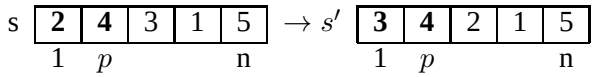


Figura 3: Representação do movimento *Swap Hub*.

3.5. Procedimento de Busca Local

Para encontrar o ótimo local da solução, foram consideradas duas estratégias de busca local: (i) Melhor Vizinho e (ii) Primeira de melhora. A estratégia Melhor Vizinho (*Best Improvement*) 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 contrário, a solução corrente é um ótimo local, ou seja, todos os vizinhos são piores ou tem qualidade igual a ela. A outra busca local avaliada foi a estratégia de primeira de melhora (*First Improvement*), que se assemelha muito à busca local de melhor vizinho. Porém, na busca de primeira melhora, a primeira solução encontrada que melhorar o valor da função objetivo é usada como solução corrente na próxima iteração e a busca na vizinhança da solução corrente é encerrada. Neste trabalho, foi utilizada a estratégia de busca local de Melhor Vizinho. O pseudocódigo desta estratégia de busca local é mostrado no Algoritmo 3. Neste algoritmo, o movimento de troca do nó hub na posição i pelo nó não hub na posição j é representado por $SwapHub(s, i, j)$.

Algoritmo 3: Busca Local

Entrada: s
Saída: s
$s' \leftarrow s, s'' \leftarrow s'$;
para $i = 1 \ldots, p$ **faça**
  para $j = p+1 \ldots, n$ **faça**
    $s' \leftarrow SwapHub(s'', i, j)$;
    se $f(s') < f(s)$ **então** $s \leftarrow s'$;
  fim
fim
se $f(s) < f(s'')$ **então** $s \leftarrow buscaLocal(s)$;
Retorne s;

3.6. Algoritmo *Iterated Local Search* (ILS) implementado

Para explorar vizinhanças mais distantes e evitar que a busca fique presa em ótimos locais, foi utilizada a meta-heurística ILS, em que seu procedimento, definido por Stützle [1998], consiste em explorar o espaço de soluções por meio de perturbações em ótimos locais [Lourenço et al., 2019]. A perturbação precisa ser suficientemente forte para permitir que a busca local explore diferentes soluções e fraca o suficiente para evitar um reinício aleatório. O pseudocódigo utilizado é mostrado no Algoritmo 4.

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

Algoritmo 4: ILS

Entrada: $ilsMax$
Saída: s
$s_0 \leftarrow construcaoInicial()$;
$nivel \leftarrow 2$;
$nivelMax \leftarrow p - 1$;
$iter \leftarrow 0, melhorIter \leftarrow iter$;
$s \leftarrow buscaLocal(s_0)$;
enquanto $iter - melhorIter \leq ilsMax$ **faça**
  $iter \leftarrow iter + 1$;
  $s' \leftarrow perturbacao(s, nivel)$;
  $s'' \leftarrow buscaLocal(s')$;
  se $f(s'') < f(s)$ **então**
    $s \leftarrow s''$;
    $melhorIter \leftarrow iter$;
    $nivel \leftarrow 2$;
  senão
    se $nivel \leq nivelMax$ **então** $nivel \leftarrow nivel + 1$;
  fim
fim
Retorne s;

A estrutura básica do algoritmo ILS consiste na construção de uma solução inicial; a aplicação de um procedimento de busca local sobre essa solução inicial, a fim de intensificar, ou seja, explorar a região atual, retornando uma solução melhorada; em seguida, é aplicado um procedimento de perturbação, visando diversificar, ou seja sair desse ótimo local e explorar novas regiões do espaço de soluções. O parâmetro $nivel$ é o responsável por controlar o nível de perturbação em cada iteração do algoritmo. Este parâmetro é iniciado em $nivel = 2$, ou seja, o procedimento de perturbação inicia-se trocando 2 hubs. Essa definição se deve ao fato do procedimento de busca local já realizar a troca de apenas 1 hub. O procedimento de perturbação é descrito na Subseção 3.7 pelo Algoritmo 5. Após a perturbação, uma nova busca local é então aplicada a essa solução perturbada, verificando-se, então, pelo valor da função de avaliação, se o ótimo local obtido é melhor que a solução corrente. Se a nova solução for melhor que a solução corrente, então move-se para esta nova solução, se reinicia o nível de perturbação e uma nova iteração é realizada. Caso contrário, ocorre um aumento gradativo no nível de perturbação, ou seja, aumenta-se o número de hubs trocados. O parâmetro $nivelMax$ é o responsável por limitar o nível máximo de perturbação, ou seja, o valor máximo que o parâmetro $nivel$ pode chegar. Este valor foi limitado a $p - 1$ para evitar um reinício da solução. O procedimento é interrompido após um certo número máximo de iterações sem melhora no valor da função objetivo. Esse controle é realizado pelo parâmetro $ilsMax$, cujo valor foi definido através do *software* livre de calibração IRACE [López-Ibáñez et al., 2016].

3.7. Procedimento de Perturbação

O método de perturbação utilizado para escapar de ótimos locais embaralha a solução s, fazendo um embaralhamento separado para os hubs, ou seja, para as primeiras p posições da solução, e para os nós não hubs, correspondentes às demais posições. Em seguida, são realizadas trocas de hubs usando o movimento *Swap Hub*. Dessa maneira, busca-se uma nova configuração de hubs para que sejam avaliadas regiões do espaço de soluções ainda não exploradas. A quantidade de hub trocados é definida conforme o nível de perturbação desejado, dado pelo parâmetro $nivel$. O Algoritmo 5 mostra o pseudocódigo do procedimento de perturbação.

4. Resultados dos experimentos computacionais

O algoritmo proposto foi desenvolvido em linguagem C++ e executado em um computador com processador Intel i5-7500 CPU 3.4GHz, com 8GB de memória RAM e sistema operacio-

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

Algoritmo 5: Perturbação

Entrada: $s, nivel$
Saída: s'
$s' \leftarrow embaralhaSolucao(s)$;
para $i = 1, .., nivel$ **faça**
  $s' \leftarrow swapHub(s', i, p + i)$;
fim
Retorne s';

nal Windows 10. Foram realizados no mesmo computador os testes com o modelo matemático do UMApHCP usando as API *Concert Technologies* da versão acadêmica do solver CPLEX 20.1 em linguagem C++.

Para a realização dos experimentos computacionais foi utilizado o grupo de instâncias AP (*Australian Post*), introduzido em Ernst e Krishnamoorthy [1996]. Este grupo de instâncias é baseado em dados reais do serviço postal australiano. Os fatores de custo e desconto no cálculo do transporte são, respectivamente, $\gamma = 1,0$, $\alpha = 0,75$ e $\beta = 1,0$. Este trabalho utilizou as instâncias com até 40 nós nas comparações entre os resultados do algoritmo proposto e a solução via CPLEX, pois, para instâncias acima desse valor, não foi possível obter soluções por meio desse método exato, por exceder o tempo disponível de 7200s para concluir a execução do algoritmo. As instâncias de 50 a 200 nós foram utilizadas nas avaliações entre o algoritmo proposto e os resultados obtidos por Ernst et al. [2009] através do método *Branch and Bound* e por Brimberg et al. [2017] através da meta-heurística Basic Variable Neighborhood Search (BVNS). O *gap* do valor médio de função objetivo da solução obtida pelo ILS foi calculado por:

$$gap(\%) = \frac{Valor\ médio - Valor\ ótimo}{Valor\ ótimo} \times 100. \tag{10}$$

Para encontrar os valores mais adequados para os parâmetros do algoritmo proposto, foi utilizado o *software* IRACE [López-Ibáñez et al., 2016]. Para estes experimentos, foram selecionadas instâncias variadas, buscando uma calibração dos parâmetros não tendenciosa. Assim, foi selecionada a primeira instância do grupo de 10 nós, segunda instância do grupo de 20 nós, e assim até a escolha da sétima instância do grupo de 200 nós. Os parâmetros de alvo do IRACE foi configurado para avaliar entre a construção inicial gulosa, que seleciona os nós que estão mais próximos de todos os outros (heurística construtiva 1) ou mais distantes (heurística construtiva 2), para escolha inicial dos hubs. Entre os métodos de busca local *First Improvement* ou *Best Improvement* e o número de iterações sem melhora entre uma janela de 1 a 30. No final de sua execução, o IRACE retornou como melhor configuração a heurística construtiva 2, com o método de busca local *Best Improvement* e 22 como sendo o número máximo de iterações sem melhora.

Para demonstrar a diferença de cada solução inicial construída por cada heurística construtiva testada, a Figura 4 apresenta a solução inicial para a instância de 20 nós e 10 hubs construída pela heurística construtiva 1. Observa-se na representação que os nós selecionados inicialmente para hubs estão localizados no centro da distribuição. A Figura 5 apresenta a solução construída para a mesma instância pela heurística construtiva 2, melhor avaliada pelo IRACE. A Figura 6 apresenta o *gap* do valor de função objetivo das heurísticas construtivas em relação a melhor solução para cada instância. A heurística construtiva 1 obteve melhores resultados em um número significativo de instâncias, mas observa-se pela linha de tendência que a heurística construtiva 2 se aproxima da melhor solução, e em algumas instância obtém gap igual a 0%, a medida que o número de nós e hubs aumenta. O inverso acontece com o método construtivo 1.

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

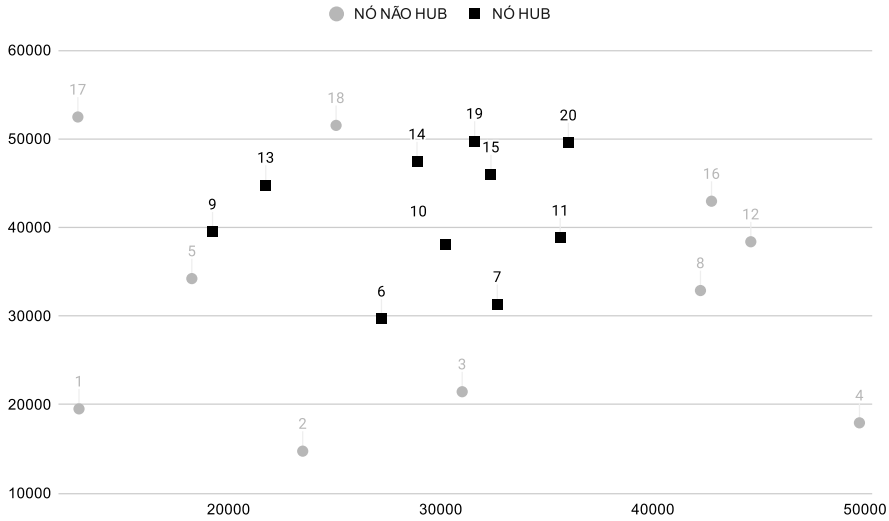


Figura 4: Solução inicial para a instância com 20 nós e 10 hubs usando a heurística Construtiva 1.

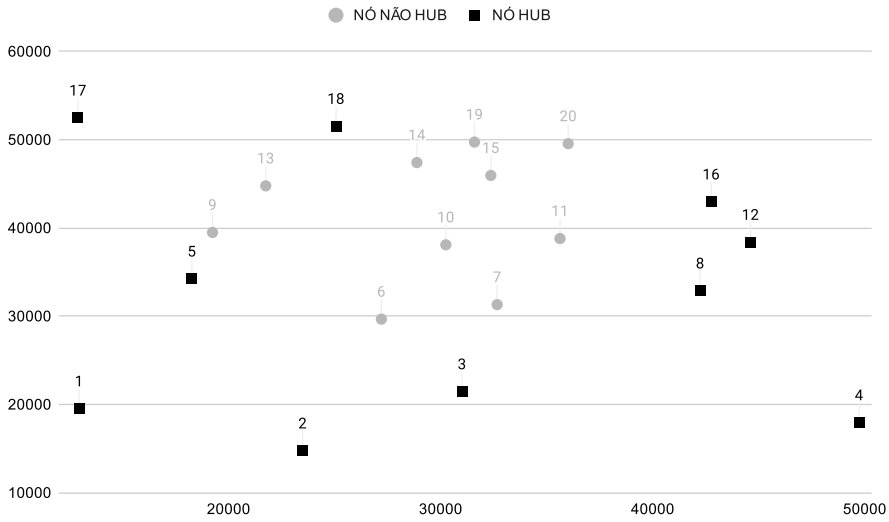


Figura 5: Solução inicial para a instância com 20 nós e 10 hubs usando a heurística Construtiva 2.

Para avaliar o desempenho do algoritmo proposto foram executadas 30 vezes cada instância com número máximo de iterações sem melhora de 1 e 22. O tempo de execução do algoritmo foi calculado desde a construção inicial até o fim da execução do ILS. A Tabela 1 apresenta os resultados do algoritmo para as instâncias do grupo de 10 a 40 nós, esses resultados foram comparados aos resultados exatos obtidos através do CPLEX. Observa-se um bom desempenho do algoritmo, com baixo tempo de execução e apenas não obtendo resultado ótimo na instância com 20 nós e 3 hubs. Nesta instância, foram obtidos gap igual a 4,95% para número máximo de iterações sem melhora igual a 1 e 1,37% para 22.

A Tabela 2 apresenta os resultados do algoritmo para as instâncias do grupo com 50 a 200 nós. Esses resultados foram comparados com os resultados obtidos por Ernst et al. [2009] através do método *Branch and Bound* e por Brimberg et al. [2017] através da meta-heurística Basic

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

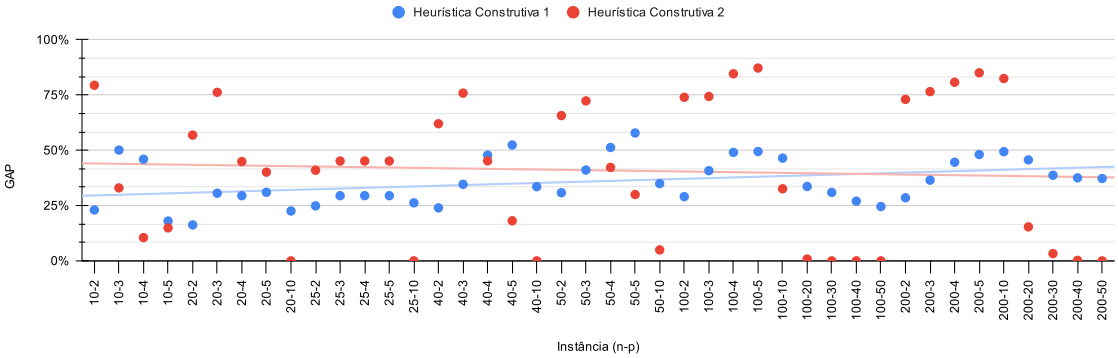


Figura 6: Gráfico do gap da solução construída para cada instância pelos dois métodos construtivos.

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

		Método Exato		Meta-heurística ILS			
Instância		CPLEX		1 ilsMax		22 ilsMax	
$\mid N \mid$	$\mid H \mid$	Valor	Tempo(s)	GAP	Tempo(s)	GAP	Tempo(s)
10	2	39.922,11	0,48	0,00%	0,00	0,00%	0,00
	3	32.713,93	0,37	0,00%	0,00	0,00%	0,01
	4	31.577,96	0,43	0,00%	0,00	0,00%	0,01
	5	30.371,32	0,43	0,00%	0,00	0,00%	0,01
20	2	45.954,15	7,10	0,00%	0,00	0,00%	0,02
	3	40.909,59	8,46	4,95%	0,01	1,37%	0,04
	4	38.320,25	15,26	0,00%	0,01	0,00%	0,04
	5	37.868,15	11,85	0,00%	0,01	0,00%	0,05
	10	37.868,15	6,25	0,00%	0,01	0,00%	0,10
25	2	51.533,30	37,37	0,00%	0,00	0,00%	0,03
	3	45.552,50	28,57	0,00%	0,01	0,00%	0,05
	4	45.552,50	54,12	0,00%	0,01	0,00%	0,07
	5	45.552,50	32,78	0,00%	0,02	0,00%	0,10
	10	45.552,50	25,64	0,00%	0,02	0,00%	0,27
40	2	61.140,80	1539,50	0,00%	0,02	0,00%	0,10
	3	56.309,88	1708,39	0,00%	0,04	0,00%	0,20
	4	51.279,14	5187,50	0,00%	0,04	0,00%	0,28
	5	49.741,20	4691,89	0,00%	0,06	0,00%	0,39
	10	49.741,20	1791,33	0,00%	0,13	0,00%	1,10
Média			797,25	0,26%	0,02	0,07%	0,15

Variable Neighborhood Search (BVNS), os tempos marcados com '-' o autor não informou os valores obtidos. Observa-se muito bons resultados de gap em cada grupo das instâncias a medida que o número de hubs vai aumentando, mas uma melhora pequena mesmo com o aumento do número de iterações.

5. Conclusões e trabalhos futuros

Este artigo apresentou uma implementação do algoritmo ILS aplicada à solução do problema UMApHCP. O objetivo do problema é minimizar o máximo custo entre todas as demandas do sistema. Diante dos testes realizados o algoritmo apresentou um desempenho muito bom, visto que nos experimentos com as instâncias do grupo de 10 a 40 nós, foram obtidos gap médio de 0,26% e 0,07% e Tempo médio de 0,02s e 0,15s, com ilsMax igual a 1 e 22, respectivamente. Em

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

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

Instância		Método Exato		Meta-heurística ILS			
		Branch and Bound		1 ilsMax		22 ilsMax	
$\|N\|$	$\|H\|$	Valor	Tempo(s)	GAP	Tempo(s)	GAP	Tempo(s)
50	2	61.179,03	0,01	0,00%	0,03	0,00%	0,19
	3	56.729,94	0,02	1,72%	0,07	0,33%	0,51
	4	52.905,77	0,03	2,94%	0,08	2,25%	0,67
	5	50.707,87	0,05	0,00%	0,22	0,00%	0,91
	10	50.707,87	0,23	0,00%	0,31	0,00%	2,31
100	2	63.197,10	0,06	0,27%	0,35	0,12%	1,89
	3	57.925,66	0,17	0,00%	0,61	0,00%	2,91
	4	54.704,51	-	5,14%	0,99	1,52%	6,02
	5	53.949,33	2,34	0,76%	1,95	0,75%	7,25
	10	51.860,03	4,06	0,00%	3,35	0,00%	20,63
	20	51.860,03	-	0,00%	8,34	0,00%	62,21
	30	51.860,03	-	0,00%	11,84	0,00%	112,25
	40	51.860,03	-	0,00%	17,25	0,00%	159,53
	50	51.860,03	-	0,00%	22,04	0,00%	210,52
200	2	67.083,28	-	1,40%	2,53	0,27%	18,55
	3	62.945,55	3,67	1,67%	5,98	1,26%	28,48
	4	59.420,93	-	3,60%	6,92	3,06%	42,35
	5	57.419,32	-	2,97%	11,85	2,97%	57,68
	10	55.958,75	-	0,10%	85,26	0,08%	233,21
	20	55.958,75	-	0,00%	107,26	0,00%	570,55
	30	55.958,75	-	0,00%	151,59	0,00%	1.091,26
	40	55.958,75	-	0,00%	238,42	0,00%	1.799,11
	50	55.958,75	-	0,00%	267,71	0,00%	2.776,57
Média				1,14%	32,57	0,55%	313,28

boa parte das instâncias, ele alcançou uma solução ótima com tempos inferiores a 1 min. Quando não alcançou tal solução, o algoritmo encontrou uma solução com valor de função objetivo muito próximo do valor de função objetivo da solução ótima, apresentando gap médios pequenos. Ainda pode-se considerar que o número de hub do sistema interfere diretamente no resultado do algoritmo, afetando no tamanho da vizinhança. Além disso, uma boa construção inicial aumenta a chance do algoritmo conseguir chegar na solução ótima.

Como trabalhos futuros, aponta-se: (i) a proposição de um método de construção inicial adaptativo de acordo com os parâmetros da instância, número de nós e hubs; (ii) a proposição de outras formas de perturbação, para explorar regiões do espaço de busca mais promissoras; (iii) o desenvolvimento de outros algoritmos que aumentem o nível de perturbação somente após algumas tentativas sem sucesso, pois a região de busca pode não ter sido explorada adequadamente; (iv) realizar testes com outras instâncias encontradas na literatura que possam também avaliar a solução proposta.

Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Fundação de Apoio à Pesquisa do Estado de Minas Gerais (FAPEMIG) e ao Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) pelo apoio ao desenvolvimento deste trabalho.

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888

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.

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

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.

Goldman, A. (1969). Optimal locations for centers in a network. *Transportation Science*, 3(4): 352–360.

Kara, B. Y. e Tansel, B. C. (2000). On the single-assignment p-hub center problem. *European Journal of Operational Research*, 125(3):648–655.

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.

Lourenço, H. R., Martin, O. C., e Stützle, T. (2019). *Iterated Local Search: Framework and Applications*, p. 129–168. Springer International Publishing, Cham. URL `https://doi.org/10.1007/978-3-319-91086-4_5`.

Oliveira, F. A., de Sá, E. M., de Souza, S. R., e Souza, M. J. F. (2023). Ils-based algorithms for the profit maximizing uncapacitated hub network design problem with multiple allocation. *Computers & Operations Research*, 157:106252. ISSN 0305-0548. URL `https://www.sciencedirect.com/science/article/pii/S0305054823001168`.

Silva, J. F., Martins, F. C., Silva, M. A. L., e da Silva, S. R. d. (2019). 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 Sá, E. M., de Souza, S. R., e Souza, M. J. F. (2020). Uma abordagem exata do problema p-hub centro. In *Anais do LII Simpósio Brasileiro de Pesquisa Operacional*. SBPO.

Stützle, T. (1998). Local search algorithms for combinatorial problems. *Darmstadt University of Technology PhD Thesis*, 20.

DOI: 10.59254/sbpo-2023-174888
http://live-proceedings.pantheonsite.io/p/174888