Algoritmo Genético Aplicado à Solução do Problema p-*hub* Centro Não Capacitado de Múltiplas Alocações

Jardell Fillipe da Silva[1], **Flávio Vinícius Cruzeiro Martins**[1], **Maria Amélia Lopes Silva**[1], **Sérgio Ricardo de Souza**[1]

[1]Programa de Pós-Graduação em Modelagem Matemática e Computacional
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675 – Nova Gameleira – 30.510-000
Belo Horizonte – MG – Brasil

jardell.jfs@gmail.com, mamelials@gmail.com,
flaviocruzeiro@cefetmg.br, sergio@dppg.cefetmg.br

Resumo. *Este trabalho estuda a solução de uma variante do problema* p-Hub *Centro (pHCP), denominada problema* p-Hub *Centro não capacitado de múltipla alocação (UMApHCP), usando algoritmo genético híbrido. O problema consiste em definir* p hubs *em um grafo completo, de forma que o custo máximo de transporte do grafo seja minimizado. Para a geração da população inicial, usa-se a fase de construção da metaheurística GRASP. Além disso, para cada configuração de* hub *gerada pelo AG, um algoritmo de tempo polinomial é aplicado para determinar a alocação ótima dos nós do grafo. Para avaliar esta implementação, utilizou-se de* 2 *conjuntos de instâncias, disponíveis na literatura, com até* 300 *nós e até* 40 hubs*. Testes computacionais são realizados para comprovar a eficiência da proposta.*

Abstract. *This work addresses a variant of the* p-hub *Center (pHCP) problem, named the Unassigned Multi-Allocation Center* p-Hub *Problem (UMApHCP). The problem is to define* p hubs *in a complete graph so that the maximum transport cost of the graph is minimized. A Hybrid Genetic Algorithm (GA) is used to solve the problem addressed, with the initial population generated by the GRASP metaheuristic construction phase. In addition to each* hub *configuration generated by the GA, a polynomial-time algorithm is applied to determine the optimal allocation of the nodes of the graph. Two sets of instances with up to* 300 *nodes and* 40 hubs *available in the literature were used to evaluate the GA. Computational tests are performed to prove the efficiency of the proposal.*

1. Introdução

Em redes em que o custo de transporte máximo está reduzido a um limitante, deve-se dimensioná-la de forma que o maior custo de transporte seja reduzido. Tomando-se um sistema com demanda origem-destino entre todos os pontos do grafo, dado por uma rede ponto a ponto, em que todos os nós estão interligados entre si, uma forma mais eficiente de se configurar este sistema seria a utilização de uma topologia denominada eixo-raio. Uma rede eixo-raio caracteriza-se por possuir nós concentradores de fluxo (*hubs*). Definir *hubs* e alocar clientes a eles é um problema denominado Problema de Localização de *Hubs* (HLP). Deste problema de localização de *hubs*, extrai-se uma variante, denominada Problema de Localização de p*-Hubs* (pHLP), que consiste em definir p nós que servirão de *hubs* e minimizar a distância máxima percorrida pela rede (critério $\min \sum$). Porém, problemas de localização de *hubs* são contidos por especificidades e variantes. Para uma boa revisão sobre problemas de localização de *hubs*, indica-se a leitura de [Farahani et al. 2013], que apresenta variantes do problema de localização de *hubs* e define-o como uma próspera e nova área da teoria de localização de concentradores.

O Problema p*-hub* Centro (pHCP) é um problema de localização de p *hubs*, que consiste em minimizar o maior custo/distância entre todas as demandas origem-destino de fluxo, na forma de um critério $\min\text{-}\max$. O pHCP é aplicado a problemas como, por exemplo, a definição de instalações de pontos de urgência e emergência, problemas de transporte de alimentos perecíveis, sistemas de transporte em que motoristas estão sujeitos a limites de tempo de viagem, dentre outros casos. O pHCP foi proposto inicialmente por [Campbell 1994], que, além de introduzir o problema, apresentou modelos matemáticos de otimização linear inteira para o mesmo. [Kara and Tansel 2000] apresentaram novas formulações e comprovaram sua complexidade como um problema NP-Difícil. [Campbell et al. 2007] apresentou novas formulações e estudou sub-problemas do pHCP, demonstrando que alguns destes sub-problemas são polinomialmente solúveis.

[Meyer et al. 2009] propõem um método de duas fases para a resolução do pHCP, incluindo conceitos heurísticos. Inicialmente, se constrói um conjunto de combinações possíveis, utilizando-se da solução via *Branch-&-Bound* e um algoritmo de caminho mínimo. Em seguida, aplica-se uma implementação do algoritmo de Otimização em Colônia de Formigas (ACO). [Brimberg et al. 2017b] estudaram, por sua vez, o Problema de Alocação Simples não Capacitado (USApHCP). Eles propuseram, para a resolução deste problema, uma heurística baseada no método de Busca em Vizinhança Variável Geral (GVNS), para o qual foram adotadas duas estruturas de vizinhanças para a busca no ambiente de soluções. Testes computacionais foram realizados e comprovaram a eficiência do método. [Brimberg et al. 2017a] apresentam uma metaheurística de Busca em Vizinhança Variável Básica (BVNS) para resolução do Problema de Alocação Múltipla não Capacitado do p*-hub* Centro (UMApHCP). Testes computacionais foram realizados com as instâncias de referência para o problema. Dois modelos matemáticos de 3 e 4 índices e uma heurística *Multi-Start* foi desenvolvida como base para os testes. Após cada configuração de *hubs* gerada pela metaheurística BVNS, é dada a alocação do UMApHCP por uma adaptação do algoritmo de caminho mínimo de Floyd-Warshall.

O presente trabalho propõe a resolução do UMApHCP usando algoritmo genético, considerando-se a minimização do custo máximo de transporte da rede. O restante deste trabalho é estruturado da seguinte maneira: a Seção 2 apresenta a caracterização do pro-

blema. Na Seção 3 mostra-se a metodologia utilizada. A Seção 4 apresenta o respectivo algoritmo genético. Na Seção 5 são apresentados os resultados computacionais e, por fim, na Seção 6, considerações finais e direções futuras de trabalho são explanadas.

2. Caracterização do Problema

O UMApHCP consiste em definir nós concentradores de fluxo, em um grafo completo, em que todos pares origem-destino possuem demanda, de forma a minimizar o custo máximo de transporte entre estes pares. O UMApHCP constitui-se, portanto, de um grafo completo $G = (N, A)$, em que $N = \{1, 2, \ldots, i, \ldots, n\}$ representa o conjunto de nós e $A = \{(i,j) \mid i,j \in N\}$ representa o conjunto de arcos. Seja, então, Z o conjunto de p nós que serão *hubs*, de forma que $|Z| = p$, e aloque as demandas de origem-destino a Z. Todo os elementos do conjunto N de nós são candidatos a entrar em Z e todos os itens de Z não possuem restrição de capacidade (não capacitado). Todos os nós não-*hubs* conectam-se a pelo menos um *hub* (múltiplas alocações). O custo de transporte c_{ij} entre o nó i e o nó j é dado por:

$$c_{ij} = \gamma d_{ih_i} + \alpha d_{h_i h_j} + \beta d_{h_j j} \tag{1}$$

Observe, na Expressão (1), que o fator d_{ih_i} representa a distância de coleta entre o nó origem i e o *hub* h_i; o fator $d_{h_i h_j}$ representa a distância entre o *hub* h_i e o *hub* h_j; e o fator $d_{h_j j}$ representa a distância de distribuição entre o *hub* h_j e o nó destino j. Note que as constantes γ, α e β são descontos de transporte atribuídos aos *hubs*, de forma que $c_{ij} = c_{ji}$ e $c_{ij}, c_{ii} \geq 0$.

3. Metodologia

3.1. Codificação da Solução

A codificação de um indivíduo é dada por um vetor Z de p posições, que indicam os nós que serão *hubs* no sistema. A Expressão (2) representa um indivíduo, em que os nós 1, 3 e 4 são determinados como *hubs* em um sistema em que $n = 5$ e $p = 3$.

$$Z_1 = \left[\begin{array}{ccc} 1 & 3 & 4 \end{array}\right], \qquad Z_2 = \left[\begin{array}{ccc} 4 & 1 & 3 \end{array}\right] \tag{2}$$

Observe, na Expressão (2), que os vetores Z_1 e Z_2 de *hubs* representam a mesma solução. Note, contudo, que a ordem em que os *hubs* são inseridos não interfere no valor da função de aptidão do indivíduo.

3.2. Decodificação da Solução

Para decodificar uma solução precisa-se, além do vetor Z, de duas matrizes auxiliares que determinam a alocação dos clientes aos *hubs*. A primeira matriz determina a alocação do nó de origem a um *hub*, enquanto a segunda determina a qual *hub* o nó de destino está alocado. Cada elemento da matriz contém o índice do vetor Z de *hubs* em que o cliente está alocado. Como exemplo, considere as Expressões (3) e (4), que representam uma solução para o UMApHCP, com $n = 5$ e $p = 3$, de forma que a Expressão (3) mostra os nós que serão *hubs* e as matrizes M_1 e M_2 da Expressão (4) representam, respectivamente, o *link* do nó de origem e o *link* do nó de destino de uma demanda origem-destino.

$$Z = \left[\begin{array}{ccc} 1 & 3 & 4 \end{array}\right] \tag{3}$$

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 3 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 3 & 2 & 2 \end{bmatrix} \quad M_2 = \begin{bmatrix} 1 & 2 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 & 3 \\ 1 & 2 & 2 & 3 & 2 \\ 1 & 2 & 2 & 3 & 3 \\ 1 & 2 & 2 & 3 & 3 \end{bmatrix} \tag{4}$$

Assim, para atender à demanda $(2, 5)$, deve-se fazer o caminho $(2 \rightarrow 3 \rightarrow 4 \rightarrow 5)$, pois, na matriz M_1, a demanda $(2, 5)$ está ligada ao índice 2 do vetor Z e ligada ao índice 3 do vetor Z na matriz M_2. Portanto, a demanda $(2, 5)$ está ligada ao arco de *hubs* $(3, 4)$.

3.3. Adaptação do Algoritmo de Floyd-Warshall

Para alocar clientes aos hubs e definir as configurações das matrizes auxiliares, a partir da premissa apresentada em [Campbell et al. 2007], de que alocar clientes aos *hubs*, no UMApHCP, é polinomialmente solúvel, optou-se pela utilização de um algoritmo determinístico de caminho mínimo. Para tal, utilizou-se de uma adaptação do algoritmo de Floyd-Warshall, apresentada em [Brimberg et al. 2017a]. A alteração do algoritmo de Floyd-Warshall garante a redução da complexidade deste algoritmo de $O(N^3)$ para $O(N^2p)$, reduzindo, portanto, o tempo de processamento do método. Para toda configuração de Z gerada, o algoritmo de Floyd-Warshall é executado, de modo que se determine as matrizes M_1 e M_2 de alocação e o custo entre todos os pares origem-destino, sendo esse custo armazenado em uma matriz $C_{n \times n}$ criada pelo método. A matriz C é iniciada com cada elemento tendo valor infinito positivo, caso a demanda (i, j) não possua conexão direta, e com a distância entre o nó origem e o nó destino multiplicado pelo fator de desconto, caso haja conexão direta entre a demanda (i, j).

4. Algoritmo Genético Proposto

Para solucionar o UMApHCP, propõe-se a utilização de um Algoritmo Genético tradicional, que será descrito a seguir.

4.1. Construção da Solução Inicial

Para construir uma solução, para o UMApHCP, utilizou-se da fase de construção inicial da metaheurística GRASP [Feo and Resende 1995]. Inicialmente, uma lista LRC de *hubs* candidatos a entrar na solução é criada. Essa lista é determinada pela maior distância de cada nó em relação a todos os outros nós do sistema, de forma que aqueles que possuem a maior distância menor são inseridos no inicio da lista. O tamanho desta é definida pela variável per, que determina o quão guloso o método será. O valor de per varia entre 0 e 1, sendo que, quando seu valor for 0, o método será totalmente guloso, e quando seu valor for 1, será totalmente aleatório. Para construir uma população de boa qualidade e diversa, o valor de per varia durante a construção da população inicial, e seu valor é dado por:

$$per = \frac{\log i}{\log tam\_populacao} \tag{5}$$

A Expressão (5) determina a valor da variável per durante o processo de construção da população inicial. A variável i representa o índice do indivíduo que está sendo inserido na população. O valor de per é determinado em base logarítmica para que não sejam inseridos muitas indivíduos gulosos na população inicial e a diversidade da mesma seja garantida.

Algoritmo 1: Constrói População Inicial

```
   Entrada: tam_pop,
   Saída: pop
1  pop ← ∅
2  ins ← 1
3  per ← 0
4  repita
5  |   ind ← ∅
6  |   cont ← 1
7  |   repita
8  |   |   Construa a Lista Restrita de Candidatos (RCL)
9  |   |   Aleatoriamente, condicionado a per, selecione um elemento s da RCL
10 |   |   ind ← ind ∪ s
11 |   |   cont ← cont + 1
12 |   até cont = p ;
13 |   pop ← pop ∪ ind
14 |   ins ← ins + 1
15 |   per ← Atualiza per (ins, tam_pop)
16 até ins = tam_pop;
17 retorna pop
```

O Algoritmo 1 apresenta a construção da população inicial. O parâmetro de entrada é a dimensão da população (tam_pop). A saída é a população gerada (pop). Observe que o laço da linha 16 é responsável pela criação da população e o laço interno da linha 12 é responsável pela criação de um indivíduo. Observa-se também no algoritmo que o individuo é criado conforme um percentual de aleatoriedade (variável per), apresentado na linha 9, que é atualizada na Linha 15 a partir da Expressão (5).

4.2. Função de aptidão

Para avaliação de aptidão dos indivíduos no problema mono-objetivo utilizou-se de uma função de aptidão, que representa a própria função objetivo do UMApHCP, ou seja:

$$apt_1 = \max\left(C_{ij}\right) \forall i,j 1...n \tag{6}$$

Nesta Expressão, atribui-se à função de aptidão apt_1 o maior valor da matriz de custo C.

4.3. Cruzamento

Para o algoritmo genético proposto, desenvolveu-se quatro cruzamentos e dois métodos que combinam estes quatro cruzamentos. A seleção dos indivíduos que serão enviados para o cruzamento é dada por um torneio binário. O torneio binário consiste em: dados dois indivíduos aleatoriamente, selecione aquele que possuir melhor aptidão. Para cada cruzamento, o torneio binário é repetido por duas vezes, de maneira a selecionar dois pais para o mesmo cruzamento. A seguir, são apresentados os cruzamentos implementados para o algoritmo genético proposto.

Cruzamento de 1 ponto de corte O `cruzamento de um ponto de corte` é uma forma tradicional de realizar cruzamento em um algoritmo genético. Dada a seleção de dois pais, o `cruzamento de um ponto de corte` consiste em gerar um ponto de corte aleatório. O filho 1 recebe, do pai 1, os genes anteriores ao ponto de corte 1, e o filho 2 recebe, do pai 2, os genes anteriores ao ponto de corte 1. A partir do ponto de corte, os genes são alterados, de forma que o filho 1 receba os genes do pai 2 e o filho 2 receba os genes do pai 1. Para garantir a factibilidade do problema, inclui-se conceitos do operador

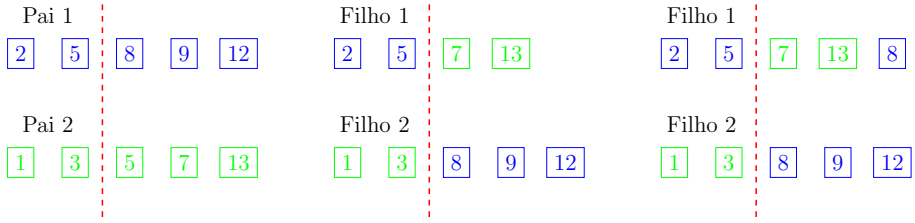

Figura 1. Cruzamento de 1 ponto de corte.

de cruzamento OX. Quando o gene a ser inserido já estiver contido no cromossomo, insere-se os genes referentes à primeira parte de seu pai original. Na Figura 1, o filho 1 recebeu os genes de valor 2 e 5 referentes ao pai 1; já o filho 2 recebeu os genes de valores 1 e 3, referentes ao pai 2. Em seguida, o filho 1 recebeu os genes referente ao pai 2, de valor 7 e 13, pois o gene de valor 5 já estava contido no cromossomo. O filho 2 recebeu os genes de valor 8, 9 e 12, referentes ao pai 1. Na etapa final, para garantir a factibilidade, inseriu-se, no filho 1, o gene de valor 8, referente à segunda parte do cromossomo do pai 1.

Cruzamento de 2 pontos de corte Similar ao `cruzamento de um ponto de corte`, o `cruzamento de dois pontos de cortes` consiste em, dados os pais, dois pontos de corte aleatórios são gerados. O filho recebe os genes de um pai, entre o ponto de corte 1 e o ponto de corte 2. No passo dois, são inseridos os genes do outro pai, fora do intervalo entre os pontos de corte, a partir do primeiro gene do cromossomo. No passo dois, para garantir a factibilidade, caso algum gene já esteja inserido no cromossomo, genes entre os pontos de corte do pai original são inseridos no cromossomo, partindo do primeiro ponto de corte, até que o cromossomo esteja completo. A Figura 2

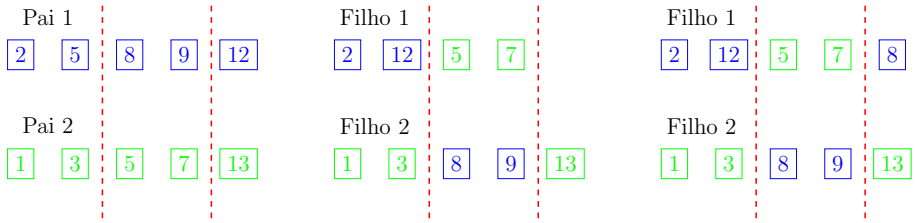

Figura 2. Cruzamento de 2 pontos de corte.

mostra o `cruzamento de dois pontos de corte`. Neste caso, aleatoriamente, os pontos de cortes gerados foram 2 e 4. O filho 1 recebeu os genes de valor 5 e 7 do pai 2 e o filho 2 recebeu os genes de valor 8 e 9 do pai 1. Em seguida, o filho 2 recebeu os genes de valor 1, 3 e 13 referentes ao pai 2. Já o filho 1 receberia os genes 2, 5 e 12 do pai 1, porém o gene de valor 5 já estava inserido no cromossomo, sendo assim inseriu-se o gene de valor 8 pertencente ao pai 1.

Cruzamento Aleatório Para o `cruzamento aleatório`, igualmente aos cruzamentos anteriores, 2 pais são escolhidos. A partir daí, enquanto o cromossomo do filho não estiver completo, aleatoriamente seleciona-se um pai e um gene a ser inserido na solução. Isso é repetido até que o cromossomo do filho esteja completo. A Figura 3 representa o `cruzamento aleatório`. Neste caso, o filho recebe aleatoriamente um gene de um dos pais, até que o cromossomo do filho esteja completo. Note que o filho 1

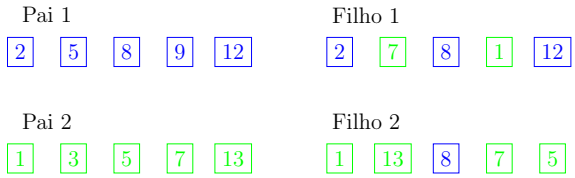

Figura 3. Cruzamento aleatório.

recebe os genes de valor 2, 8 e 12 do pai 1 e os genes de valor 7 e 1 do pai 2, enquanto o filho 2 recebe os genes de valor 1, 13, 7 e 5 do pai 2 e apenas o gene de valor 8 do pai 1.

Cruzamento Fixo O `cruzamento fixo` baseia-se em fixar os genes que se repetem em ambos os pais. Em seguida, é criada uma lista auxiliar, com os valores dos genes que não se repetem no pai 1 ou no pai 2. A partir daí, seleciona-se, aleatoriamente, pelo índice do vetor auxiliar, o gene que entrará no cromossomo do filho 1, ou seja, o gene pertencente ao vetor auxiliar do pai 1 ou do pai 2. Após, obrigatoriamente o filho 2 recebe o gene do vetor auxiliar não selecionado anteriormente. Estas ações são repetidas até o final dos índices do vetor auxiliar. A Figura 4 representa o `cruzamento fixo`. Note

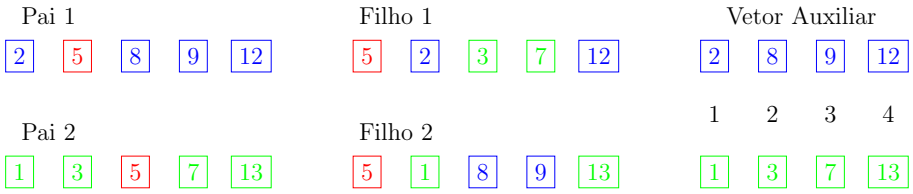

Figura 4. Cruzamento fixo.

que o gene com valor 5 está repetido no pai 1 e no pai 2. Assim, o gene é herdado tanto no filho 1 quanto no filho 2. A partir daí, cria-se um vetor auxiliar para cada pai, com os genes que não se repetem. Os filhos recebem os valores dos genes de forma aleatória para cada índice do vetor auxiliar. Observa-se, então, no sorteio entre os valores de índice 1, que o filho 1 recebeu o gene de valor 8 referente ao pai 1 e o filho 2 o gene de valor 1, referente ao pai 2. Após, o filho 1 recebeu os genes de valor 3 e 7 do pai 2 e o gene de valor 12 do pai 1. Já o filho 2 recebeu os genes 8 e 9 do pai 1 e o gene 13 do pai 2.

4.4. Escolha do Cruzamento

Além dos cruzamentos descritos anteriormente, outros dois métodos, denominados `Todos` e `Aprendizado`, que combinam os cruzamentos anteriores, foram desenvolvidos. No método `Todos`, os 4 cruzamentos são escolhidos aleatoriamente, em forma de roleta. Note-se que todos têm a mesma chance de serem escolhidos, durante todo o processo de evolução. No método `Aprendizado`, os cruzamentos iniciam-se com chances iguais e, à medida que um certo cruzamento é escolhido, caso melhore os filhos, recebe um beneficio percentual pela sua escolha. Dessa maneira, o método tende a escolher cruzamentos com benefícios maiores e que, além disso, tendem a melhorar seus descendentes. Assim, pode-se fazer com que o método opte apenas por um dos cruzamentos ao longo das gerações. Como gerações diferentes podem apresentar comportamentos distintos e com o objetivo de explorar espaços de busca diversos, à medida em que a população evolui, o benefício dos cruzamento é reiniciado, com base nas gerações do algoritmo.

4.5. Mutação

A mutação é dada pela alteração de mut bits do vetor de *hubs* Z. Está alteração consiste em retirar um nó *hub* do vetor de *hubs* e transformar um nó não *hub* em nó *hub*. A quantidade mut de *hubs* a serem alterados é dada por:

$$mut = \left\lceil \frac{p \times 100}{1000} \right\rceil \tag{7}$$

Esta Expressão garante que pelo menos um bit do vetor de *hubs* é alterado. Além disso, à medida em que número de *hubs* aumenta, o número de bits a serem alterados também cresce. Note que, como p é o número de *hubs* do problema, o número de mutações está atrelado à quantidade de *hubs* a se definir.

4.6. Seleção da População Sobrevivente

A geração seguinte do algoritmo genético é obtida de seguinte forma: se, em determinada seleção de pais, não ocorrer cruzamento, os próprios pais entram na geração seguinte; de outro modo, se, em uma dada seleção de pais, ocorrer o cruzamento, os filhos farão parte da próxima geração. Além disso, o algoritmo mantém elitismo simples em todas as suas gerações, ou seja, o melhor individuo é mantido na geração seguinte.

4.7. Algoritmo Genético proposto

O Algoritmo 2 apresenta o Algoritmo Genético Proposto. Note que ele recebe, como entrada, os parâmetros tam_pop, num_ger, tx_cruz e tx_mut, que representam o tamanho da população, o numero de gerações, a taxa de cruzamento e a taxa de mutação, respectivamente. A saída é dada pela variável $melhor\_ind$, que representa o melhor indivíduo encontrado pelo algoritmo. Observe que tanto o cruzamento (linha 10) quanto a mutação (linhas 16 ou 19) ocorrem a partir de uma variável aleatória. Também pode-se observar o elitismo do algoritmo (Linha 24), de forma que a melhor solução da geração anterior é inserida na nova população, de tal modo que o melhor indivíduo da geração anterior é inserido no lugar do pior indivíduo da nova população.

5. Resultados Computacionais

Para realizar os experimentos computacionais, utilizou-se dois conjuntos de instâncias disponíveis na literatura. O primeiro é o conjunto de instâncias CAB (*Civil Aeronautics Board*), baseado no fluxo de passageiros em viagens aéreas entre cidades dos EUA no ano de 1970, introduzida por [O'kelly 1987]. Possui 60 instâncias, com até 25 nós e 2, 3 e 4 *hubs*, para α valendo de $0{,}2$ a $1{,}0$. O segundo conjunto, denominado URAND, foi introduzido por [Meyer et al. 2009] e possui instâncias com 100, 200 e 300 nós, com 2, 3, 4, 5, 10, 15, 20, 30 e 40 *hubs* a serem definidos. O valor de α é fixado em $0{,}75$.

O algoritmo proposto foi desenvolvido em linguagem JAVA, utilizando-se de um IDE Eclipse (4.11.0). Os testes computacionais foram executados 33 vezes, em um computador com sistema operacional Ubuntu 19.04, com processador *Intel Core I5* (8250U) $1{,}6$ GHz, com 8 GB de memória RAM.

Inicialmente, foi realizado um conjunto de testes para avaliar qual é método de seleção do cruzamento seria utilizado, se o método `Todos` ou o método via

Algoritmo 2: Algoritmo Genético Proposto

Entrada: tam_pop, num_ger, tx_cruz, tx_mut
Saída: $melhor\_ind$
$pop \leftarrow$ Constrói População Inicial(tam_pop)
Avalia População(pop)
$melhor\_ind \leftarrow \min(pop)$
$ger \leftarrow 1$
repita
  $cont \leftarrow 1$
  $nova\_pop \leftarrow [\ ]$
  repita
    $pais \leftarrow$ Seleciona dois pais (pop)
    se $ale[0,1] < tx\_cruz$ **então**
      $filhos \leftarrow$ Cruzamento ($pais$)
      $nova\_pop \leftarrow nova\_pop \cup filhos$
    senão
      $nova\_pop \leftarrow nova\_pop \cup pais$
    fim
    se $ale[0,1] < tx\_mut$ **então**
      Mutação($nova\_pop\,[cont]$)
    fim
    se $ale[0,1] < tx\_mut$ **então**
      Mutação($nova\_pop\,[cont+1]$)
    fim
    $cont \leftarrow cont + 2$
  até $cont = tam\_pop$;
  Insere melhor indivíduo da geração anterior($nova\_pop$, $melhor\_ind$)
  $pop \leftarrow nova\_pop$
  Avalia População(pop)
  $melhor\_ind \leftarrow \min(pop)$ $ger \leftarrow ger + 1$
até $ger = num\_ger$;
retorna $melhor\_ind$

`Aprendizado`. Para este fim, foram definidos, de maneira empírica, os seguintes parâmetros: Tamanho da População=100; Taxa de Cruzamento=$0{,}85$; Taxa de Mutação=$0{,}05$; e Número de gerações=500. Os resultados obtidos a partir do uso dos dois métodos foram comparados utilizando um teste de hipóteses paramétricos, com nível de confiança de 95%. O teste utilizado foi a ANOVA (análise de variância), que consiste em testar se duas ou mais populações são iguais ou possuem diferenças significativas. Com base nesta análise, evidenciou-se que os dois métodos possuem diferenças significativas e, a partir da afirmação obtida com a ANOVA, utilizou-se do gráfico da Figura 5 para constatar qual dos métodos é mais eficiente. Concluiu-se, então, que a utilização do método de `aprendizado` foi mais eficiente e este foi tomado, portanto, como método para a resolução do UMApHCP.

5.1. Resultados do Algoritmo Genético

Para a validação do desenvolvimento proposto, foram realizados testes computacionais para verificar o comportamento da população ao longo da evolução do algoritmo. A Figura 6 mostra a evolução da população ao longo das gerações do algoritmo. Note que a média dos valores da população encontra-se sempre próxima à melhor solução em todas as gerações; o pior indivíduo de cada geração, porém, encontra-se afastado do melhor indivíduo da população. Conclui-se, assim, que a população do algoritmo tende a ser boa e diversa. Observa-se também o processo de elitismo contido no algoritmo. Para melhor analisar o desenvolvimento proposto, foram realizadas comparações com os resultados apresentados em [Brimberg et al. 2017a] pelo algoritmo BVNS. O valor de $gap(\%)$ entre a melhor solução e a mediana das soluções, em relação à melhor solução encontrada pelo

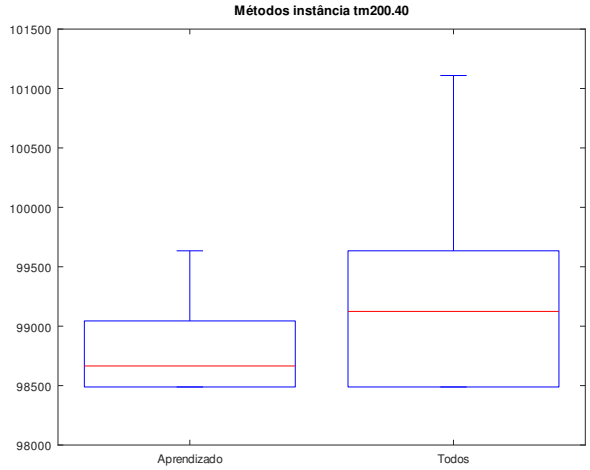

Figura 5. Resultados dos Métodos `aprendizado` e `todos` para a instância tm200.40.

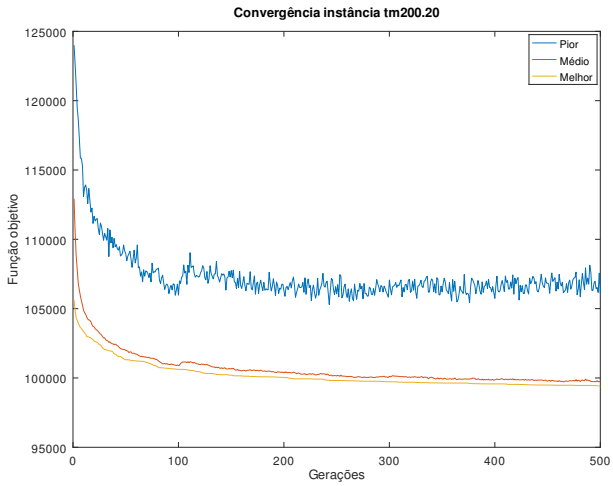

Figura 6. Comportamento da população.

BVNS, é calculado da seguinte forma:

$$gap(\%) = \left(\frac{AGpHCP\_Valor - BVNS\_Valor}{BVNS\_Valor} \right) 100 \tag{8}$$

Na Expressão (8), a variável $AGpHCP\_Valor$ é o valor encontrado no algoritmo genético proposto. Seu valor é dado pela melhor solução encontrada pelo algoritmo proposto, quando o $gap(\%)$ é em relação à melhor solução; é dado pela mediana das soluções, quanto o cálculo do $gap(\%)$ é feito em relação à mediana. A variável $BVNS\_Valor$ é o valor da melhor solução encontrado em [Brimberg et al. 2017a]. No conjunto de instâncias CAB, o algoritmo genético proposto alcançou, em todas as instâncias, o valor ótimo apresentado pelo BVNS. O $gap(\%) = 0.00\%$, tanto para a mediana quanto para a melhor solução. Este resultado valida o desenvolvimento proposto com relação às instâncias com número pequeno de nós. Os resultados para o conjunto de instâncias URAND são apresentados na Tabela 1.

Tabela 1. Resultados conjunto de instâncias URAND.

Instância	Melhor	Mediana	Melhor Lit	$gap(\%)$ Melhor	$gap(\%)$ Mediana
100.02	124922,34	124922,34	124922,34	0,00	0,00
100.03	114692	114692	114692,05	0,00	0,00
100.04	107478,75	107801,25	107478,64	0,00	0,30
100.05	105545,75	106056,25	105545,51	0,00	0,48
100.10	99144,25	99451,25	98558,78	0,59	0,91
100.15	97238,25	97238,25	97238,22	0,00	0,00
100.20	97238,25	97238,25	97238,22	0,00	0,00
100.30	97238,25	97238,25	97238,22	0,00	0,00
100.40	97238,25	97238,25	97238,22	0,00	0,00
200.02	130467	130467	130466,92	0,00	0,00
200.03	117735,45	117735,45	117735,45	0,00	0,00
200.04	111377,52	112509,5	111377,52	0,00	1,02
200.05	106820,5	109369,5	106820,05	0,00	2,39
200.10	103017	103803,75	102182,66	0,82	1,59
200.15	99680,25	100338,75	98558,78	1,14	1,81
200.20	98488,5	99634,25	98488,51	0,00	1,16
200.30	98488,5	98488,5	98488,51	0,00	0,00
200.40	98488,5	98488,5	98488,51	0,00	0,00
300.02	131341	131341	131341,06	0,00	0,00
300.03	120490,01	121409	120490,01	0,00	0,76
300.04	111880,71	114043	111880,71	0,00	1,93
300.05	108973,75	110491,75	108520,46	0,42	1,82
300.10	104161,25	105103,25	102920,26	1,21	2,12
300.15	101823,25	102511,25	100635,29	1,18	1,86
300.20	100151	101109,25	98827,2	1,34	2,31
300.30	98488,5	99299,75	98488,51	0,00	0,82
300.40	98488,5	98488,5	98488,51	0,00	0,00
Média	-	-	-	0,25	0,79

A Tabela 1 apresenta a melhor solução e a mediana encontradas pelo algoritmo proposto, e as compara com o melhor valor de função objetivo encontrado na literatura, determinando-se, assim, os valores de $gap(\%)$. Esta Tabela mostra também a média dos valores de $gap(\%)$ encontrados. Observa-se que tanto o $gap(\%)$ da melhor solução quanto o $gap(\%)$ da mediana são menores que 1%. Esta é uma medida importante da eficiência e da precisão do algoritmo genético proposto em determinar as melhores soluções para esta classe de instâncias.

Tabela 2. $gap(\%)$ **em relação ao número de nós.**

Nº de nós	100	200	300
$gap(\%)$ (Mediana)	0.19	0.83	1.29
$gap(\%)$ (Melhor)	0.00	0.12	0.42

Tabela 3. $gap(\%)$ **em relação ao número de *hubs*.**

Nº de *hubs*	2	3	4	5	10	15	20	30	40
$gap(\%)$ (Mediana)	0.00	0.11	1.08	1.56	1.54	1.22	1.16	0.27	0.00
$gap(\%)$ (Melhor)	0.00	0.0	0.00	0.14	0.87	0.77	0.45	0.00	0.00

As Tabelas 2 e 3 mostram o $gap(\%)$ das instâncias em relação ao número de nós e ao número de *hubs*, respectivamente. Estas Tabelas refletem o comportamento do algoritmo em relação ao número de nós e ao número de *hubs*. Note que, à medida em que o número de nós aumenta, o $gap(\%)$ das melhores soluções também aumenta, conforme mostra a Tabela 2. Observe que, conforme a Tabela 3, não se verifica a mesma tendência ocorrida no caso do número de nós, pois, para soluções com $p = 40$, o algoritmo alcança as melhores soluções e o valor de $gap(\%)$ da mediana também é $0{,}00$.

6. Considerações Finais e Trabalhos Futuros

Este artigo apresentou a implementação de um algoritmo genético para solucionar o problema UMApHCP. O objetivo é minimizar o custo máximo de trnsporte no grafo. Diante dos testes realizados o AG se mostrou eficiente, pois em boa parte das instâncias ele alcançou a melhor solução, quando não alcançou tal solução, o AG ficou muito próximo da mesma. Apresentando GAP's, médio, pequenos. Ainda pode-se considerar que o número de nós do sistema interfere diretamente no resultado do AG, mais não pode-se dizer o mesmo do número de *hubs*. Os resultados apresentados pelo NSGA-II, mostram também sua eficiência, pois além de mostrar que sua evolução ocorre no sentido das duas funções de aptidão. Mostra-se ainda que ele conseguiu uma melhora, em relação ao custo total de transporte do grafo, o melhor indivíduo do problema mono-objetivo.

Aponta-se como trabalhos futuros: (i) desenvolver e testar novos métodos de cruzamento e mutação; (ii) implementar a resolução do problema UMApHCP em sistemas multiagentes, utilizando-se de algoritmos genéticos; (iii) estudar e implementar o problema capacitado (CMApHCP).

Referências

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

Brimberg, J., Mladenović, N., Todosijević, R., and Urošević, D. (2017b). General variable neighborhood search for the uncapacitated single allocation p-hub center problem. *Optimization Letters*, 11:377.

Campbell, A. M., Lowe, T. J., and 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.

Farahani, R. Z., Hekmatfar, M., Arabani, A. B., and Nikbakhsh, E. (2013). Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & Industrial Engineering*, 64:1096–1109.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 9:849–859.

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

Meyer, T., Ernst, A. T., and Krishnamoorthy, M. (2009). A 2-phase algorithm for solving the single allocation p-hub center problem. *Computers and Operations Research*, 36(12):3143 – 3151.

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.