LI Simpósio Brasileiro de Pesquisa Operacional
Limeira–SP, 02 a 06 de setembro de 2019

51º SBPO LIMEIRA 2019

ARQUITETURA HÍBRIDA MULTIAGENTE APLICADA AO PROBLEMA pHUB CENTRO NÃO CAPACITADO DE MÚLTIPLAS ALOCAÇÕES

Jardell Fillipe da Silva
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET- MG
Av. Amazonas, 7675 – Nova Gameleira – Belo Horizonte – MG – Brasil
jardell.jfs@gmail.com

Maria Amélia Lopes Silva
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET- MG
Av. Amazonas, 7675 – Nova Gameleira – Belo Horizonte – MG – Brasil
mamelials@gmail.com

Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais - CEFET- MG
Av. Amazonas, 7675 – Nova Gameleira – Belo Horizonte – MG – Brasil
sergio@dppg.cefetmg.br

Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto - UFOP
Campus Universitário – Morro do Cruzeiro – Ouro Preto – MG – Brasil
marcone@edu.ufop.br

RESUMO

Neste trabalho é estudada uma variante do problema pHub Centro (pHCP), denominada problema p-Hub Centro não capacitado de múltipla alocação (UMApHCP). Considera-se um grafo completo com p hubs, para os quais devem ser alocados clientes, de forma que a maior distância entre todas as demandas origem-destino seja minimizada. Utiliza-se a arquitetura multiagente AMAM para resolução de problemas de otimização, que permite a implementação de métodos híbridos, e na qual cada agente implementa uma metaheurística. O método desenvolvido combina uma construção inicial randômica gulosa com uma busca local de primeiro de melhora, e cada agente implementa a metaheurística *Iterated Local Search*. Para cada configuração de hub gerada, um algoritmo de tempo polinomial é aplicado para determinar a alocação ótima dos nós do grafo. Testes computacionais são realizados a fim de comprovar a eficiência dessa proposta.

PALAVRAS CHAVE. pHub Centro, Sistemas Multiagentes, Metaheurísticas Híbridas.
Tópicos: MH – Metaheurísticas.

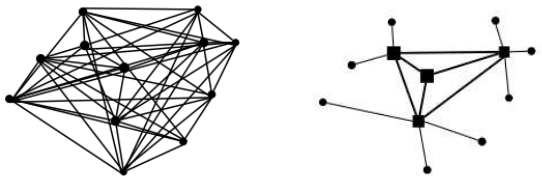
ABSTRACT

In this work, a variant of the pHub Center problem (pHCP), named Uncapacitated Multiple allocation pHub Center Problem (UMApHCP), is addressed. A complete graph with p hubs is considered, for which clients should be allocated so that the greatest distance between all source-destination demands are minimized. The AMAM multi-agent architecture for solving optimization problems is used, which allows the implementation of hybrid methods. In this architecture, each agent implements a metaheuristic. The developed method combines an initial random construction with a first improvement local search and each agent implements the Iterated Local Search metaheuristic. For each generated hub configuration a polynomial time algorithm is applied for determining the optimal allocation of the nodes of the graph. Computational tests are performed in order to prove the efficiency of this proposal.

KEYWORDS. pHub Center, Multi-agent Systems, Hybrid Metaheuristics.
Paper topics: MH – Metaheuristics.

1. Introdução

Em um sistema que dispõe de fluxo entre todos os nós de origem-destino, uma topologia mais eficiente para essa rede se dá através de um sistema eixo-raio (*hub-and-spoke network*). Este tipo de rede é caracterizada pela existência de links intermediários, que fazem com que o número de links diretos entre os nós de origem-destino sejam otimizados. Por exemplo, uma rede de n nós com conexões diretas possui $n(n-1)$ links; já em um uma rede com 1 hub, o número de conexões diminui para apenas $2(n-1)$ links [Farahani et al., 2013].



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

Figura 1: Conexão entre pares de origem-destino [de Sá et al., 2010].

A Figura 1(a), mostrada em de Sá et al. [2010], apresenta um sistema de nós, com pares de demanda origem-destino e links diretos entre todos os nós. Já a Figura 1(b) apresenta o mesmo sistema com uma topologia eixo-raio, que possuem links intermediários entre as demandas de origem-destino. Definir nós intermediários e a alocação dos clientes a eles é um problema denominado Problema de Localização de Hub (*Hub Location Problem - HLP*), no qual comumente hubs são considerados pontos de roteamento, concentração, distribuição e/ou triagem de fluxo, para gerar economicidade do sistema. Entretanto, definir quais nós serão hubs e a quais clientes alocá-los não é uma tarefa trivial. Apesar de ser um conceito originado de sistemas de telecomunicação, hubs são frequentemente utilizados em sistemas logísticos, transportes, aviação civil, remessas expressas, serviços postais, dentre outros. Para que se justifique a demanda do HLP, é necessário o movimento de pessoas, mercadorias e ou informações entre os pares de demanda origem-destino. O HLP é uma nova e prospera área de pesquisa em teoria da localização [Farahani et al., 2013]. Uma boa revisão dos problemas de localização de hubs pode ser encontrada em Alumur e Kara [2008] e Farahani et al. [2013], nas quais são apresentados o estado da arte, modelos, classificação, técnicas de solução e aplicações. O problema de localização de hubs possui diversas variantes. Sua classificação é dada, por exemplo, pelo domínio da solução, pelo critério de avaliação ($\min$-$\sum$, $\min$-$\max$), pelo número de hubs (Exógeno/Endógeno), pela capacidade dos nós, pelo custo de localização dos hubs, pela alocação dos nós (múltipla alocação ou alocação simples), pelo custo de conexão entre nós.

Uma importante variação do problema de localização de hubs é o Problema pHub Centro (*pHub Center Problem - pHCP*). Este problema consiste em minimizar o custo máximo entre os pares origem-destino (Critério $\min$-$\max$) em um sistema com demanda origem-destino entre todos os nós. O pHCP é uma variante que se destaca pela sua aplicação em problemas de localização de centros e veículos de emergência, transportes de produtos perecíveis, transportes em que motoristas estão restritos a um limite de tempo de serviço, por exemplo. O pHCP foi proposto em Campbell [1994], no qual são apresentadas as formulações de otimização linear inteira de 4 problemas de localização de hubs discreto: problema da mediana de pHub, problema de localização de Hub não capacitado, problema de cobertura de pHub e o problema pHub centro de simples alocação (*Single Allocation pHub Center Problem - SApHCP*).

Um novo modelo para o SApHCP é proposto em Kara e Tansel [2000] e comparado com o modelo básico apresentado anteriormente em Campbell [1994]. Kara e Tansel [2000] apresentaram estudos sobre aspectos computacionais e sua complexidade (NP-Difícil) e demonstram que o modelo por eles proposto é melhor em tempo de CPU e em requisitos básicos de armazenamento. Nota-se em Campbell et al. [2007] uma melhoria nas formulações, além de introduzir o problema de alocação do pHCP como um subproblema do pHCP. Resultados de complexidade e formulações para variantes do problema são apresentados. Determina-se que, em alguns casos especiais, o problema de alocação do pHCP é polinomialmente solúvel.

Em Ernst et al. [2009] novas formulações matemáticas são apresentadas e uma nova formulação de otimização inteira mista é efetivada para o problema de alocação simples, além de duas novas formulações de otimização inteira e uma abordagem *branch-and-bound*, propostas para resolver o problema de múltiplas alocações. Testes computacionais comprovam a ineficiência das modelagens com 3 e 4 índices para ambos os problemas, além de provarem sua complexidade computacional. Já Yaman [2008] também propõem duas novas formulações para o problema de localização do pHub centro, de modo a minimizar o comprimento do caminho de origem-destino mais longo, utilizando-se de um algoritmo baseado em Relaxação Lagrangeana e Busca Local para resolução do problema.

Problemas do pHub Centro com características estocásticas são abordados em Sim et al. [2009] e Yang et al. [2011]. No primeiro, são tratados problemas de entrega de encomendas expressas com tempo de viagem variável. Resultados analíticos são discutidos e uma heurística baseada em busca local é construída, afim de solucionar o problema de alocação única. Yang et al. [2011] apresenta um modelo de otimização linear inteira mista para resolução do pHCP, em que o tempo de viagem entre os pontos é considerado aleatório, além do mesmo possuir predições de acontecimento. Um método *branch-and-bound* foi construído para solucionar o problema.

Um algoritmo de resolução em duas fases para o problema de alocação única não capacitado é apresentado por Meyer et al. [2009]. Na primeira fase é construído um conjunto de combinações possíveis de hubs ótimos, usando *branch-and-bound* e um algoritmo de caminho mínimo. A segunda fase consiste em resolver o problema de alocação com uma formulação de tamanho reduzido. Adicionalmente, o algoritmo ACO (Otimização de Colônia de Formigas) é implementado, para encontrar um bom limite superior. Os testes computacionais descritos comprovam a eficiência em problemas para até 400 nós, atingindo seus resultados ótimos com tempo razoável.

Brimberg et al. [2017b] propõem uma heurística para resolução do problema de alocação simples não capacitado (USApHCP) baseado no método de Busca em Vizinhança Variável Geral (GVNS). O algoritmo GVNS possui, como ferramenta de busca local, a estratégia de Descida em Vizinhança Variável (VND). Brimberg et al. [2017b] adotam duas estruturas de vizinhança para a busca no ambiente de soluções. Testes computacionais foram realizados para verificar a eficiência do método e os resultados provam sua eficiência em relação aos métodos anteriores. Deve-se ressaltar que o GVNS alcançou todas as soluções ótimas já conhecidas, e, além disso, foi testado em instâncias de maiores dimensões não testadas anteriormente. Brimberg et al. [2017a] apresentam uma metaheurística de Busca em Vizinhança Variável Básica (BVNS) para resolução do problema de alocação múltipla não capacitado do pHub Centro (UMApHCP). Testes computacionais são 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 toda a configuração de hubs, gerada pelo BVNS, é dada a alocação do pHub Centro através de uma adaptação do algoritmo de caminho mínimo de Floyd-Warshall.

No presente trabalho, é proposta a aplicação de uma arquitetura híbrida multiagente ao

problema não capacitado de múltiplas alocações do pHub Centro (UMApHCP). A arquitetura multiagente proposta realiza a busca pela solução através de metaheurísticas, instanciadas como agentes autônomos, que atuam no ambiente de busca do problema a partir de sua capacidade de percepção e ação, sempre cooperando entre si no decorrer do processo de busca da solução.

O restante deste trabalho foi estruturado da seguinte maneira. A Seção 2 apresenta a caracterização do problema. Após, a Seção 3 mostra a metodologia utilizada. Em seguida, a Seção 4 apresenta o algoritmo proposto. A Seção 5 discute os resultados computacionais e a Seção 6 finaliza o artigo, apresentando conclusões e propostas de trabalhos futuros.

2. Caracterização do Problema

O UMApHCP consiste em definir quais nós serão hubs e quais clientes serão alocados a eles, de forma que o custo máximo de transporte entre todos os pares origem-destino seja minimizado. Os nós não hubs não possuem restrições de alocação (múltipla alocação) e os nós hubs não possuem restrição de capacidade (não capacitado). O UMApHCP é descrito na seguinte forma: dado um grafo $E = (N, A)$, 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, defina o conjunto H de nós hub, de modo que $H \subset N$ e $|H| = p$. Assim, todo nó $i \in N$ é um candidato a entrar no conjunto H; todo nó $i \in N$ é alocado a pelo menos um hub pertencente ao conjunto H; todo nó $i \in N$ não hub pode se conectar a qualquer hub em H; os hubs em H não apresentam restrições de capacidade; o transporte direto entre nós não hub não é permitido; e o custo de transporte d_{ij} é calculado por:

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

de forma que γ, α e β são descontos de transporte atribuídos aos hubs [Ernst et al., 2009]. Note, na Expressão (1), que c_{ih_i} representa o custo de coleta entre o nó origem i e o hub h_i; $c_{h_i h_j}$ representa o custo de transferência entre o hub h_i e o hub h_j; e $c_{h_j j}$ representa o custo de distribuição entre o hub h_j e o nó destino j. O objetivo do problema é minimizar o custo máximo entre os pares origem-destino do grafo.

2.1. Modelo Matemático

Esta seção apresenta uma formulação tradicional do UMApHCP, composta por quatro índices. Foi proposta em Ernst et al. [2009]. As variáveis de decisão são definidas como: (i) variável contínua s, que representa o valor da função objetivo; (ii) variável binária z_k, que está ativa se o nó k é definido como hub; (iii) variável binária de quatro índices y_{ijkl}, que está ativa se a demanda do arco (i,j) passa pelo arco formado pelos hubs (k,l). O modelo matemático é dado por:

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

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

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

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

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

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

$$y_{ijkl}, z_k \in 0,1, \quad i,j,k,l \in N \tag{8}$$

Note que a Expressão (3) garante que o número de hubs selecionados seja exatamente igual a p hubs. A Expressão (4) garante que a demanda (i,j) seja atendida apenas por um arco (k,l). As Expressões (5) e (6) garantem que os nós não hubs sejam atribuídos apenas a nós hubs. A Expressão (7) determina que a variável s seja a de maior custo, dentre todas as demandas do grafo, para que seja minimizada na função objetivo apresentada pela Expressão (2).

3. Metodologia

3.1. Representação da Solução

Representa-se uma solução do UMApHCP por um vetor binário F, de dimensão n, e uma matriz inteira $M_{n\times n}$, sendo $n = |N|$ o número de nós. Cada elemento i do vetor binário representa o nó $i \in N$, que recebe o valor 1 se o nó i é um nó hub, e 0 se o nó i não é um hub. Desta forma, na representação de F mostrada na Expressão (9), tendo $n = 5$ nós e $p = 2$ hubs, os nós 2 e 4 são nós hub e os nós 1, 3 e 5 são nós não hub. A matriz M, por sua vez, representa o caminho de todo o grafo origem-destino. Cada elemento (i,j) define o nó destino. Um exemplo desta representação é mostrado na Expressão (10). Observe que, para cumprir a demanda dada pelo arco $(1,3)$, é necessário apenas passar pelo nó hub 2; já para cumprir a demanda $(3,1)$, deve-se passar pelos nós 4 e 2, ou seja, pelo arco $(4,2)$.

$$Z = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{9}$$

$$M = \begin{bmatrix} 1 & 2 & 2 & 4 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 3 & 4 & 4 \\ 2 & 2 & 3 & 4 & 5 \\ 4 & 2 & 2 & 4 & 5 \end{bmatrix} \tag{10}$$

3.2. Vizinhança

A vizinhança do UMApHCP é dada pela troca de um nó hub com um nó não hub, sendo denominada swap_hub. Como exemplo, a configuração $Z = [10010]$ é uma solução vizinha de $Z = [01010]$, de modo que o nó 2 deixou de ser um nó hub e o nó 1 passa a ser o novo nó hub. A exploração do espaço de soluções é feita pelo movimento n_swap_hub, que consiste em realizar n trocas swap_hub em uma solução corrente. Após a geração de um vizinho, é necessário alocar clientes aos hubs, de forma que o sistema seja totalmente configurado. Sendo assim, é necessária a utilização de um método que determine o caminho entre todos os pares de demanda origem-destino e, assim, definir a configuração da matriz M. O método para obtenção do caminho mínimo, utilizado para este fim, é apresentado na subseção seguinte.

3.3. Caminho Mínimo

A alocação de clientes aos hubs é dada como polinomialmente solúvel para o UMApHCP em Campbell et al. [2007]. Desta forma, para definir essa alocação, é preciso encontrar o caminho mínimo entre todos pares de origem-destino do grafo. Para tal, é utilizada uma adaptação do algoritmo de Floyd-Warshall [Ahuja et al., 1993; Ernst e Krishnamoorthy, 1998; Brimberg et al., 2017a], que consiste em encontrar o caminho mínimo para todos os pares ordenados do grafo. A alteração do algoritmo de Floyd-Warshall garante a redução da complexidade deste algoritmo de $O(N^3)$ para $O(N^2p)$, reduzindo o tempo de processamento do método. Para calcular o custo mínimo, uma matriz $C_{n\times n}$ é criada, para armazenar o custo de toda demanda origem-destino. A matriz C é iniciada com valor infinito positivo, caso a demanda (i,j) não possua conexão direta;

Algoritmo 1: Adaptação de Floyd-Warshall

```
   Entrada: C_{n×n}, M_{n×n}, n
   Saída: C_{n×n}, M_{n×n}
1  início
2      para k = 1 até n faça
3          para i = 1 até n faça
4              se k ≠ i e ∃ caminho entre (k, i) então
5                  para j = i até N faça
6                      se C[i,j] > C[i,k] + C[k,j] então
7                          C[i,j] ← C[i,k] + C[k,j]
8                          C[j,i] ← C[i,k] + C[k,j]
9                          M[i,j] ← M[k,j]; M[j,i] ← M[k,j]
10                     fim
11                 fim
12             fim
13         fim
14     fim
15 fim
16 retorna C_{n×n}, M_{n×n}
```

e com o valor da distância multiplicado pelo fator de desconto, caso haja conexão direta entre a demanda (i, j).

O Algoritmo 1 recebe as matrizes $C_{n\times n}$ e $M_{n\times n}$, que correspondem, respectivamente, ao custo e ao caminho inicial existentes entre cada demanda (i,j) do grafo e n, que informa o número de nós. Este algoritmo retorna as matrizes $C_{n\times n}$ e $M_{n\times n}$, com o custo e o caminho mínimo de todas as demandas já calculado. Note que, na linha 4, o terceiro *loop* é executado se e somente se existir caminho já calculado entre os nós (k,i). A linha 5 determina que a avaliação seja feita somente na parte triangular superior da matriz C e replicada na parte triangular inferior, nas linhas 8 e 9.

3.4. Função de Avaliação

A avaliação de uma solução sol é dada por:

$$f(sol) = \max\{C_{ij}\}, \quad i,j \in N \tag{11}$$

valor este que deve ser minimizado. Na Expressão (11), a condição $\max$ consiste em encontrar o maior custo calculado, dentre todas as demandas (i,j) do sistema, de forma que esta seja minimizada (critério $\min\text{-}\max$).

4. Algoritmo Proposto

Para a resolução do UMApHCP, propõe-se a utilização de uma arquitetura de otimização, usando metaheurísticas, de forma que a mesma permita a criação de métodos híbridos e implemente sistemas multiagentes.

A utilização de *frameworks* na solução de problemas de otimização combinatória, em especial os que utilizam conceitos de sistemas multiagentes, permite a resolução de diversos tipos de problemas, facilitando a inserção de novos agentes (métodos de resolução) e permitindo a hibridização desses métodos. Assim, os *Frameworks* se tornam ferramentas interessantes para a resolução de problemas de otimização combinatória. Uma revisão e análise comparativa dos principais *Frameworks* encontrados na literatura é apresentada em Silva et al. [2018], identificando características desejáveis e lacunas existentes, dentre esses *frameworks*, além de destacar estruturas que apresentam sistemas multiagentes para o desenvolvimento de metaheurísticas híbridas para a solução de problemas de otimização. A partir daí, em razão da busca por recursos que potencializem o desempenho do processo de solução, como hibridizacão e estratégias adaptativas, optou-se pela

utilização do *Framework* AMAM (Arquitetura MultiAgente para Metaheurísticas) [Silva, 2007; Silva et al., 2015, 2018, 2019] para solucionar o UMApHCP.

O *framework* AMAM foi proposto inicialmente em Silva [2007], e, em seguida, novas versões foram apresentadas em Fernandes [2009], Silva et al. [2015] e Silva et al. [2019]. O *Framework* AMAM é uma arquitetura multiagente para metaheurísticas, na qual cada agente implementa uma metaheurística e age buscando a solução para o problema instanciado. Neste *framework,* o espaço de busca do problema tratado corresponde ao ambiente do sistema multiagente. Assim, com troca do ambiente, pode-se alterar com facilidade o problema a ser tratado. A facilidade de inclusão de novos agentes na estrutura multiagente do *framework* garante a escalabilidade do mesmo. A capacidade de movimentação de cada agente se dá pelas formas de manipulação da solução que cada problema possui.

A arquitetura apresenta também uma estrutura de cooperação entre os agentes, dada, na versão atual, através do compartilhamento de soluções em um espaço denominado *Pool* de Soluções. O objetivo desta estrutura de cooperação é orientar os agentes no espaço de soluções em direções mais promissoras, buscando melhorar o resultado e reduzir o tempo de solução. As regras de acesso ao *Pool* de Soluções são apresentadas em Silva et al. [2019].

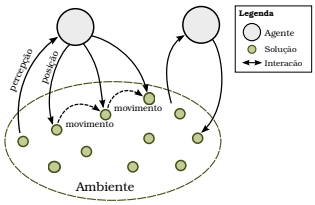


Figura 2: Estrutura do *Framework* AMAM.

A Figura 2 representa a interação do agente com ambiente de busca da solução. Note que o agente, através de sua capacidade de percepção, se posiciona no espaço de busca da solução e, assim, pode se mover pelo mesmo em busca de melhores soluções. Para tal, todo agente encapsula uma heurística/metaheurística que definirá seu comportamento no processo de busca da solução. O framework AMAM disponibiliza tanto a implementação completa de heurísticas e metaheurísticas, prontas para uso, quanto o arcabouço de heurísticas e metaheurísticas que facilitam o desenvolvimento de novos métodos.

A versão mais recente do *Framework* AMAM, lançada em dezembro de 2017, está disponível em https://github.com/mamelials/AMAM-MultiagenteArchitecture-for-Metaheuristics, sob a licença GNU LGPLv3.

4.1. Construção da Solução Inicial

Define-se uma Heurística Construtiva como um método de geração de uma solução inicial [Blum e Roli, 2003]. O processo construtivo da solução inicial se dá pela inserção de elementos em uma solução vazia até que ela se complete. No algoritmo proposto, a implementação da construção da solução inicial se dá por uma construção gulosa aleatória, semelhante à fase de construção inicial do algoritmo GRASP [Feo e Resende, 1995]. Constrói-se uma lista restrita RCL_n de candidatos e, logo em seguida, atribui-se a cada nó i a maior distância entre ele e todos os nós j. Logo após, define-se, aleatoriamente, p hubs, condicionados à distância em relação ao melhor valor encontrado na RCL_n.

O algoritmo 2 mostra como a solução inicial para o UMApHCP é gerada. Uma lista de candidatos é construída na linha 4, sendo que a linha 5 é responsável por garantir a aleatoriedade

Algoritmo 2: Construção Gulosa Aleatória

Entrada: p, N, $dist_{n \times n}$, α
Saída: sol
1 $sol \leftarrow \emptyset$
2 $ins \leftarrow 1$
3 **repita**
4 Construa a Lista Restrita de Candidatos (RCL)
5 Aleatoriamente, condicionado a α, selecione um elemento s da RCL
6 $sol \leftarrow sol \cup s$
7 $ins \leftarrow ins + 1$
8 **até** $ins = p$;
9 **retorna** *sol*

do método, de forma que sua gulosidade está condicionada a um percentual α. Neste trabalho adotou-se o valor de $\alpha = 0{,}75$.

4.2. Busca Local

Busca Local é a forma de refinar uma dada solução pela exploração de seus possíveis vizinhos [Blum e Roli, 2003]. Partindo de uma solução sol corrente, caminha-se pelo espaço de busca, vizinho a vizinho, afim de melhorar a solução, até que esta encontre um ótimo local. Para resolução do UMApHCP foi implementada uma heurística de Primeira Melhora como técnica de busca local. A busca local é dada por uma exploração da vizinhança da solução e, assim que um dado vizinho melhore a solução corrente, esta solução é aceita, e passa a ser a solução corrente do método. Esse procedimento é repetido enquanto houver melhoria na solução corrente. Este procedimento é mostrado no Algoritmo 3.

Algoritmo 3: Busca Local Primeira melhora

Entrada: $sol\_corrente$
Saída: $melhor\_sol$
1 $melhor\_sol \leftarrow sol\_corrente$
2 $melhorou \leftarrow$ ***false***
3 **repita**
4 Selecione $sol' \in N(sol)$;
5 **se** $f(sol') < f(melhor\_sol)$ **então**
6 $melhor\_sol \leftarrow sol'$
7 $melhorou \leftarrow$ ***true***
8 **senão**
9 $melhorou \leftarrow$ ***false***
10 **fim**
11 **até** *melhorou* = ***false***;
12 **retorna** *melhor_solução*

No procedimento do Algoritmo 3, a solução corrente é refinada para encontrar um ótimo local em relação a sua vizinhança. Note que a linha 4 gera um novo vizinho, e sua função é avaliada de forma a verificar se tal vizinho é melhor que a solução corrente.

4.3. Agente ILS

Metaheurísticas são métodos que portam técnicas eficientes para resolução de problemas de otimização. Seu poder consiste em tratar de soluções com pouca ou nenhuma informação do problema [Alba, 2005]. Estabeleceu-se que o agente do *Framework* implementa a metaheurística *Iterated Local Search* (ILS). Este é um método de busca iterativo, que se utiliza de perturbações progressivas na solução para tentar escapar da armadilha de ótimos locais, ou pesquisar diferentes ótimos locais. Iterativamente, este método constrói soluções, aplicando, em soluções correntes conhecidas, perturbações, tendo, como principal objetivo, a diversificação da busca. O ILS é uma metaheurística de fácil entendimento conceitual, porém, de uma grande eficiência em diversos problemas computacionais difíceis [Lourenço et al., 2001]. Basicamente, o ILS é composto por quatro

Algoritmo 4: *Iterated Local Search*

Entrada: $iter_{\max}, nivel_{\max}$
Saída: sol^*
1 $sol^0 \leftarrow$ Gerar Solução Inicial
2 $sol^* \leftarrow$ Busca Local(sol^0)
3 $iter \leftarrow 1$
4 $nivel \leftarrow 1$
5 **repita**
6 $\quad sol' \leftarrow$ Perturba($sol^*, nivel$)
7 $\quad sol'' \leftarrow$ Busca Local(sol')
8 $\quad$ **se** $f(sol'') < f(sol^*)$ **então**
9 $\quad\quad sol^* \leftarrow sol''$
10 $\quad\quad iter \leftarrow 1$
11 $\quad\quad nivel \leftarrow 1$
12 $\quad$ **senão**
13 $\quad\quad iter \leftarrow iter + 1$
14 $\quad\quad$ **se** $nivel < nivel_{\max}$ **então**
15 $\quad\quad\quad nivel \leftarrow nivel + 1$
16 $\quad\quad$ **fim**
17 $\quad$ **fim**
18 **até** $iter = iter_{\max}$;
19 **retorna** sol^*

módulos: (i) geração da solução inicial; (ii) busca local; (iii) pertubação; e (iv) critério de aceitação. O processo do ILS consiste em criar uma solução inicial: aplica-se uma busca local, retornando um ótimo local. A partir daí, enquanto o critério de aceitação não é alcançado, aplica-se perturbações em níveis na solução corrente e, em seguida, novamente, é aplicada a busca local na solução perturbada. Ao fim do processo, retorna-se um ótimo local.

O Algoritmo 4 indica o procedimento da metaheurística ILS. O parâmetro $nivel_{\max}$ de perturbação do ILS é dado pelo parâmetro de entrada p, que determina o número de hubs a ser selecionados, sendo que o valor do mesmo corresponde a p-1 níveis de pertubações, para que não sejam geradas soluções totalmente aleatórias para o método, de forma que ele não se perca no espaço de busca. A perturbação em nível do ILS é executado na linha 6 e o critério de parada, na forma de número de execuções sem melhora, é inserido na linha 18.

5. Resultados Computacionais

Para realização dos experimentos computacionais do UMApHCP foi utilizado o grupo de instância URAND, conjunto de instâncias de maior dimensão introduzidas em Meyer et al. [2009]. Utilizou-se de instâncias com 100 e 200 nós, de forma a definir 2, 3, 4 e 5 hubs.

O *framework* AMAM foi desenvolvido em linguagem JAVA, utilizando-se do IDE Eclipse (4.11.0). Todos os testes computacionais foram realizados em um computador *Intel Core I5* (8250U) 1,6 GHz (4 Cores 8 Threads) com 8 GB de memória RAM e sistema operacional *Ubuntu 18.04*. Os parâmetros utilizados são os mesmos apresentados em Brimberg et al. [2017a] e o número de agentes aplicados à estrutura, para resolução do problema, foram 1, 2, 4 e 8 agentes. Os testes foram executados por 30 vezes, partindo de uma solução inicial diferente.

A Tabela 1 compara os resultados obtidos pelo BVNS apresentado em Brimberg et al. [2017a] e os resultados encontrados com a utilização do *framework* AMAM, de modo a verificar sua eficiência. O *gap*% é calculado pela expressão:

$$gap(\%) = 100\frac{AMAM\_valor - BVNS\_valor}{BVNS\_valor} \tag{12}$$

Pelos resultados apresentados na Tabela 1, observa-se que o *framework* AMAM alcançou os melhores valores obtidos no BVNS em todas as instâncias apresentadas, sendo o *gap*(%)$= 0.00$ em todas

Instância	AMAM Valor	BVNS Valor	GAP (%)
tm100.02	124922.34	124922.34	0.00
tm100.03	114692.05	114692.05	0.00
tm100.04	107478.64	107478.64	0.00
tm100.05	105545.51	105545.51	0.00
tm200.02	130466.92	130466.92	0.00
tm200.03	117735.45	117735.45	0.00
tm200.04	111377.52	111377.52	0.00
tm200.05	106820.05	106820.05	0.00

Tabela 1: Comparação AMAM $\times$ BVNS para o grupo de instância URAND [Brimberg et al., 2017a].

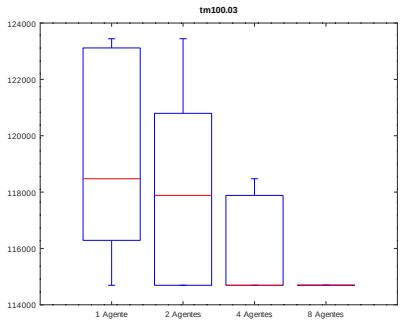


Figura 3: Boxplot dos resultados para a instância tm100.03, com 1, 2, 4 e 8 agentes.

elas. Com isso, conclui-se que o *framework* AMAM é eficiente em relação à busca pela solução do UMApHCP. A Figura 3 mostra o boxplot com os resultados obtidos na instância tm100.03, com a inserção de 1, 2, 4 e 8 agentes.

Na Figura 3 observa-se a melhoria na robustez do *framework* AMAM. À medida em que insere-se agentes, a mediana e a dispersão dos resultados diminui. A Tabela 2 apresenta os resultados obtidos em cada instância com números de agentes variando entre 1, 2, 4 e 8. Esta tabela apresenta o melhor, o pior e a média dos valores encontrados com o uso do *Framework* AMAM, para 1, 2, 4 e 8 agentes.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou a utilização do *framework* AMAM para a resolução do UMApHCP. O *framework* AMAM é uma arquitetura multiagente para a resolução de problemas de otimização. Os resultados obtidos mostram que a utilização do *framework* AMAM é uma boa alternativa para a solução de problemas de otimização combinatória, destacando-se que a metaheurística *Iterated Local Search* implementada via AMAM alcançou os melhores resultados disponíveis na literatura para a classe de instâncias considerada. Além disso, pode-se verificar que a adição de agentes na arquitetura torna o sistema mais robusto e preciso.

Como trabalhos futuros, apontam se os seguintes: (i) inserção de maior número de agentes na arquitetura; (ii) implementar outras metaheurísticas como agente; (iii) apresentar algoritmos que trabalhem com outras estruturas de vizinhanças; e (iv) estudar e implementar o problema capacitado do pHub Centro (CMApHCP).

Agradecimentos

Os autores agradecem ao CEFET-MG (Centro Federal de Educação Tecnológica de Minas Gerais), à UFV (Universidade Federal de Viçosa), à UFOP (Universidade Federal de Ouro Preto) e às agências CAPES, CNPq e FAPEMIG pelo apoio ao desenvolvimento deste trabalho.

(a) 1 agente

Instância	Agentes	Melhor	Pior	Média
tm100.04	1	107478.64	117016	111010.27
tm100.05	1	105545.51	106056.25	105800.87
tm200.02	1	130466.92	130466.92	130466.92
tm200.03	1	117735.45	122081	120754.90
tm200.04	1	111377.52	115251.25	112443.45
tm200.05	1	106820.05	109052.75	107100.70

(b) 2 agentes

Instância	Agentes	Melhor	Pior	Média
tm100.04	2	107478.64	114715.5	109616.45
tm100.05	2	105545.51	106056.25	105800.87
tm200.02	2	130466.92	130466.92	130466.92
tm200.03	2	117735.45	122081	119301.70
tm200.04	2	111377.52	112704	111584.86
tm200.05	2	106820.05	109052.75	107023.43

(c) 4 agentes

Instância	Agentes	Melhor	Pior	Média
tm100.04	4	107478.64	107478.64	107478.64
tm100.05	4	105545.51	105545.51	105545.51
tm200.02	4	130466.92	130466.92	130466.92
tm200.03	4	117735.45	121456.5	118445.34
tm200.04	4	111377.52	111377.52	111377.52
tm200.05	4	106820.05	106820.05	106820.05

(d) 8 agentes

Instância	Agentes	Melhor	Pior	Média
tm100.03	8	114692.05	114692,1	114692,1
tm100.04	8	107478.64	107478.64	107478.64
tm100.05	8	105545.51	105545.51	105545.51
tm200.02	8	130466.92	130466.92	130466.92
tm200.03	8	117735.45	120775	119255.22
tm200.04	8	111377.52	111377.52	111377.52
tm200.05	8	106820.05	106820.05	106820.05

Tabela 2: Comparação entre os resultados alcançados variando o número de agentes.

Referências

Ahuja, R. K., Magnanti, T. L., e Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

Alba, E. (2005). *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience.

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.

Blum, C. e Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308.

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

Brimberg, J., Mladenovic, N., Todosijevic, R., e Urosevic, 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., 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.

de Sá, E. M., de Camargo, R. S., e de Miranda Júnior, G. (2010). Redes eixo-raio aplicada ao transporte público. In *Anais do XLII SBPO*, p. 2415–2426, Bento Gonçalves, RS.

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. (1998). Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. *European Journal of Operational Research*, 104(1): 100–112.

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

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

Fernandes, F. C. (2009). Arquitetura multiagente metaheurísticos cooperativos para resolução de problemas de otimização combinatória. Dissertação de mestrado, Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Belo Horizonte, Brasil.

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.

Lourenço, H. R., Martin, O., e Stützle, T. (2001). A beginner's introduction to Iterated Local Search. In *Proceedings of the 4th Metaheuristics International Conf. (MIC 2001)*, p. 1–11, Porto, Portugal.

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

Silva, M. A. L. (2007). Modelagem de uma arquitetura multiagente para a solução, via metaheurísticas, de problemas de otimização combinatória. Dissertação de mestrado, Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Belo Horizonte, Brazil.

Silva, M. A. L., de Souza, S. R., Souza, M. J. F., e de Oliveira, S. M. (2015). A multi-agent metaheuristic optimization framework with cooperation. In *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, p. 104–109, Natal, Brazil.

Silva, M. A. L., de Souza, S. R., Souza, M. J. F., e Bazzan, A. L. C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Systems with Applications*, 131:148 – 171.

Silva, M. A. L., de Souza, S. R., Souza, M. J. F., e de França Filho, M. F. (2018). Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. *Applied Soft Computing*, 71:433–459.

Sim, T., Lowe, T. J., e Thomas, B. W. (2009). The stochastic p-hub center problem with service-level constraints. *Computers & Operations Research*, 36(12):3166–3177.

Yaman, H. (2008). Star p-hub median problem with modular arc capacities. *Computers & Operations Research*, 35(9):3009–3019.

Yang, K., Liu, Y., e Zhang, X. (2011). Stochastic p-hub center problem with discrete time distributions. In *Proceedings of the 8th International Symposium on Neural Networks (ISNN 2011)*, volume Part II, p. 182–191. Springer.