AGENTE VNS_AMAM APLICADO AO CMApHCP: UMA ABORDAGEM MULTIAGENTE

Jardell Fillipe da Silva[1,2], **Maria Amélia Lopes Silva**[2],
Sérgio Ricardo de Souza[1], **Marcone Jamilson Freitas Souza**[1,3]

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

[2] Universidade Federal de Viçosa (UFV) – campus Florestal
CEP 35.690-000 Florestal (MG), Brasil

[3] Departamento de Computação – Universidade Federal de Ouro Preto (UFOP)
CEP 35.400-000, Ouro Preto (MG), Brasil

jardell.silva@ufv.br, mamelia@ufv.br, sergio@cefetmg.br,
marcone@ufop.edu.br

RESUMO

Este artigo apresenta a instanciação de uma metaheurística híbrida, na forma de agente autônomo, no *Framework* multiagente AMAM, com o objetivo de solucionar o Problema p-*hub* Centro Capacitado de Múltiplas Alocações (CMApHCP). O CMApHCP consiste em definir p *hubs*, em um sistema de demanda origem-destino, com a finalidade de minimizar o custo máximo de todo o sistema. Esta classe de problemas é aplicada em casos práticos que priorizam a rapidez na coleta/entrega. A instância desenvolvida do *Framework* AMAM consiste em um algoritmo híbrido baseado na metaheurística VNS e na busca local VND, tendo a construção das soluções iniciais efetivada pela fase inicial da metaheurística GRASP. Esta instância, na forma de agente, replicada no próprio framework, é usada nos testes computacionais, que comprovam a eficiência da proposta.

PALAVRAS CHAVE. Metaheurísticas Híbridas, Sistemas Multiagentes, *Framework* AMAM.

Tópicos: Metaheurísticas, Otimização Combinatória, Inteligência Computacional

ABSTRACT

This article presents the instantiation of a hybrid metaheuristic, in the form of an autonomous agent, in the AMAM Multi-Agent Framework, to solve the Capacitated Multiple Allocation p-hub Center Problem (CMApHCP) Problem. This problem consists of defining p *hubs* in a source-destination demand system to minimize the maximum cost of the whole system. This class of problems is applied in practical cases that prioritize speed in pickup/delivery. The developed instance of the AMAM Framework consists of a hybrid algorithm based on the VNS metaheuristic and the VND local search, in which the construction of the initial solutions is performed by the initial phase of the GRASP metaheuristic. This instance in the form of an autonomous agent, replicated in the own framework, is used in the computational tests, proving the efficiency of the proposal.

KEYWORDS. Hybrid metaheuristics, Multiagent Systems, Framework AMAM .

Paper topics: Metaheuristics, Combinatorial Optimization, Computational Intelligence.

1. Introdução

Frameworks são abstrações que agrupam códigos que se repetem em um projeto de desenvolvimento de *software*. O preceito de um *framework* é a reutilização de códigos. Assim, a utilização de *frameworks* torna o desenvolvimento de aplicações menos oneroso e complexo, permitindo aumentar a produtividade e a qualidade do processo. *Frameworks* para Otimização usando Metaheurísticas (*Metaheuristic Optimization Framework - MOF*), conforme [Parejo et al., 2012], têm atraído cada vez mais o interesse de pesquisadores da área. A utilização de *frameworks* na solução de problemas de otimização combinatória permite a resolução de diversos tipos de problemas, facilita a inserção de novos métodos de solução, e, em alguns casos, permite até mesmo a hibridização destes métodos. A literatura recente [Parejo et al., 2012; Silva et al., 2018] mostra que estas ferramentas se tornam cada dia mais relevantes no contexto da utilização de métodos metaheurísticos para a solução de problema de otimização combinatória.

O *Framework* AMAM, adotado neste trabalho, foi proposto em Silva [2007] e define uma Arquitetura MultiAgente para Metaheurística. Nesta estrutura, cada agente implementa uma metaheurística/metaheurística híbrida, que age de forma a buscar uma solução para o problema a ser solucionado. Neste *framework*, o espaço de busca do problema tratado corresponde ao ambiente do sistema multiagente. Desta forma, com uma simples troca do ambiente, altera-se com facilidade o problema a ser solucionado. A facilidade de inclusão de novos agentes na estrutura multiagente do *framework* garante a sua escalabilidade. Cada agente é definido no *framework* com capacidades adaptativas, desenvolvidas a partir dos conceitos de Aprendizagem por Reforço. A arquitetura apresenta também uma estrutura de cooperação entre os agentes, através do compartilhamento 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.

O presente artigo apresenta uma aplicação, desenvolvida a partir do *Framework* AMAM, para a solução de um problema de localização de facilidades denominado CMApHCP (*Capacitated Multiple Allocation p-hub Center Problem*). Este tipo de problema, uma das mais importantes variantes do problema de localização de *hubs*, é aplicado em casos práticos que priorizam a rapidez na coleta/entrega. Casos reais com esta característica podem ser encontrados em sistemas de urgência e emergência e sistemas de entrega de alimentos perecíveis, que têm, como característica intrínseca, a limitação do tempo de entrega. Em um sistema constituído de um grande numero de nós, definir centros de distribuição e definir o fluxo de operação é uma tarefa onerosa. Farahani et al. [2013] apresentam problemas de localização de *hubs* como uma nova e prospera área na teoria de localização. Alumur e Kara [2008] e Farahani et al. [2013] apresentam o estado da arte, modelos, classificação, técnicas de solução e aplicações dos problemas de localização de *hubs*.

O principal objetivo deste artigo é apresentar a aplicação instanciada, a partir do *Framework* AMAM, para a solução do CMApHCP, denominada aqui como VNS_AMAM_CMApHCP. Esta aplicação utiliza agentes nomeados como VNS_AMAM. O agente VNS_AMAM é formado por uma metaheurística híbrida, baseada na metaheurística VNS, com busca local implementada pelo método VND e a construção da solução inicial definida com base na fase de construção da metaheurística GRASP. As características específicas do problema foram inseridas no ambiente do sistema multiagente, fazendo uso das facilidades apresentadas pelo *framework*. Da mesma forma, o agente VNS_AMAM foi inserido e replicado. O propósito da aplicação é a solução do problema CMApHCP buscando obter resultados competitivos com a literatura atual. Da mesma forma, o agente VNS_AMAM foi inserido no *framework* e replicado para a realização dos testes computacionais, buscando obter resultados competitivos com a literatura atual.

Este artigo está estruturado da seguinte forma. Na Seção 2, é apresentado o *Framework*

AMAM, sua estrutura e suas principais características. A seguir, na Seção 3, o problema CMApHCP é contextualizado, apresentando sua caracterização e seu modelo matemático. A seção 4 apresenta a instanciação do agente VNS_AMAM no *framework* e as representações computacionais do problema. Os experimentos computacionais e seus resultados são descritos na Seção 5. Por fim, na Seção 6, são apresentadas as considerações finais e as direções futuras.

2. Framework AMAM

O *Framework* AMAM foi proposto inicialmente em Silva [2007]. A versão atual está apresentada em Silva et al. [2019]. Este *Framework* é uma arquitetura multiagente para metaheurísticas, na qual cada agente implementa uma metaheurística e age de forma a buscar uma solução para um problema. No *framework*, o espaço de busca do problema corresponde ao ambiente do sistema multiagente. Assim, com uma simples troca do ambiente, altera-se com facilidade o problema a ser tratado. A capacidade de movimentação de cada agente pelo ambiente se dá pelas formas de manipulação da solução que cada problema disponibiliza. A facilidade de inclusão de novos agentes na estrutura multiagente do *framework* garante a sua escalabilidade, sendo que a adição desses agentes afeta minimamente o restante da estrutura. Estes agentes possuem a capacidade de interagir e perceber o ambiente de forma cooperativa, além de possuírem capacidades adaptativas.

A estrutura de cooperação entres os agentes da arquitetura AMAM é realizada através do compartilhamento de soluções em um espaço denominado *Pool* de Soluções. O objetivo desta estrutura 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]. As seções a seguir detalham os principais elementos e o funcionamento do *framework* AMAM.

2.1. Estrutura de Funcionamento

Os principais elementos que definem o *Framework* AMAM são: (i) Ambiente: corresponde ao espaço de busca do problema tratado. Assim, é formado por todas as soluções do problema, e engloba também todas as informações necessárias para a solução; (ii) Agente: correspondente ao agente propriamente dito, incorpora o método de solução e define suas habilidades e sua forma de interação com o ambiente; (iii) Cooperação: responsável pela interação e cooperação entre os agentes. As seções de 2.2 a 2.4 apresentam com mais detalhes estes elementos que compõem o *framework*.

2.2. Ambiente

O Ambiente do *Framework* AMAM é o próprio espaço de busca do problema. O Ambiente é composto principalmente pelo seguintes elementos: (i) Problema: estrutura de dados que descreve o problema, envolvendo, assim, a leitura dos dados da instância e as informações específicas necessárias para sua solução; (ii) Solução: representação computacional da solução, definindo funções que operam sobre ela, como avaliação da função objetivo e inserção de elementos na solução; e (iii) Movimento: definição dos movimentos do agente no ambiente, compondo a lista de possíveis movimentos. É importante ressaltar que os movimentos são definidos no ambiente por serem elementos específicos para cada problema tratado; entretanto, são manipulados pelo agente sem que este tenha a necessidade de conhecer informações específicas do problema.

2.3. Agente

O agente no *Framework* AMAM incorpora uma metaheurística/metaheurística híbrida e tem a função de buscar a solução para um dado problema. O agente possui as seguintes habilidades: (i) Percepção do Ambiente: característica que torna o agente capaz de enxergar parcialmente o ambiente ao qual faz parte; (ii) Posicionamento: capacidade de se posicionar no espaço de busca

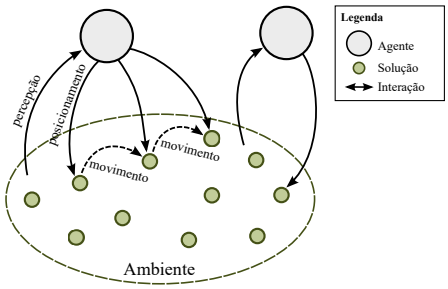

Figura 1: Interação do Agente no *Framework* AMAM [Silva et al., 2019].

da solução, seja através de uma nova solução ou pela escolha de uma solução existente; (iii) Movimento: condição de se movimentar pelo espaço de busca, de uma solução a outra, usando, por exemplo, estruturas de vizinhanças ou operadores genéticos (Algoritmos Genéticos). A Figura 1 apresenta as interações dos agentes com o Ambiente. Observe que o agente tem a capacidade de perceber o ambiente, e, a partir dele, se posicionar e se movimentar pelo espaço busca.

Silva et al. [2019] descreve as habilidades auto adaptativas dos agentes do *Framework* AMAM. Utilizando os conceitos de aprendizado de máquina, as habilidades adaptativas permitem ao agente se adaptar às características específicas do problema de otimização. Neste sentido, os agentes do *Framework* AMAM possuem duas formas de aprendizagem implementadas: (i) Autômatos de Aprendizagem: primeira forma de aprendizagem incorporada ao *framework* AMAM, no qual o aprendizado é semelhante ao apresentado em Narendra e Thathachar [1974] ou à roleta utilizada em Algoritmos Genéticos. Este algoritmo é aplicado na escolha da ordem de vizinhança do método de busca local, baseado na metaheurística *Variable Neighborhood Descend* (VND) [Mladenović e Hansen, 1997]; (ii) *Q-Learning*: também utilizado na escolha da estrutura de vizinhança da busca local, é baseado no método de Aprendizado por Reforço *Q-Learning* e utiliza uma tabela *Q* para determinar os valores dos pares estado-ação no decorrer do processo de busca. Esta tabela define a recompensa recebida por determinada ação.

2.4. Cooperação

O *Framework* AMAM possui estruturas que permitem a interação e cooperação entre os agentes. A principal estrutura responsável pela troca de informações é denominada *Pool* de Soluções. No *Pool* são compartilhadas soluções geradas pelos agentes no decorrer de sua busca. A estrutura de cooperação da AMAM tem, como principal objetivo, guiar os agentes por regiões mais promissoras no espaço de busca. O *Pool* de Soluções possui um tamanho pré-definido e regras de acesso, tanto para leitura quanto para escrita. As soluções são armazenadas no *Pool* ao final de cada iteração do agente e obedecendo às regras de diversidade do próprio *Pool*. O funcionamento do *Pool* é descrito em detalhes em Silva et al. [2019].

3. Problema *pHub* Centro - CMA*p*HCP

Uma das mais importantes variações do problema de localização de *hubs* é o Problema p*Hub* Centro (p*Hub Center Problem* - p*HCP*). 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 de urgência/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 quatro problemas de localização de *hubs*. Kara e Tansel [2000] propõem um novo modelo e

o compara com o modelo básico apresentado anteriormente em Campbell [1994]. Kara e Tansel [2000] apresentam estudos sobre aspectos computacionais e sua complexidade (NP-Difícil) e demonstram que o modelo por eles proposto é melhor em tempo de execução 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. Em Ernst et al. [2009], novas formulações matemáticas são apresentadas e uma nova formulação de otimização inteira mista é proposta 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.

Brimberg et al. [2017b] propõem uma heurística para resolução do Problema de Alocação Simples não Capacitado (USApHCP), baseado na metaheurística *General Variable Neighbourhood Search* (GVNS). O algoritmo GVNS possui, como ferramenta de busca local, a estratégia *Variable Neighbourhood Descent* (VND). Brimberg et al. [2017a] apresentam uma metaheurística *Basic Variable Neighbourhood Search* (BVNS) para resolução do Problema de Alocação Múltipla não Capacitado do p*Hub* Centro (UMApHCP).

Farahani et al. [2013] apresentam uma revisão sobre problemas de localização, discutindo modelos e métodos de solução de diferentes classes desses problemas. Dentro deste contexto, o presente artigo direciona seus esforços para solucionar uma variante do Problema p-*hub* Centro Capacitado de Múltiplas Alocações (CMApHCP), caracterizado na Seção 3.1.

3.1. Caracterização do problema

A caracterização do modelo matemático para o CMApHCP se dá a partir da descrição do UMApHCP, apresentado em Ernst et al. [2009], de forma que os nós *hubs* passam a possuir restrições de capacidade. Desta forma, o CMApHCP 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* possuem restrições de capacidade (capacitado). O CMApHCP é descrito da 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 apresentam restrições de capacidade; o transporte direto entre nós não *hub* não é permitido. 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/penalidades de transporte atribuídos aos *hubs*, conforme Ernst et al. [2009]. Na Expressão (1), 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. Considerando-se que, para efetivar o arco (i, i), quando i é um nó não *hub*, temos que necessariamente passar por um nó *hub*, assume-se, então, que $c_{ii} \geq 0$. O objetivo do problema é minimizar o custo máximo entre os pares origem-destino do grafo. Assim, seja s uma variável de decisão contínua, que representa o valor da função objetivo; z_k uma variável de decisão binária, de modo que $z_k = 1$ se o nó k é definido como *hub* e $z_k = 0$, caso contrário; e y_{ijkl} uma variável de decisão binária de quatro índices, tal que $y_{ijkl} = 1$ se a demanda do arco (i, j) passa pelo arco formado pelos *hubs* (k, l), e $y_{ijkl} = 0$, caso contrário.

Em sistema reais de demanda origem-destino, é mais efetivo considerar restrições de capacidade aos nós *hubs*. Desta forma, a capacidade, normalmente, está ligada tanto ao *hub* de coleta quanto ao de distribuição. Portanto, este modelo do CMApHCP considera a capacidade no *hub* de coleta e no *hub* de distribuição e, assim, o fluxo trafegado é descontado tanto da capacidade do *hub*

de coleta, quanto na capacidade do *hub* de distribuição. Para construir o modelo matemático do CMApHCP com Capacidade Dupla, são incluídas, no modelo matemático do problema não capacitado (UMApHCP), restrições de capacidade, de forma que o modelo completo para o problema é descrito por:

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

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

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

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

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

$$\sum_{i \in N} \sum_{j \in N} w_{ij} \left[\sum_{l \in N} (y_{ijkl} + y_{ijlk}) - y_{ijkk} \right] \leq cap_k, \quad k \in N \tag{7}$$

$$s \geq \sum_{k \in N} \sum_{l \in N} (\gamma c_{ik} + \alpha c_{kl} + \beta c_{lj}) y_{ijkl}, \quad i, j \in N \tag{8}$$

$$y_{ijkl}, z_k \in \{0, 1\}, \quad s \in \mathbb{R}_+, \quad i, j, k, l \in N \tag{9}$$

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* e a Expressão (8) determina que a variável s seja o limitante superior de custo, dentre todas as demandas do grafo. A Expressão (7) inclui restrições de capacidade nos nós de coleta k e distribuição l ($y_{ijkl} + y_{ijlk}$), sendo w_{ij} a demanda entre os nós (i, j) e cap_k a capacidade do nó k. Além disso, para toda demanda distribuída pelo mesmo nó de coleta, o fluxo é descontado apenas da capacidade do nó responsável pela coleta do fluxo ($-y_{ijkk}$).

4. Aplicação VNS_AMAM_CMApHCP

A aplicação VNS_AMAM_CMApHCP foi desenvolvida, a partir do *Framework* AMAM, para solucionar o CMApHCP. Para tal, o sistema multiagente foi instanciado com agentes autônomos denominados VNS_AMAM. O agente VNS_AMAM é composto por uma metaheurística híbrida baseada na metaheurística VNS, combinada com elementos da metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*). A metaheurística híbrida implementada pelo agente faz uso da fase de construção da metaheurística GRASP para construção de soluções iniciais e da busca local baseada na heurística VND.

A Figura 2 apresenta a aplicação VNS_AMAM_CMApHCP. Nela são representados os quatro agentes VNS utilizados na aplicação desenvolvida, assim como suas capacidades de percepção do ambiente e as formas como se posicionam e se movimentam pelo espaço busca. Através do ambiente, os agentes têm também acesso ao *Pool* de soluções, tanto para a inserção de novas soluções, quanto para a leitura das soluções ali incluídas. As seções a seguir apresentam todos os detalhes desta implementação. A instanciação do ambiente do sistema multiagente desenvolvida para o problema CMApHCP é detalhada na Seção 4.1, enquanto a Seção 4.2 apresenta o agente VNS_AMAM e as particularidades do método implementado por ele.

4.1. Definição do ambiente do sistema multiagente

4.1.1. Representação da Solução

A solução do CMApHCP é representada aqui por um vetor Z, de dimensão p, e uma matriz M de valores inteiros, de dimensão $n \times n$, sendo $n = |N|$ o número de nós do grafo. Cada

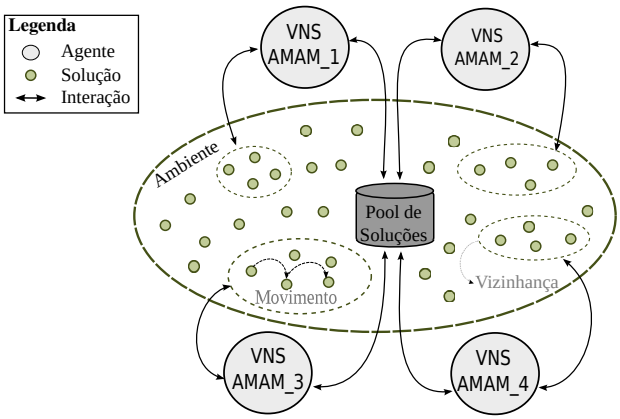

Figura 2: Instanciação do *Framework* AMAM para o CMApHCP - VNS_AMAM_CMApHCP.

elemento do vetor Z representa o nó $n \in N$, que será determinado como nó *hub*. Desta forma, na representação de Z mostrada na Expressão (10), tem-se, para $n = 5$ nós e $p = 3$ *hubs*, os nós 1, 2 e 4 como nós *hub* e os nós 3 e 5 como nós não *hub*.

$$Z = \begin{bmatrix} 4 & 1 & 2 \end{bmatrix} \tag{10}$$

A matriz M, por sua vez, representa o caminho de todo o grafo com demanda origem-destino. Cada elemento (i, j) desta matriz recebe um número que indica o arco que irá atender à demanda (i, j). Um exemplo desta representação é mostrado na Expressão (11). Os arcos são determinados a partir dos índices do vetor Z de *hubs* (de 0 a $p - 1$). Para que seja definido o número que identifica cada arco, utilizamos a fórmula $hub\_coleta \times p + hub\_distribuicao$, em que hub_coleta é o índice do vetor Z correspondente ao hub de coleta e $hub\_distribuicao$ é o índice do vetor Z correspondente ao hub de distribuição. Observe que, para cumprir a demanda $(1, 3)$, é atribuído o arco 2. Isto representa o caminho $1 \rightarrow 4 \rightarrow 2 \rightarrow 3$.

$$M = \begin{bmatrix} 5 & 4 & 2 & 5 & 8 \\ 4 & 0 & 8 & 0 & 2 \\ 7 & 7 & 5 & 7 & 8 \\ 7 & 0 & 0 & 0 & 8 \\ 7 & 8 & 8 & 8 & 8 \end{bmatrix} \tag{11}$$

A Expressão (13) apresenta a numeração dos arcos em um sistema com 3 *hubs*. Note que o arco composto pelo *hub* de coleta com índice 0 e o *hub* de distribuição de índice 2 é nomeado como arco 2. Para se obter o caminho reverso, a partir do arco de transporte e definir os nós de coleta e distribuição, deve-se dividir o valor do arco pelo número p de *hubs* para determinar o índice do *hub* de coleta, e o resto da divisão do arco pelo número p de *hubs*, para determinar o índice do *hub* de distribuição. Desta forma, definindo o hub de coleta e o hub de distribuição do arco 2, tem-se: **coleta:** $2 \div 3 = 0 \rightarrow Z_0 \rightarrow$ nó = 4; **distribuição:** $2 \mod 3 = 2 \rightarrow Z_2 \rightarrow$ nó = 2.

$$\begin{matrix} & 0 & 1 & 2 \end{matrix} \tag{12}$$

$$\begin{matrix} 0 \\ 1 \\ 2 \end{matrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \tag{13}$$

4.1.2. Estrutura de Vizinhança

O agente VNS_AMAM utiliza seis estruturas de vizinhança para explorar o espaço de soluções do CMApHCP. Estas estruturas de vizinhança definem a forma como o agente se movimenta pelo ambiente. Elas são divididas em dois tipos: as estruturas de vizinhança que operam sobre o o vetor Z de *hubs* e as estruturas de vizinhança que operam sobre a matriz de arcos de transporte M. No primeiro grupo são definidas duas estruturas de vizinhança que operam sobre o vetor de *hubs*: (i) `swap_hub`; e (ii) `shift_hub`. A vizinhança obtida pelo movimento `swap_hub` consiste em em trocar um nó *hub* por um nó não *hub*. Por exemplo, uma solução vizinha do vetor Z, apresentado na Expressão (10), é dada pela Expressão (14). Observe que o nó 4 deixou de ser *hub* e o nó 3 é o novo *hub*. Os movimentos referentes a esta estrutura de vizinhança correspondem a trocar i *hubs* do vetor Z, em que i varia de 1 até $p-1$.

$$Z' = \left[\begin{array}{ccc} \mathbf{3} & 1 & 2 \end{array} \right] \tag{14}$$

A vizinhança `shift_hub` consiste em trocar os *hubs* de posição no vetor Z. Como exemplo, a Expressão (15) representa uma solução vizinha do vetor Z mostrado inicialmente na Expressão (10). Note que os *hubs* 2 e 4 foram trocados. Foram definidos $p-1$ movimentos a partir desta estrutura de vizinhança, que consiste em realizar i trocas `shift_hub` na solução, para i variando de 1 até $p-1$.

$$Z' = \left[\begin{array}{ccc} \mathbf{2} & 1 & \mathbf{4} \end{array} \right] \tag{15}$$

No segundo grupo são definidas quatro estruturas de vizinhança que operam sobre a matriz M: (i) `swap_arco`; (ii) `swap_coleta`; (iii) `swap_distribuição`; e (iv) `shift_arco`. A estrutura de vizinhança denominada `swap_arco` consiste em trocar o arco de transporte que atende a uma dada demanda. Para isso, é trocado o arco por um novo arco. Por exemplo, para trocar o arco de entrega da demanda $(1,3)$, que atualmente é atendida pelo arco 2, devemos simplesmente definir um novo arco, gerando uma solução vizinha a M. Note que a demanda $(1,3)$ deixou de ser atendida pelos *hubs* 4 e 2 (arco 2) e passou a ser atendida apenas pelo *hub* 2 (arco 8).

$$M = \begin{bmatrix} 5 & 4 & 2 & 5 & 8 \\ 4 & 0 & 8 & 0 & 2 \\ 7 & 7 & 5 & 7 & 8 \\ 7 & 0 & 0 & 0 & 8 \\ 7 & 8 & 8 & 8 & 8 \end{bmatrix} \xrightarrow{\texttt{swap\_arco}} M' = \begin{bmatrix} 5 & 4 & \mathbf{8} & 5 & 8 \\ 4 & 0 & 8 & 0 & 2 \\ 7 & 7 & 5 & 7 & 8 \\ 7 & 0 & 0 & 0 & 8 \\ 7 & 8 & 8 & 8 & 8 \end{bmatrix} \tag{16}$$

A vizinhança `swap_coleta` consiste em trocar o *hub* de coleta de uma dada demanda. Para alterar o *hub* de coleta da demanda, devemos trocar o arco correspondente a ela, de forma que a troca só pode ser feita por algum arco da mesma linha do *hub* de coleta do arco atual. Para definir qual a linha, efetuamos a operação $arco \div p$. Por exemplo, considerando a matriz M da Expressão (16), para trocar o arco de coleta da demanda $(1,3)$, atendida pelo arco 2, deve-se alterar o arco 2 por um dos arcos contidos na linha 1 da matriz apresentada em (13), que são os arcos 0 e 1.

A estrutura denominada `swap_distribuição` consiste em trocar o arco de transporte, de forma que apenas o *hub* de distribuição seja alterado. Para isso, o arco só pode ser trocado por um dos arcos contidos na coluna obtida pela operação $arco \mod p$. Por exemplo, para trocar o *hub* de distribuição da demanda $(1,3)$ da matriz M da Expressão (16)), atendida pelo arco 2, somente os arcos 5 e 8 podem ser considerados. Por fim, a estrutura de vizinhança `shift_arco` consiste em trocar o arco de duas demandas do grafo. Como exemplo, a Expressão (17) representa uma solução vizinha (M') à solução apresentada na Expressão (11). Observe que os arcos das demandas $(1,3)$ e

$(5, 3)$ foram trocados.

$$M = \begin{bmatrix} 5 & 4 & \mathbf{2} & 5 & 8 \\ 4 & 0 & 8 & 0 & 2 \\ 7 & 7 & 5 & 7 & 8 \\ 7 & 0 & 0 & 0 & 8 \\ 7 & 8 & \mathbf{8} & 8 & 8 \end{bmatrix} \xrightarrow{\texttt{shift\_arco}} M' = \begin{bmatrix} 5 & 4 & \mathbf{8} & 5 & 8 \\ 4 & 0 & 8 & 0 & 2 \\ 7 & 7 & 5 & 7 & 8 \\ 7 & 0 & 0 & 0 & 8 \\ 7 & 8 & \mathbf{2} & 8 & 8 \end{bmatrix} \quad (17)$$

4.1.3. Função de Avaliação

A avaliação de uma solução sol é dada por:

$$f(sol) = \max\{C_{ij}\} + \max\left\{0, -\beta\left[\sum_{k=1}^{p}(cap_k - f\_hub_k) \ : \ (cap_k - f\_hub_k) < 0\right]\right\}, \ i, j \in N \quad (18)$$

Este valor deve ser minimizado. A condição $\max\{C_{ij}\}$ consiste em encontrar o maior custo, calculado dentre todas as demandas (i, j) da instância. Ela é somada ao maior fator entre zero (0) e a somatória da capacidade excedida dos *hubs*, multiplicada por um fator de penalidade β. Nesta expressão, f_hub_k representa o fluxo trafegado no *hub* k, segundo a forma de atribuição de capacidade, e o fator de penalização β é definido como o custo em percorrer todas as demandas (i, j).

4.2. Agente VNS_AMAM

O método implementado nesta proposta pelo agente do *Framework* é a metaheurística híbrida formada pela metaheurística VNS (*Variable Neighborhood Search*), tendo a construção da solução inicial baseada na fase inicial da metaheurística GRASP (*Greedy Randomized Adaptative Search Procedure*), conforme apresentado na Seção 4.2.1.

O VNS é um método de busca local [Mladenović e Hansen, 1997], que utiliza de uma sequência de vizinhanças, $V_1, \ldots, V_k$, para tentar escapar de ótimos locais ou pesquisar diferentes ótimos locais. Mladenović e Hansen [1997] apresentam três fatores explorados pelo VNS: (i) o ótimo local de uma vizinhança não é necessariamente o ótimo local de outra estrutura de vizinhança; (ii) um ótimo global é um ótimo local para k estruturas de vizinhanças; (iii) em muitos problemas de otimização, ótimos locais de diferentes estruturas de vizinhanças estão próximos entre si.

O processo de busca do VNS consiste em criar uma solução inicial; aplicar uma busca local, retornando um ótimo local e tomando esta solução como a solução corrente. A partir daí, enquanto o critério de aceitação não é alcançado, gera-se um vizinho (s') a partir da estrutura de vizinhança V_k. Em seguida é aplicada novamente a busca local no vizinho s'. Caso a solução seja melhor que a solução corrente, a solução corrente passa a ser s' e a estrutura de vizinhança selecionada é $k = 1$. Se o vizinho não for melhor que a solução corrente, a vizinhança passa a valer $k + 1$. Este processo é repetido até $k = |V|$. Ao final do processo, retorna-se um ótimo local em relação a $k = |V|$ estruturas de vizinhanças.

As estruturas de vizinhança utilizadas na fase de geração de vizinhos são as estruturas `swap_hub` e `shift_hub`, apresentadas na Seção 4.1.2. Além destas duas estruturas, uma terceira vizinhança é utilizada e corresponde à coleta de uma solução no *Pool* de Soluções do *Framework*. Esta estrutura de vizinhança permite ao agente buscar informações disponibilizadas por outros agentes no ambiente.

4.2.1. Construção da Solução Inicial

No algoritmo proposto, a implementação da construção da solução inicial se dá por uma construção gulosa aleatória, semelhante à fase de construção inicial da metaheurística GRASP [Feo e Resende, 1995]. A construção da solução é dividida em duas etapas: (i) o vetor Z de *hubs*

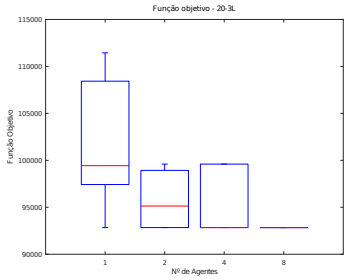

Figura 3: Boxplot dos resultados para a instância AP20_3L, com 1, 2, 4 e 8 agentes.

é construído; (ii) os nós não *hubs* são alocados aos *hubs* de coleta e distribuição. Inicialmente, é definida uma lista restrita RCL_n de candidatos e, logo em seguida, é atribuído a cada nó t o fator custo beneficio.Este fator é obtido calculando a maior distância entre ele e todos os nós j dividida pela capacidade do nó t, sendo definido pelo função guia $g(t)$, dada por $g(t) = \max(d_{ti}) \div cap_t \ \ \forall\, i \neq t, i \in C$. Nesta expressão, C corresponde aos nós candidatos a serem *hubs* na solução, d_{ti} é o custo de saída do nó t e chegada ao nó i e cap_t é a capacidade do nó t. Após, define-se, aleatoriamente, p *hubs*, condicionados ao melhor valor de custo beneficio encontrado na RCL_n. A segunda etapa consiste em alocar os nós não *hubs* aos *hubs*. Para isso é utilizado o algoritmo de caminho mínimo de Floyd-Warshall.

4.2.2. Busca Local

A técnica de busca local implementada pelo `VNS_AMAM` é baseada no método VND (*Variable Neighborhood Descend*). Este método consiste em encontrar ótimos locais considerando as n estruturas de vizinhança. Seguindo uma determinada ordem de vizinhança, a busca local é realizada considerando a vizinhança corrente. Caso a solução gerada seja melhor que a atual, ela passa a ser a solução corrente e a primeira estrutura de vizinhança é novamente usada. Caso contrário, a estrutura de vizinhança seguinte é aplicada. Esse processo é repetido até que as n estruturas de vizinhanças não consigam melhorar a solução corrente.

O VND faz uso das estruturas de vizinhança que operam sobre a matriz de arcos de transporte, apresentadas na Seção 4.1.2. A ordem de utilização destas estruturas é: (i) `swap_arco`; (ii) `swap_coleta`; (iii) `swap_distribuição` e; (iv) `shift_arco`. Esta ordem é definida a partir da complexidade das estruturas de vizinhança.

5. Resultados para o CMApHCP usando o Agente `VNS_AMAM`

Testes computacionais foram realizados com o objetivo de validar a aplicação `VNS_AMAM_CMApHCP`. Para a realização destes experimentos, utilizou-se o conjunto de instâncias AP - (*Australian Post*), introduzido em Ernst e Krishnamoorthy [1996], que foi constituído a partir de dados reais do Serviço Postal Australiano. As instâncias deste conjunto possuem até 200 nós. Todos os testes computacionais foram executados por 30 vezes em um computador *Intel Core I5* (8250U), 1,6 GHz (4 Cores 8 *Threads*), 8 GB de memória RAM e sistema operacional *Ubuntu* 19.04.

Os testes foram realizados com instâncias tendo até 25 nós, em que os resultados exatos, apresentados em Silva et al. [2020], representam soluções ótimas. Os fatores de desconto/penalidade usados no cálculo do custo de transporte foram $\gamma = 3,0$; $\alpha = 0,75$; e $\beta = 2,0$. O conjunto de instâncias utilizado foi o conjunto de instâncias `L` (menos apertados) e o critério de parada foi o tempo computacional de 7200 segundos. Os resultados computacionais obtidos foram comparados com os resultados determinados pelo `Concert_CPLEX` mostrados em Silva et al. [2020]. Os resultados são apresentados na Tabela 1.

Tabela 1: Resultado com diferentes números de agentes

Instância	Concert CPLEX	VNS AMAM	*gap*% FO	Tempo (s) VNS_AMAM
AP10_2	99805,28	99805,28	0,00	35,30
AP10_3	70337,49	70337,49	0,00	33,70
AP10_4	68714,17	68714,17	0,00	40,12
AP10_5	55439,28	55439,28	0,00	53,81
AP20_2	110220,25	110220,25	0,00	200,00
AP20_3	92839,94	92839,94	0,00	501,98
AP20_4	82439,73	82439,73	0,00	520,31
AP20_5	74162,48	74162,48	0,00	303,76
AP20_10	47794,95	47794,95	0,00	123,54
AP25_2	118497,02	118497,02	0,00	3624,31
AP25_3	102737,89	102737,89	0,00	987,45
AP25_4	89747,25	-	-	7200,00
AP25_5	82234,52	-	-	7200,00
AP25_10	53964,09	53964,09	0,00	997,63

A Tabela 1 apresenta os resultados obtidos na execução do VNS_AMAM para a solução do CMApHCP com penalidade de coleta e entrega (3-2). A coluna "Instância" apresenta a instância testada. As colunas "Concert_CPLEX" e "VNS_AMAM" apresentam os resultados obtidos, respectivamente, com os métodos exato e heurístico para o CMApHCP. A Coluna "*gap* $\%$" apresenta o *gap* da solução heurística em relação à solução exata. A coluna "Tempo(s) VNS_AMAM" apresenta a média do tempo computacional obtido utilizando o método VNS_AMAM_CMApHCP.

Os resultados mostram que a aplicação VNS_AMAM_CMApHCP obtém os resultados ótimo, apresentados em Silva et al. [2020], nas instâncias testadas, exceto, nas instâncias AP25_4 e AP25_5, em que, não foi encontrado solução factível com o tempo computacional estabelecido. A Figura 3 apresenta os resultados obtidos com 30 execuções da instância AP20_3L. Observe que à medida que novos agentes são adicionados, os resultados se aproximam do ótimo para a instância. Além disso, deve-se destacar que todas as execuções com 8 agentes obtiveram a solução ótima.

6. Considerações finais e direções futuras

Este artigo propõe a utilização de uma aplicação do *Framework* AMAM para a solução do problema CMApHCP. O método VNS_AMAM_CMApHCP implementado alcançou, com o limite de tempo estabelecido, as soluções ótimas obtidas pelo método Concert_CPLEX de Silva et al. [2020]. Os resultados encontrados para o problema capacitado mostram que a utilização de sistemas multiagentes é eficiente. A continuidade destes estudos mostra a necessidade de aprofundamento da aplicação desenvolvida, bem como a possibilidade de se utilizar novas técnicas para solução. A escalabilidade da aplicação é confirmada nos experimentos, já que os resultados melhoram à medida que novos agentes são inseridos na busca.

Agradecimentos

Os autores agradecem à CAPES (código de financiamento 001), à FAPEMIG (processo PPM CEX 0676/17), ao CNPq (processo 303266/2019-8), ao CEFET-MG, à UFV-campus Florestal e à UFOP pelo apoio ao desenvolvimento deste trabalho.

Referências

Alumur, S. e Kara, B. Y. (2008). Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1 – 21. ISSN 0377-2217.

Brimberg, J., Mladenović, N., Todosijević, R., e Urošević, D. (2017a). A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem. *Optimization Letters*, 11(2):313–327. ISSN 1862-4480.

Brimberg, J., Mladenović, N., Todosijević, R., e Urošević, D. (2017b). General variable neighborhood search for the uncapacitated single allocation p-hub center problem. *Optimization Letters*, 11(2):377–388. ISSN 1862-4480.

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. ISSN 0377-2217.

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. ISSN 0305-0548.

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

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(4):1096 – 1109. ISSN 0360-8352.

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

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. ISSN 0377-2217.

Mladenović, N. e Hansen, P. (1997). Variable neighbourhood search. *Computers & Operations Research*, 24(11):1097–1100.

Narendra, K. S. e Thathachar, M. A. L. (1974). Learning automata - a survey. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-4(4):323–334. ISSN 0018-9472.

Parejo, J. A., Ruiz-Cortés, A., Lozano, S., e Fernandez, P. (2012). Metaheuristic optimization frameworks: a survey and benchmarking. *Soft Computing*, 16(3):527–561.

Silva, J. F., Silva, M. A. L., de Sa, 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.

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 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.