PRODUÇÃO ONLINE
REVISTA CIENTÍFICA ELETRÔNICA DE ENGENHARIA DE PRODUÇÃO
ISSN 1676-1901

ALGORITMOS HEURÍSTICOS E FORMULAÇÕES EXATAS APLICADOS AO PROBLEMA DE ESTRATIFICAÇÃO DE UNIDADES PRIMÁRIAS DE AMOSTRAGEM

HEURISTIC ALGORITHMS AND EXACT FORMULATIONS APPLIED TO THE PROBLEM OF STRATIFICATION OF PRIMARY SAMPLING UNITS

José André de Moura Brito* E-mail: jose.m.brito@ibge.gov.br
Gustavo Silva Semaan** E-mail: gustavosemaan@ufrrj.br
*Instituto Brasileiro de Geografia e Estatística (IBGE), Rio de Janeiro, RJ, Brasil.
** Universidade Federal Rural do Rio de Janeiro (UFRRJ), Três Rios, RJ, Brasil.

Resumo: Este artigo investiga o Problema de Estratificação de Unidades Primárias de Amostragem (PEUPA), um caso real e de elevada complexidade computacional, formulado como um Problema de Agrupamento Capacitado. Foram desenvolvidas e avaliadas múltiplas abordagens de otimização, incluindo a Heurística da Envoltória Convexa, quatro metaheurísticas baseadas no Otimizador de Chaves Aleatórias (BRKGA, ILS, LNS e VNS), um algoritmo GRASP e duas formulações de Programação Matemática. Experimentos computacionais conduzidos com 30 instâncias reais do Censo Demográfico 2022, de diferentes portes, foram analisados por meio de Testes de Hipóteses — especificamente, o Teste de Friedman (não paramétrico) e o Teste de Wilcoxon pareado com Correção de Bonferroni —, revelando diferenças estatisticamente significativas entre as abordagens. Os resultados demonstram a superioridade e a estabilidade do GRASP frente às demais técnicas, consolidando-o como uma solução eficiente e promissora para a resolução do PEUPA em contextos práticos de amostragem estatística.

Palavras-chave: Amostragem. Agrupamento. Metaheurísticas. Programação Matemática. Otimização.

Abstract: This article tackles the Primary Sampling Units Stratification Problem (PEUPA)—a complex real-world challenge—by formulating it as a Capacitated Clustering Problem. Several optimization approaches were developed and evaluated, including the Convex Hull Heuristic, four metaheuristics leveraging the Random-Key Optimizer framework (BRKGA, ILS, LNS, and VNS), a GRASP algorithm, and two Mathematical Programming formulations. Computational experiments on 30 real instances from the 2022 Demographic Census database, encompassing diverse problem sizes, were assessed through rigorous Hypothesis Testing, including the non-parametric Friedman Test and the pairwise Wilcoxon Test with Bonferroni Correction. The results consistently demonstrate that GRASP outperforms the other approaches, providing a stable, efficient, and practical solution for the PEUPA in statistical sampling applications.

Keywords: Sampling. Clustering. Metaheuristics. Mathematical Programming. Optimization.

1 INTRODUÇÃO

Vivemos na era da informação e do Big Data (Tosi, 2024), em que os institutos de estatística oficial produzem e disponibilizam, diariamente, uma grande quantidade de dados e estatísticas, obtidos mediante a realização de suas pesquisas amostrais e levantamentos censitários. Tais estatísticas são utilizadas para os mais variados fins pelos governos, pesquisadores e população em geral. Em específico, no caso do Brasil, os gestores municipais, estaduais e federais, uma vez providos de tais informações, têm a possibilidade de planejar e implementar diversas políticas públicas que proporcionem o bem-estar geral para a população, traduzido, por exemplo, em melhorias na educação, segurança pública e saúde (Bispo Dos Santos, 2025; Batista *et al*., 2016; Mendes, 2015)

Considerando esta grande demanda, aliada à utilidade dessas informações, os institutos oficiais de estatística têm buscado, cada vez mais, o aprimoramento de suas metodologias de pesquisa, atentando ao equilíbrio em um binômio que abarca a qualidade - traduzida em estatísticas precisas e acuradas - e as restrições orçamentárias impostas. Ainda neste sentido, o "sucesso" no equacionamento deste binômio tem um reflexo direto na implementação de planos amostrais de qualidade e na consequente realização das pesquisas (Cochran, 1977; Lohr, 2021).

A depender do objetivo da pesquisa, da população alvo (Bolfarine; Bussab, 2005) a ser investigada, do orçamento disponível, da precisão definida e do cadastro para seleção da amostra, podem ser adotados/combinados diferentes esquemas de amostragem como, por exemplo, Amostragem Aleatória Simples, Amostragem de conglomerados ou a Amostragem Estratificada (Cochran, 1977; Lohr, 2021).

Em particular, a aplicação de um Plano Amostral com Amostragem Estratificada (AE) exige a definição prévia de estratos, que correspondem a grupos mutuamente exclusivos e constituídos por unidades de amostragem como, por exemplo, Setores Censitários[1], que apresentem alta homogeneidade (baixa variância interna). Além disso, em algumas pesquisas, por considerar questões operacionais e/ou metodológicas, tais estratos devem conter um número mínimo de unidades.

[1] Unidade territorial de coleta e divulgação de dados estatísticos do IBGE.

Um exemplo de problema em que este tipo de restrição aparece diz respeito à Amostra Mestra (IBGE, 2007; Malaguti; Alves, 2024), que corresponde a um conjunto de unidades de área selecionadas de um cadastro, segundo um método probabilístico de seleção. Ao definir uma amostra mestra tem-se, como objetivo, selecionar subamostras para as diversas pesquisas como, por exemplo, aquela utilizada pelo IBGE na realização da PNAD-Contínua (IBGE, 2023).

Em uma das etapas de definição da amostra mestra, devem ser formados estratos constituídos por, pelo menos, 150 Unidades Primárias de Amostragem (UPAs). Ainda neste sentido, tais estratos devem ter alto grau de homogeneidade — associado ao menor valor possível de uma expressão de estimador de variância (IBGE, 2007; Albieri; Dias, 2017). Assim, define-se o Problema de Estratificação de UPAs, referido neste artigo como PEUPA.

Esta aplicação real pode ser mapeada em um Problema de Agrupamento Capacitado (PAC), cuja função objetivo é não linear (variância do estimador) e a restrição de capacidade está relacionada ao número mínimo de UPAs por estrato. Assim como outros problemas de agrupamento, o PAC é de difícil resolução computacional, característica que demanda, com frequência, a combinação de abordagens híbridas de otimização, envolvendo, por exemplo, o uso de formulações (Negreiros *et al*., 2022) e algoritmos heurísticos baseados em metaheurísticas como Busca Tabu, GRASP e Algoritmos Genéticos, dentre outras, conforme apresentado no *survey* de Levin (2024).

Considerando a relevância e a complexidade do PEUPA, este artigo traz a proposta de um conjunto de algoritmos baseados em: (i) metaheurística GRASP (Resende; Ribeiro, 2016); (ii) adaptação da heurística da envoltória convexa (HEC) (GUIGNARD; AHLATCIOGLU, 2021) e (iii) metaheurísticas (Martí; Pardalos; Resende, 2018) *Biased Random-Key Genetic Algorithm* (BRKGA), *Iterated Local Search* (ILS), *Large Neighborhood Search* (LNS) e *Variable Neighborhood Search* (VNS).

Em particular, os algoritmos derivados dessas metaheurísticas, foram adaptados e implementados a partir da combinação de um conjunto de procedimentos, disponíveis no Otimizador de Chaves Aleatórias (OCA), proposto em (Chaves *et al*., 2024). Complementarmente, foram implementadas duas formulações

para o PEUPA, sendo uma de programação linear inteira mista e a outra de programação quadrática inteira.

Considerando o exposto nesta seção, o objetivo deste trabalho consiste em investigar diferentes abordagens de otimização aplicadas ao Problema de Estratificação de Unidades Primárias de Amostragem (PEUPA), incluindo formulações de programação linear inteira, heurística e metaheurísticas. Busca-se avaliar a capacidade dessas abordagens em produzir estratos homogêneos, respeitando as restrições operacionais do PEUPA, bem como comparar seu desempenho computacional em instâncias reais derivadas do Censo Demográfico de 2022.

Além da introdução, este artigo está dividido em mais quatro seções. A seção dois traz uma descrição do PEUPA e a sua relação com o problema de agrupamento capacitado, além de uma revisão da literatura. A seção três detalha o algoritmo GRASP, o algoritmo baseado na HEC e finaliza apresentando uma visão geral do OCA, incluindo os procedimentos de uso comum nas quatro metaheurísticas utilizadas neste trabalho, o decodificador proposto e a função de penalidade adotada. A seção quatro traz um conjunto de resultados computacionais e análises estatísticas, derivadas da aplicação dos seis algoritmos e das duas formulações em um conjunto de 30 instâncias, construídas a partir de dados do censo demográfico de 2022 - disponíveis no site do IBGE. Por fim, a seção 5 traz as conclusões e futuros desdobramentos do trabalho.

2 AMOSTRA MESTRA E O PROBLEMA DE ESTRATIFICAÇÃO DE UNIDADES PRIMÁRIAS DE AMOSTRAGEM

Durante a fase de planejamento das pesquisas, as equipes técnicas dos institutos de estatística têm, como uma de suas tarefas, definir as amostras levando em conta, por exemplo, o cadastro, o orçamento disponível e o nível de precisão, definido previamente e relacionado às estatísticas que serão produzidas a partir da aplicação das pesquisas. Para alcançar tal objetivo de forma eficaz, utiliza-se a amostragem probabilística (Lohr, 2021; Bolfarine; Bussab, 2005; Cochran, 1977), que abarca uma série de métodos, incluindo a amostragem aleatória simples (AAS) e a amostragem estratificada (AE).

Em particular, visando melhorar a precisão das estimativas que serão produzidas para a população que será investigada, além de atender questões administrativas, elabora-se um plano amostral que contemple a amostragem estratificada (Lohr, 2021; Bolfarine; Bussab, 2005; Cochran, 1977). Neste tipo de amostragem, a partir das informações da população disponíveis no cadastro - insumo indispensável às etapas de planejamento e aplicação da pesquisa -, a população é dividida em grupos mutuamente exclusivos, denominados estratos.

No âmbito do IBGE, conforme descrito em IBGE (2023) e IBGE (2007), a Amostragem Estratificada é utilizada, por exemplo, na definição de grupos de UPAs, o que implica resolver o PEUPA. Este problema é abordado na etapa de planejamento de pesquisas domiciliares, com o objetivo de integrar algumas pesquisas realizadas pelo IBGE. Mais especificamente, trata-se da utilização de um único cadastro de seleção e de uma amostra comum, chamada de Amostra Mestra (Malaguti; Alves, 2024), composta por um conjunto de unidades de área selecionadas a partir de um cadastro, utilizando um método probabilístico de seleção (Bolfarine; Bussab, 2005). A partir dela é possível formar subamostras para atender a diferentes pesquisas. Nesta amostra, as UPAs correspondem a agrupamentos de setores censitários, que são definidos com base em critérios específicos, conforme discutido em IBGE (2007) e Albieri e Dias (2017).

A utilização de uma amostra mestra corresponde a uma prática adotada por diversos institutos oficiais de estatística ao redor do mundo, com o objetivo de racionalizar custos e melhorar a qualidade das pesquisas domiciliares. Em Malaguti e Alves (2024) são citados exemplos sobre a tentativa de implementação de amostras mestras pelo *Statistics South Africa*, pelo *National Statistics Office das Filipinas*, pelo *Bangladesh Bureau of Statistics* e pelo *Statistics Canada*. Também na Argentina, o *Instituto Nacional de Estadística y Censos* (INDEC) utiliza uma amostra mestra probabilística, conglomerada e estratificada por áreas, sendo as UPAs definidas por aglomerados urbanos e as unidades secundárias são setores censitários, selecionados com base na quantidade de moradores (Muiños, 2018). Já na Colômbia, o *Departamento Administrativo Nacional de Estadística* (DANE) mantém um Cadastro Mestre atualizado, e utiliza uma amostra probabilística e estratificada (Dane, 2016).

No caso do Brasil, o IBGE deu início a um projeto de reformulação de suas pesquisas domiciliares amostrais durante os anos 2000, tendo em vista uma crescente demanda por informações socioeconômicas e demográficas vinda dos mais diferentes setores e grupos, incluindo, por exemplo, gestores e pesquisadores.

> A Amostra Mestra do IBGE é planejada com base em informações obtidas nos censos demográficos, e é amparada na Base Territorial (que contém todos os setores censitários) preparada para cada censo. Decorre então que qualquer avanço metodológico dos censos será refletido na Amostra Mestra e, subsequentemente, nas pesquisas amostrais domiciliares (MALAGUTI; ALVES, 2024).

Em uma etapa importante, relativa ao planejamento da Amostra Mestra, deve-se definir os estratos estatísticos, cada um composto por um número mínimo de UPAs, tendo como referência a avaliação de uma expressão de variância que deve ser minimizada, de forma a garantir a produção de estratos mais homogêneos[2]. Mais especificamente, cada estrato deve ter, pelo menos, 150 UPAs, sendo a alocação das UPAs aos estratos efetuada mediante avaliação da variância do estimador de total da renda domiciliar (IBGE, 2007; Albieri; Dias, 2017), considerando o plano amostral normalmente adotado em pesquisas domiciliares, qual seja, a amostragem conglomerada (Cochran, 1977; Lohr, 2021) com seleção de UPAs com probabilidade proporcional ao número de Domicílios Particulares Permanentes (DPPs).

Em resumo, busca-se minimizar a expressão de variância definida em (1), que contempla a soma das variâncias V_h (definidas por (2)-(3)) dos estratos denotados por E_h ($h=1, ..., L$), sendo N_i e N_j iguais, respectivamente, ao número de DPPs[3] na *i-ésima* e *j-ésima* UPAs e Y_i e Y_j iguais à renda total domiciliar na *i-ésima* e *j-ésima* UPAs.

$$V(t_x) = \sum_{h=1}^{L} V_h \tag{1}$$

$$V_h = \sum_{\forall i,j \in E_h} d_{ij} \tag{2}$$

tal que

$$d_{ij} = N_i N_j \left(\frac{Y_i}{N_i} - \frac{Y_j}{N_j} \right)^2 \tag{3}$$

[2] Implica produzir estatísticas com maior precisão a partir das amostras de UPAs selecionadas dos estratos.

[3] Número de domicílios particulares permanentes. Correspondem a construções destinadas a servir de moradia a uma ou mais pessoas e que, na data de referência do censo, eram utilizadas como tal.

Considerando esta definição, o PEUPA pode ser mapeado em um conhecido problema da literatura, qual seja, o problema de agrupamento capacitado com soma mínima de distâncias (PACSMD) (Levin, 2024). O problema de agrupamento capacitado (PAC) (Levin, 2024; Zhou *et al*., 2018; Martinez-Gavara *et al*., 2017) já é, por si só, de difícil resolução computacional (Mulvey; Beck, 1984) e que, por sua vasta aplicabilidade, tem sido muito estudado nas últimas décadas. O problema de agrupamento com soma mínima de distâncias (PASMD) é abordado, por exemplo, em Hansen e Jaumard (1997), que traz uma discussão de como certos problemas de agrupamento são formulados a partir de modelos de Programação Inteira, Programação Linear e Programação Quadrática. O PASMD também possui abordagens para sua resolução por meio de algoritmos baseados nas metaheurísticas VNS, AG e GRASP (Brito; Brito, 2008; Nascimento; Toledo; Carvalho, 2010; Serpa, 2011).

Em Levin (2024) é apresentado um *survey* que traz uma descrição completa desses três tipos de problemas, além de outros problemas de agrupamento correlatos e um substancial conjunto de referências com variadas abordagens, incluindo algoritmos baseados nas metaheurísticas VNS, GRASP e Busca Tabu, dentre outras. Adicionalmente, em Albieri e Dias (2017) e Brito, Montenegro e Freitas (2011) podem ser encontradas abordagens heurísticas, baseadas na metaheurística ILS e em Otimização Microcanônica (Montenegro; Torreão; Maculan, 2003), propostas para a resolução do PEUPA.

Por fim, em Negreiros *et al*. (2022) é apresentada uma revisão ampla de modelos de agrupamento capacitado aplicados a diversos problemas reais. Os autores propõem diferentes formulações matemáticas que agregam restrições de capacidade, típicas de cenários práticos como logística, saúde e estatística. Além disso, destacam como a inclusão dessas restrições torna o problema de agrupamento mais desafiador do ponto de vista computacional; fato que demanda o uso combinado de técnicas de programação inteira, metaheurísticas e métodos híbridos.

2.1 Formulações para o PEUPA

Ao considerar a expressão da variância como função objetivo e a restrição de capacidade relativa ao mínimo de 150 UPAs por estrato, o PEUPA pode ser inicialmente formulado como um problema de Programação Quadrática Inteira (PQI) (Chaovalitwongse; Androulakis; Pardalos, 2008; Lee; Leyffer, 2012; Yagi *et al*., 2022), conforme apresentado a seguir:

$$F1\text{: Minimizar } \sum_{h=1}^{L}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_i^h x_j^h \tag{4}$$

$$\sum_{h=1}^{L} x_i^h = 1; i = 1, \dots, n \tag{5}$$

$$\sum_{i=1}^{n} x_i^h \geq 150; h = 1, \dots, L \tag{6}$$

$$x_i^h \in \{0,1\}; i = 1, \cdots, n; h = 1, \dots, L \tag{7}$$

Aplicando-se à função objetivo de F1 a linearização utilizada em Nascimento *et al*., (2010), onde é abordado o PASMD, produz-se a seguinte formulação de Programação Linear Inteira Mista (PLIM):

$$F2\text{: Minimizar } \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}z_{ij} \tag{8}$$

sujeito a (5)-(7)

$$x_i^h + x_j^h - 1 \leq z_{ij}; h = 1, \dots, L, i = 1, \dots, n-1, j = i+1, \dots, n \tag{9}$$

$$z_{ij} \geq 0; i = 1, \dots, n-1, j = i+1, \dots, n \tag{10}$$

Nestas duas formulações, a função objetivo a ser minimizada em (4) e (8) corresponde à expressão de variância definida em função das equações (1)-(3). A variável binária x_{ih} assume valor 1 se a *i-ésima* UPA é alocada ao *h-ésimo* estrato E_h ($h=1, \dots, L$) e z_{ij} é a variável que assume valor 1 se, simultaneamente, duas UPAS *i* e *j* são alocadas a um mesmo estrato. A restrição (5) garante que cada uma das *n* UPAs será alocada a exatamente um dos estratos E_h; a restrição (6) garante que cada um dos estratos terá, pelo menos, 150 UPAs alocadas. As restrições do tipo (9) garantem

que z_{ij} assume valor um se, simultaneamente, x_i^h e x_j^h assumem valor 1. Por fim, a Tabela 1 traz as expressões relacionadas aos totais de variáveis e restrições das formulações F1 e F2.

Tabela 1 - Números de variáveis e restrições das duas formulações

Formulação	Variáveis	Restrições
F1	$n \cdot L$	$n + L$
F2	$n \cdot L + \frac{n(n-1)}{2}$	$(n + L) + L \cdot \frac{n(n-1)}{2}$

Fonte: Autoria própria.

Conforme será detalhado na seção seguinte, a formulação F1 serviu como base para a implementação de um algoritmo adaptado a partir da HEC, proposta por Guignard e Ahlatcioglu (2021). Ambas as formulações foram implementadas e executadas no solver GUROBI, que contém rotinas destinadas à resolução de problemas de Programação Quadrática Inteira (Yagi *et al*., 2022) e de Programação Linear Inteira Mista (Wolsey, 1998; Conforti *et al*., 2014).

3 ALGORITMOS PROPOSTOS

A presente seção descreve o conjunto de algoritmos que foram implementados e aplicados à resolução do PEUPA, a saber: (1) um algoritmo baseado na metaheurística GRASP (Resende; Ribeiro, 2016); (2) uma adaptação da Heurística da Envoltória Convexa, proposta em Guignard e Ahlatcioglu (2021); e (3) algoritmos baseados nas metaheurísticas BRKGA, ILS, LNS e VNS, implementadas - de forma geral - no Otimizador de Chaves Aleatórias (OCA) proposto em (Chaves *et al*., 2024), o qual pode ser utilizado para resolver diversos problemas de otimização combinatória, incluindo problemas de agrupamento.

3.1 Algoritmo GRASP

O GRASP (Resende; Ribeiro, 2016) é uma metaheurística que é aplicada em duas etapas - Construção e Busca Local. Na etapa de construção, uma solução viável inicial s_o é formada de maneira iterativa, mediante aplicação de um procedimento semi-guloso. Na segunda etapa é aplicada sobre s_o uma busca local, que tem como

objetivo produzir uma solução s' tal que $f(s') < f(s_o)$ (problema de minimização). A melhor das soluções s', obtida após as m iterações completas, consistindo da aplicação da construção e da busca local, corresponde à solução do problema. Na construção da solução, gera-se uma Lista de Candidatos (LC), contendo os elementos que podem ser adicionados à solução parcial s_p (solução s_o em construção). Esses elementos são avaliados aplicando-se uma função gulosa $g(.)$ e são ordenados conforme seu desempenho. O melhor elemento é incorporado a s_p, a LC é atualizada, e o processo se repete até que LC=∅. Para introduzir diversidade, utiliza-se uma Lista Restrita de Candidatos (LRC), composta pelos elementos da LC que, se inseridos em s_p, produzem o menor acréscimo na função objetivo (problemas de minimização).

De acordo com Resende e Ribeiro (2016), para definir a LRC, um dos critérios que pode ser considerado diz respeito ao uso de um parâmetro α. Em linhas gerais, em cada iteração da fase de construção, são calculados os valores de g_{min} e g_{max}, que correspondem, respectivamente, ao menor e maior acréscimo produzidos pela inserção de um elemento da LC na solução parcial s_p, segundo a função $g(.)$. Assim sendo, temos a LRC definida da seguinte forma:

$$LCR = \{t \in LC | g(t) \leq g_{min} + \alpha\,(g_{max} - g_{min})\}\ \alpha \in [0,1] \tag{11}$$

De acordo com o que está preconizado na literatura, recomenda-se o uso de valores intermediários para este parâmetro, com o objetivo de melhorar o desempenho do GRASP, evitando soluções "muito determinísticas" (gulosas) ou "muito aleatórias" (comumente com pouca qualidade). A seguir, com base nos conceitos do GRASP, apresentamos os procedimentos de construção e busca local que foram propostos e implementados para a resolução do PEUPA.

3.1.1 Procedimento de Construção

Uma solução é representada por L conjuntos E^k ($k=1, ...,L$) aos quais todas as n UPAs serão alocadas. Cada conjunto E^k é inicialmente definido por dois valores inteiros, correspondentes aos índices das UPAs selecionadas dentre as n UPAs disponíveis na LC.

Para definir quais UPAs (índices) da LC serão alocadas aos L estratos, define-se uma lista de candidatos restrita LRC, que contém, em cada uma de suas componentes, duplas de índices associados às UPAs candidatas à inserção nos L conjuntos E^k. Nesta etapa, busca-se escolher duplas de UPAs (i, j) tais que d_{ij} seja mínima (expressão definida na equação (3)), sendo a escolha dessas duplas e a consequente definição dos conjuntos E^k efetuada em L etapas, seguindo o procedimento apresentado no pseudocódigo a seguir (Figuras 1 e 2).

Nas linhas 1-3 define-se a LC, os conjuntos E^k e calcula-se a matriz de distâncias D, cujas entradas correspondem aos valores d_{ij} obtidos a partir da equação (3). Entre as linhas 5-17 são construídos os conjuntos E^k $(k=1, ..., L)$, avaliando-se, para cada E^k, a inclusão de duas UPAs i e j com o menor valor de d_{ij}. Para isso, são selecionadas a partir da LC (linha 7), em cada uma das L iterações, n_a duplas de índices (incluídos em uma matriz $A_{na\times 2}$) relativos às possíveis duplas de UPAs por estrato, sendo calculados os valores de g, g_{min}, g_{max} (linhas 8-12) utilizados na definição da (LRC) (linha 13). Por fim, seleciona-se da LRC um elemento r associado ao índice da linha da matriz A, sendo as entradas $A[r,1]$ e $A[r,2]$ correspondentes aos índices - das UPAs - que serão incluídas no conjunto E^k, com a atualização da LC efetuada em seguida (linha 16).

Após a definição dos L grupos iniciais E^k, formados por 2 elementos cada, os demais elementos pertencentes à LC são alocados aos L grupos também utilizando uma LRC (linhas 18 até 47). Em específico, entre as linhas 19 e 34, são selecionadas, da LC, n_a amostras, cada uma com $2L$ índices, correspondentes às possíveis UPAs que podem ser alocadas a cada um dos L estratos. Em 23, define-se uma matriz P, relacionada ao produto cartesiano entre os elementos de cada amostra e os elementos de cada um dos L grupos E^k, tomando-se as somas entre os elementos (UPAs) que estão alocados aos grupos (estratos) e novos elementos que são candidatos à alocação.

Dando continuidade ao processo de construção, produz-se, para cada linha de P, uma matriz $M_{L\times L}$, que contém as somas das distâncias entre cada um dos L grupos e os L pares de elementos (UPAs) candidatos à inserção. Assim, para determinar a qual grupo (estrato) cada uma das duplas de elementos pode ser alocada, resolve-se

um problema de designação (linha 31), aplicando-se o Algoritmo Húngaro (CORMEN *et al*., 2022).

Figura 1 - Procedimento de Construção

Algoritmo 1 - Procedimento de Construção

1: Defina $LC = \{1, \ldots n\}$
2: Defina $E^k = \emptyset \ , \ \forall k \in \{1, \ldots, L\}$
3: Calcule D - matriz com todas as entradas $d_{ij} \ \ i = 1, \ldots, n \ ; \ j = 1, \ldots, n$
4: $k \leftarrow 0$
5: **Enquanto** $(k < L)$ **faça**
6: $k \leftarrow k + 1$
7: Selecione, aleatoriamente de LC, n_a duplas de índices (i, j) definindo a matriz $A_{n_a \times 2}$
8: **Para** $s = 1, \ldots, n_a$ **faça**
9: $g[s] \leftarrow d_{ij}$ (tomando os valores i e j em $A[s,]$)
10: **fim Para**
11: $g_{min} \leftarrow min_{\forall s \in \{1, \ldots, n_a\}}\{g\}$
12: $g_{max} \leftarrow max_{\forall s \in \{1, \ldots, n_a\}}\{g\}$
13: $LCR = \{t \in LC | g(t) \leq g_{min} + \alpha(g_{max} - g_{min})\}$
14: Selecione um elemento $r \in LCR$
15: $E^k \leftarrow (A[r, 1], A[r, 2])$
16: $LC \leftarrow LC \backslash \{A[r, 1], A[r, 2]\}$
17: **fim Enquanto**
18: **Enquanto** $(|LC| \geq 2L)$ and $(mincap < cap)$ **faça**
19: Selecione aleatoriamente (c/ reposição) de LC n_a amostras de $2L$ índices (i, j) definindo a matriz $A_{n_a \times 2L}$
20: **Para** $s = 1, \ldots, n_a$ **faça**
21: **Para** $e = 1, \ldots, L$ **faça**
22: **Para** $k = 1, \ldots, L$ **faça**
23: Defina $P_{w \times 2}$ (matriz) como produto cartesiano dos elementos de $A[s, (2e - 1) : (2e)]$ com os elementos de E^k
24: $soma \leftarrow 0$
25: **Para** $z = 1, \ldots, w$ **faça**
26: $soma \leftarrow soma + d_{ij} \ \ ((i, j) \in P[z,])$
27: **fim Para**
28: $M[e, k] \leftarrow soma + d_{qh} \ \ ((q, h) \in A[s, (2e - 1) : (2e)])$
29: **fim Para**
30: **fim Para**
31: Aplique Algoritmo Húngaro em M e obtenha as L designações entre os L grupos E^k e as L duplas de elementos de $A[s,]$
32: $c_d[s] \leftarrow custo.total.desig$
33: Armazene em $Aloc[s, 1 : L]$ as L designações definidas em (31)
34: **fim Para**

Fonte: Autoria própria.

Nas linhas 32 e 33 são armazenados os custos e as designações ótimas (aos grupos) produzidas por este algoritmo. Nas linhas 35 até 38, considerando os custos armazenados, define-se a LRC que contém os índices associados às alocações que produzem os menores acréscimos, avaliando-se a soma de distâncias entre os elementos candidatos à inclusão (UPAs) e os seus respectivos grupos. Em seguida, entre as linhas 41 e 46 são atualizados os *L* grupos, seguida pela atualização da LC e de *mincap*. A aplicação deste procedimento construtivo - entre as linhas 1 e 47 -, garante o cumprimento imediato da restrição relativa ao mínimo 150 UPAs alocadas por estrato. Por fim, caso |LC| > 0, ou seja, ainda existem UPAs para alocação aos

estratos, as linhas de 48 até 63 são executadas. Nesta parte, em cada iteração, o elemento (UPA) na LC que produz o menor acréscimo, considerando a sua distância (parcela da variância) em relação aos *L* estratos, é adicionado ao estrato e a LC é atualizada. A solução s_o (conjuntos E^k, *k=1, ..., L*) produzida por este procedimento é utilizada como solução inicial para aplicação dos dois procedimentos de busca local que serão descritos na subseção 3.1.2.

Figura 2 - Procedimento de Construção (continuação)

Algoritmo 1 - Procedimento de Construção (continuação)

35: $g_{min} \leftarrow min_{\forall s \in \{1,\ldots,n_a\}}\{c_d\}$
36: $g_{max} \leftarrow max_{\forall s \in \{1,\ldots,n_a\}}\{c_d\}$
37: $LCR = \{t \in LC | g(t) \leq g_{min} + \alpha(g_{max} - g_{min})\} \quad \alpha \in [0,1]$
38: Selecione um elemento $r \in LCR$
39: $a \leftarrow Aloc\,[r, 1:L]$
40: $ne \leftarrow A\,[r,]$
41: **Para** $k = 1, \ldots, L$ **faça**
42: $j \leftarrow a\,[k]$
43: $E^j \leftarrow E^j \bigcup ne[(2k-1):(2k)]$
44: **fim Para**
45: $LC \leftarrow LC \backslash \{ne\}$
46: $mincap \leftarrow mincap + 2L$
47: **fim Enquanto**
48: **Enquanto** $|LC| > 0$ **faça**
49: $d_{min} \leftarrow \infty$
50: **Para** $k = 1, \ldots, L$ **faça**
51: Defina $Q_{w \times 2}$ (matriz) como produto cartesiano entre os elementos da LC e E^k
52: **Para** $s = 1, \ldots w$ **faça**
53: $i \leftarrow Q\,[s, 1]$ (Elemento da LC)
54: $j \leftarrow Q\,[s, 2]$ (Elemento de E^k)
55: **Se** $d_{ij} < d_{min}$ **Então**
56: $d_{min} \leftarrow d_{ij}$
57: $t \leftarrow Q\,[s, 1]$
58: $g_{min} \leftarrow k$
59: **fim Se**
60: **fim Para**
61: **fim Para**
62: $E^{g_{min}} \leftarrow E^{g_{min}} \bigcup t$
63: $LC \leftarrow LC - \{t\}$
64: **fim Enquanto**
65: Retorne $(E^1, E^2, \cdots, E^L)$

Fonte: Autoria própria.

3.1.2 Procedimento de Busca Local (Refinamento)

De forma a aprimorar as soluções produzidas na fase de construção do GRASP e garantir, consequentemente, a produção de estratos mais homogêneos quanto à

expressão de variância - função objetivo do PEUPA -, foram propostos dois procedimentos de busca local, que são aplicados em sequência e preservam a restrição da capacidade.

A Buscal Local 1 (BL1) avalia, para cada um dos *n* elementos distribuídos nos *L* grupos E^k, se a sua realocação a outro grupo (receptor) simultaneamente reduz o valor da função objetivo e preserva a viabilidade da solução. Ou seja, o elemento *i* (UPA), avaliado na iteração atual, será realocado do seu atual grupo E^k (doador) a um dos *L−1* grupos (receptores), caso $|E^k| > 150$. Ainda neste sentido, escolhe-se como grupo receptor $E^g(q \in \{1, ..., L\}$ tq $g \neq k)$ aquele em que a adição de *i* e sua respectiva exclusão de E^k produzir a maior redução na variância, ou seja, tem por base a estratégia *best improvement* (Gendreau; Potvin, 2010). Já na Busca Local 2 (BL2) - aplicada após a BL1 -, são efetuadas, essencialmente, tentativas de trocas entre os elementos de dois grupos (sem alterar a sua cardinalidade), avaliando, em cada tentativa, se há redução no valor da variância.

O pseudocódigo relacionado à BL1 é apresentado na Figura 3. Nas linhas de 1 e 2 define-se o vetor t_g relativo aos totais de elementos em cada um dos grupos e inicializa-se a variável de controle "melhorou" com valor *True*. Essencialmente, em cada iteração completa - linhas 5 até 28 -, onde são avaliados os elementos (UPAs) que estão alocados a grupos com mais de 150 elementos, são avaliadas realocações dos objetos a outros grupos (linhas 12-19). Se, ao final desta iteração completa, pelo menos um dos objetos realocados produzir redução na função objetivo, a variável "melhorou" receber valor *True*, e uma nova iteração completa é iniciada, onde novamente serão avaliados todos os objetos. A BL1 é aplicada enquanto ocorrer redução na variância.

Figura 3 - Procedimento de Busca Local 1 (BL1)

Algoritmo 2 - Procedimento de Busca Local 1 - BL1

1: Defina t_g como o vetor com o total de elementos em cada um dos L grupos E^k
2: $melhorou \leftarrow True$
3: **Enquanto** (melhorou=True) **faça**
4: $melhorou \leftarrow False$
5: **Para** (i=1,..., n) **faça**
6: Atribua a j o índice do grupo ao qual o i-ésimo objeto está alocado
7: **Se** $(t_g[j] > capacidade)$ **Então**
8: $dj \leftarrow \sum_{\forall r \in E^j} d_{ir}$
9: $g \leftarrow \{1, \ldots, L\} - \{j\}$
10: $minimo \leftarrow +\infty$
11: $reducao \leftarrow False$
12: **Para** $(w \in g)$ **faça**
13: $dw \leftarrow \sum_{\forall r \in E^w} d_{ir}$
14: $dif \leftarrow dw - dj$
15: **Se** $(dif < 0) and (dif < minimo)$ **Então**
16: $minimo \leftarrow dif$
17: $qmin \leftarrow w$
18: $melhorou \leftarrow True$
19: $reducao \leftarrow True$
20: **fim Se**
21: **fim Para**
22: **Se** (reducao=True) **Então**
23: $E^j \leftarrow E^j - \{i\}$
24: $E^{qmin} \leftarrow E^{qmin} \bigcup i$
25: Atualize o vetor t_g
26: **fim Se**
27: **fim Se**
28: **fim Para**
29: **fim Enquanto**
30: Retorne $(E^1, E^2, \cdots, E^L)$

Fonte: Autoria própria.

O pseudocódigo relacionado à BL2 é apresentado na Figura 4. Durante certo número de iterações limitado pelo total de melhorias (reduções na função objetivo), são selecionados dois grupos E^i e E^j dentre os L disponíveis (linha 6), sendo selecionados, em seguida, n_e elementos de cada um deles (linhas 7-8). Em um passo posterior, constrói-se uma matriz P_w (linha 9), que contém o produto cartesiano entre os elementos desses dois grupos. Cada linha de P tem duplas compostas por um elemento de E^i e um elemento de E^j, sendo esses elementos trocados entre estes dois grupos, e avaliando (linhas 11-15) se tal operação produz redução no valor da *fobj*. A

dupla de elementos (de *P*) que produzir a maior redução é trocada entre os grupos E^i e E^j (linhas 17-21) e o contador de melhorias é atualizado. O processo de busca termina quando o total de melhorias produzidas for alcançado *MaxMelhorias* ou se não ocorrer redução no valor da variância a partir das *w* trocas efetuadas, durante o número de iterações igual a *Maxtentativas*.

Figura 4 - Procedimento de Busca Local 2 (BL2)

Algoritmo 3 - Procedimento de Busca Local 2 - BL2

1: Defina D - matriz com todas as entradas d_{ij} $i = 1, \ldots, n$; $j = 1, \ldots, n$
2: Defina t_g o vetor com o total de elementos em cada um dos L grupos E^k
3: $melhorias \leftarrow 0$
4: $tentativas\_sem\_melhoria \leftarrow 0$
5: **Enquanto** (($melhorias <$ MaxMelhorias) and ($tentativas\_sem\_melhoria <$ Maxtentativas)) **faça**
6: Selecione aleatoriamente dois valores $i, j \in \{1, \ldots, L\}$
7: $E_1 \leftarrow sample(E^i, ne)$ (selecione de E^i uma amostra aleatória de n_e elementos)
8: $E_2 \leftarrow sample(E^j, ne)$ (selecione de E^j uma amostra aleatória de n_e elementos)
9: Defina a matriz P_w como produto cartesiano entre os elementos de E_1 e E_2 ($w = n_e^2$)
10: Defina d_f como vetor com w entradas
11: **Para** (i=1,..., w) **faça**
12: $r \leftarrow P[i, 1]$
13: $t \leftarrow P[i, 2]$
14: $d_f[i] \leftarrow \sum_{\forall s \in E_1} d_{st} - \sum_{\forall s \in E_1} d_{sr} + \sum_{\forall u \in E_2} d_{ur} - \sum_{\forall u \in E_2} d_{ut} - 2d_{rt}$
15: **fim Para**
16: $v \leftarrow \underset{i \in \{1,\ldots,w\}}{\text{argmin}}\ d_f[i]$
17: **Se** ($df[v] < 0$) **Então**
18: $E^i \leftarrow E^i - \{P[v, 1]\} \bigcup \{P[v, 2]\}$
19: $E^j \leftarrow E^j - \{P[v, 2]\} \bigcup \{P[v, 1]\}$
20: $melhorias \leftarrow melhorias + 1$
21: $tentativas\_sem\_melhoria \leftarrow 0$
22: **Senão**
23: $tentativas\_sem\_melhoria \leftarrow tentativas\_sem\_melhoria + 1$
24: **fim Se**
25: **fim Enquanto**
26: Retorne E

Fonte: Autoria própria.

3.2. Heurística da Envoltória Convexa (HEC)

A HEC (GUIGNARD; AHLATCIOGLU, 2021) é aplicada em problemas definidos por: (i) uma função *f(x)* não linear; (ii) restrições lineares e (iii) com variáveis 0-1 (binárias), conforme o problema P1 a seguir. A HEC também é definida como uma

matheurística[4], tendo em vista que é baseada em um Algoritmo de Programação Matemática denominado Decomposição Simplicial (DS) (Von Hohenbalken, 1977), que corresponde a uma generalização do Algoritmo de Frank-Wolfe (FRANK; WOLFE, 1956), especialmente adaptada para problemas de grande porte e com variáveis inteiras. Ainda neste sentido, ao aplicar a HEC, o objetivo é produzir, rapidamente, soluções viáveis para P1 de boa qualidade, particularmente, para problemas convexos.

$$(P1)\ \text{Minimizar} f(x) \tag{12}$$

$$Ax \leq b \tag{13}$$

$$x \in B = \{0,1\}^n \tag{14}$$

Basicamente, em cada iteração da HEC, resolve-se um problema P2 (a seguir), com uma função objetivo que corresponde a uma linearização da função de P1, combinada com as restrições de P1, seguida pela resolução de um problema de programação não-linear, definido pela função objetivo de P1 e uma única restrição linear.

A fim de explicar as etapas da aplicação da HEC, considere o problema P1 e adicionalmente, seja *H* a sua envoltória convexa que contempla as soluções inteiras viáveis.

$$H = \text{Conv}\{x \mid Ax \leq b, x \in B\} \tag{15}$$

A meta é encontrar um vértice de *H* com melhor valor de função objetivo. Para isso, pode-se utilizar o Método da Decomposição Simplicial (Von Hohenbalken, 1977) múltiplas vezes, dentro da execução da HEC, com um diferente ponto inicial em cada vez. Inicialmente, define-se a Relaxação da Envoltória Convexa (REC) de P1 como o Problema de Minimizar *f(x)* sobre a Envoltória Convexa *H* das soluções inteiras viáveis de P1. Ainda neste sentido, considere o subproblema relacionado à envoltória convexa na *k*-iteração, onde o objetivo é minimizar, sobre *H*, a função objetivo de P1

[4] Consiste em uma abordagem híbrida que combina métodos matemáticos exatos com técnicas heurísticas ou metaheurísticas para resolver problemas complexos de otimização.

linearizada em um ponto viável x^k, mais especificamente, temos problema de minimizar *L(x)* sobre *x* em *H* tal que:

$$L(x) = f(x^k) + \nabla f(x^k) \cdot (x - x^k) \tag{16}$$

Neste ponto, temos o seguinte problema de Programação Linear Inteira associado:

$$(P2) \text{ Minimizar } L(x) \tag{17}$$

$$Ax \leq b \tag{18}$$

$$x \in B = \{0,1\}^n \tag{19}$$

Considere que y^k é a solução ótima de P2 e que depende de x^k, uma vez que *L(x)* depende do ponto de linearização x^k adotado. A cada iteração, resolve-se um problema P2 com uma função objetivo linearizada diferente *L(x)*, enquanto o conjunto de restrições definido por (18)-(19) permanece inalterado.

Ou seja, a solução para P2, denotada por y^k, corresponde a um vértice de sua envoltória convexa, sendo usado para expandir a área de busca na próxima iteração. Mais especificamente, em cada iteração da HEC, é produzido um novo vértice, que é adicionado à combinação convexa constituída por $\{x^0, y^1, y^2, \ldots, y^{k-1}\}$ nas iterações anteriores. A ideia é construir um novo ponto de linearização x^k para a próxima iteração, melhorando a aproximação do mínimo de *f(x)* - função do problema original P1.

A seguir são listados os passos gerais de aplicação da HEC, que são repetidos até que $f(x^k) = f(x^{k-1})$, ou seja, mesmo adicionando um novo vértice y^k ao conjunto e resolvendo um Problema de Programação Não Linear (PPNL) (passo 3) sobre o novo fecho convexo, o mínimo de *f(x)* permanece o mesmo.

Passo 1: Considere os vértices obtidos até a iteração *k-1*:

$$\{v^1, v^2, \ldots, v^k\} = \{x^{(0)}, y^{(1)}, y^{(2)}, \ldots, y^{(k-1)}\}$$

Passo 2: Forme a combinação convexa desses vértices:

$$x = \sum_{i=1}^{k} \lambda_i v^i, \text{ com } \sum_{i=1}^{k} \lambda_i = 1, \lambda_i \geq 0$$

Passo 3: Resolva o PPNL abaixo utilizando a função objetivo de P1

$$\min_{\lambda} \left\{ f\left(\sum_{i=1}^{k} \lambda_i v^i \right) \right\}$$

sujeito a:

$$\sum_{i=1}^{k} \lambda_i = 1, \lambda_i \geq 0 \; \forall i \in \{1, \cdots, k\}$$

Passo 4: Obtenha o novo ponto de linearização $x^{(k)} = \sum_{i=1}^{k} \lambda_i^* v^i$, sendo a λ^* um vetor correspondente à solução ótima do PPNL. Este ponto será usado para: (i) Linearizar *f(x)* novamente, produzindo *L(x)* e (ii) Resolver P2 e gerar novo vértice $y^{(k)}$.

De acordo com Guignard e Ahlatcioglu (2021), é conveniente/recomendado começar a solução do primeiro problema P2 a partir de uma solução inteira viável x^0, que faz parte da família de vértices para execução do algoritmo DS. Ou seja, este ponto inicial é uma solução inteira viável do problema original, que satisfaz (13) e (14).

Como o ponto inicial escolhido tem impacto na solução final obtida, uma possível estratégia consiste em resolver diversos Problemas de Programação Linear Inteira do tipo P3, com diferentes funções objetivo lineares, mantendo as restrições de P1.Dentro desta perspectiva, o vetor *c* é alterado em cada execução - com diferentes valores -, forçando o solver a explorar diferentes regiões do espaço viável, produzindo, consequentemente, diferentes pontos iniciais x^0 que serão utilizados na primeira iteração de cada problema P2.

(P3) Minimizar $c^T x$ (20)

$$Ax \leq b \quad (21)$$

$$x \in B = \{0,1\}^n \quad (22)$$

Considerando a estrutura de P1, observa-se que a formulação F1 (seção 2) pode ser adaptada e resolvida mediante aplicação da HEC. Ou seja, no caso do PEUPA, *L(x)* corresponde a uma linearização da função objetivo apresentada na formulação F1 (equação 23), e as restrições consideradas em P2 e P3 correspondem àquelas também definidas em F1. Em relação aos coeficientes da função objetivo de P3, foram gerados, para a sua resolução, $n \times L$ valores segundo uma distribuição uniforme em [0,1].

$$f(x) = \sum_{h=1}^{L} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i^h x_j^h \tag{23}$$

3.3. Otimizador de Chaves Aleatórias (OCA)

Nesta seção é apresentada uma descrição geral do OCA proposto em Chaves *et al*. (2024), que consiste em um método de busca local estocástico, tendo por base o conceito de chaves aleatórias, que compõem um conjunto de vetores de números reais no intervalo [0, 1).

Esses vetores, por sua vez, são decodificados em soluções viáveis para cada problema de otimização em questão, utilizando decodificador desenvolvido especificamente para o problema. O otimizador é modular e permite a integração de diversas metaheurísticas clássicas, em particular, aquelas que foram consideradas neste trabalho, sejam elas: BRKGA, ILS, LNS e VNS. Ele agrega um conjunto de procedimentos de uso geral, no sentido que são utilizados em alguns dos passos de cada uma das metaheurísticas, uma vez que estes operam, diretamente, no vetor de chaves aleatórias.

Como todas as soluções são representadas na forma de vetores de chaves aleatórias, independentemente do problema ou do algoritmo, qualquer procedimento que trabalhe sobre esses vetores pode ser utilizado em qualquer metaheurística. Na Tabela 2 são listados os procedimentos disponíveis no OCA. Maiores detalhes sobre estes procedimentos podem ser encontrados nos algoritmos de 1 até 7, apresentados no trabalho de Chaves *et al*. (2024). Adicionalmente, a descrição dessas quatro

metaheurísticas e seus os pseudocódigos podem ser encontrados, respectivamente, na seção 4.5 e no apêndice do referido trabalho (algoritmos 8, 13, 14 e 16).

Tabela 2 - Procedimentos principais do OCA

Componente	Descrição	Função Principal
Perturbação	Modifica uma solução aplicando movimentos como *Swap, Swap Neighbor, Mirror e Random*.	Diversificação
Mistura	Combinação de dois vetores de soluções, podendo usar complementos e introduzir ruído aleatório.	Exploração
Busca Local	Conjunto de heurísticas que refinam soluções - todas combinadas no RVND, que consiste em uma variante do VNS. Inclui: *Swap, Mirror, Farey e Nelder-Mead*.	Intensificação

Fonte: Autoria própria.

Considerando que estes procedimentos, conjuntamente com as metaheurísticas, podem ser aplicados em qualquer problema de otimização, para a sua implementação na resolução do PEUPA, foi necessário, basicamente, definir um procedimento decodificador e um procedimento que efetua o cálculo da função objetivo definida nas equações de (1)-(3), conforme descrito na subseção seguinte.

3.3.1. Representação da Solução, Decodificador e Função Objetivo

No caso do PEUPA, cada vetor *u* de chaves aleatórias foi definido por *n* entradas relacionadas ao total de UPAs que devem ser alocadas aos *L* estratos, sendo o procedimento decodificador definido pelos seguintes passos:

1. Definir um vetor a com n posições preenchidas, inicialmente, com zero.
2. Dividir o intervalo $[0,1]$ em L subintervalos $I_k, k = 1, \dots, L$, sendo L igual ao número de estratos estatísticos aos quais as UPAs serão alocadas. Dessa forma, temos: $I_1 = \left[0, \frac{1}{L}\right)$, $I_2 = \left[\frac{1}{L}, \frac{2}{L}\right), \dots, I_k = \left[\frac{k-1}{L}, \frac{k}{L}\right), \dots, I_L = \left[\frac{L-1}{L}, 1\right)$.
3. Preencher um vetor u com *n* valores gerados segundo $U([0,1])$.
4. Tomar cada entrada $u_i \in u$ e atribuir à entrada a_i correspondente um valor $k \in \{1, \cdots, L\}$ igual ao número do intervalo $I_k = \left[\frac{k-1}{L}, \frac{k}{L}\right)$ tal que $u_i \in I_k$.

Após a aplicação deste decodificador, cada entrada a_i do vetor a conterá o número do estrato ao qual a i-ésima UPA será alocada. Assim, o exemplo a seguir ilustra a aplicação desse decodificador, supondo $L = 3, n = 10$ e $u = \{0{,}314, 0{,}634, 0{,}826, 0{,}118, 0{,}217, 0{,}300, 0{,}691, 0{,}041, 0{,}825, 0{,}838\}$. Temos três subintervalos definidos por: $I_1 = \left[0, \frac{1}{3}\right), I_2 = \left[\frac{1}{3}, \frac{2}{3}\right), I_3 = \left[\frac{2}{3}, 1\right)$. Tomando cada valor de u e verificando em qual dos subintervalos está, produz-se $a = \{1,2,3,1,1,1,3,1,3,3\}$.

Cabe salientar que todos os procedimentos reportados na Tabela 2 - componentes de todas as metaheurísticas consideradas neste trabalho -, são aplicados aos vetores de chaves aleatórias *u*. Ainda neste sentido, considerando aplicação de tais procedimentos nos vetores *u*, seguida pela decodificação, não é possível garantir, sempre, a produção de vetores solução *a* que sejam viáveis para o PEUPA. Mais especificamente, considerando a restrição de capacidade relacionada ao número mínimo de 150 UPAs por estrato, podem ser produzidos vetores *a* que definam estratos de UPAs que não satisfaçam esta restrição. Ou seja, durante a aplicação dos algoritmos relacionados às metaheurísticas, podem ser geradas algumas soluções inviáveis.

De forma a contornar este problema, seguindo o que está preconizado na literatura, uma possível alternativa diz respeito ao uso de métodos de penalidade, ou seja, utilizar uma função para penalizar soluções que violem alguma restrição. Em linhas gerais, um método de penalidade transforma um problema restrito em um problema irrestrito (ou reduz seu número de restrições), mediante a introdução de um termo que é adicionado à função objetivo do problema original, penalizando qualquer violação de restrição - que foram desconsideradas. Em Jiao *et al*. (2013), Gaspar-Cunha *et al*. (2016) e Garcia *et al*.(2023) podem ser encontrados aplicações e métodos (incluindo penalidades) para tratar problemas de otimização com restrições.

Neste trabalho, após avaliar/estudar algumas das funções de penalidade propostas na literatura, considerando a função objetivo do PEUPA (24) e o vetor *a* produzido a partir da aplicação do procedimento decodificador, foi proposta a função de penalidade V_p definida em (25)-(26). Nesta função, *cap* é igual ao número mínimo de UPAs por estrato (150) e n_h é igual ao total de UPAs alocadas ao *h*-ésimo estrato.

$$V(t_x) = \sum_{h=1}^{L} \sum_{\forall a_i, a_j \in E_h} d_{ij} \quad (24)$$

$$V_p = V(t_x) + V(t_x) \times \frac{\max\limits_{h=1,\dots,L} \{df_h\}}{cap} \quad (25)$$

$$df_h = \max(cap - n_h, 0), h = 1, \dots, L \quad (26)$$

4 RESULTADOS

Esta seção traz resultados e análises relativos a um conjunto de experimentos computacionais realizados com o algoritmo GRASP, a Heurística da Envoltória Convexa (HEC), os quatro algoritmos heurísticos baseados nas metaheurísticas BRKGA, ILS, LNS e VNS, além das formulações F1 e F2 apresentadas na seção 2.

Os algoritmos e as formulações foram implementados em linguagem R, utilizando o software RStudio - versão 2025.05.0+496. Em particular, para a resolução das duas formulações5 e dos problemas P2 e P3, relativos à aplicação da HEC, foi utilizado o solver GUROBI (versão 12.0), disponível no pacote gurobi do R. Adicionalmente, para a resolução do PPNL - associado ao passo 3 da HEC -, foi utilizado pacote alabama do R (função auglag6). Todos os experimentos foram realizados em um computador dotado de processador I9-13900F (com 24 núcleos), 128 GB de RAM e sistema operacional Windows 11.

4.1. Instâncias Utilizadas

Para a realização dos experimentos com os algoritmos e formulações, foram utilizadas 30 instâncias de teste construídas a partir dos dados do censo demográfico de 2022[7] (disponíveis para uso público), armazenados em arquivos com informações dos domicílios e de renda do responsável pelo domicílio - disponíveis por setor censitário.

[5] Na formulação F1 foram utilizados os métodos aplicados a problemas de programação quadrática e na formulação F2 foram utilizados métodos de programação inteira.
[6] Função que implementa o algoritmo do Lagrangiano Aumentado.
[7] Dados em: http://www.ibge.gov.br/estatisticas/sociais/populacao/22827-censo-demografico-2022.html?=&t=downloads

Cada instância corresponde a um arquivo associado a um município Brasileiro[8], contendo, por setor censitário - aqui considerado como uma UPA -, as variáveis: Domicílios Particulares Permanentes Ocupados e Valor do rendimento nominal médio mensal das pessoas responsáveis por domicílios particulares permanentes ocupados. Essas variáveis estão relacionadas, respectivamente, aos termos N_i e Y_i que fazem parte da expressão de variância, definida na seção 2 deste artigo.

A Tabela 3 a seguir traz informações sobre as 30 instâncias utilizadas nos experimentos. A coluna **Instância** traz a idenfiticação da instância, em específico, os dois primeiros caracteres correspondem às UFs e os 7 caracteres após "_" indicam o código do município associado. A coluna **NUPA** diz respeito ao número de UPAs da instância (setores) e a coluna **L** indica o número de estratos considerados para alocação das UPAs, a partir da aplicação dos algoritmos e das formulações. As instâncias assinaladas com '*' foram utilizadas em um experimento - reportado na subseção 4.2.2 - , para avaliar a estabilidade do algoritmo GRASP.

Tabela 3 - informações sobre as 30 instâncias utilizadas

Instância	NUPA	L	Arquivo	NUPA	L
SC_4208203	400	2	AC_1200401*	662	4
SC_4216602	413	2	CE_2303709	717	4
AL_2700300*	421	2	PA_1500800	738	4
RN_2408003	430	2	PE_2604106*	794	5
RS_4316907	441	2	MG_3170107	842	5
MA_2105302	453	3	RJ_3300456	980	5
RS_4309209	464	3	SC_4209102	1044	5
RR_1400100	506	3	RN_2408102	1076	5
RJ_3300407	526	3	RJ_3301009*	1307	5
MT_5107602*	565	3	SP_3547809	1637	5
MA_2111201	589	3	PA_1501402	2031	5
SP_3552809	591	3	GO_5208707	2289	5

[8] Para fins de experimentos, foram inicialmente gerados os arquivos para municípios que tinham, pelo menos, 400 (UPAs), totalizando 152 arquivos. Desses, foram selecionados 30 arquivos utilizados como instâncias de teste; o que corresponde a cerca de 20% do total.

SP_3552502	643	4	SP_3509502	2502	5
BA_2933307	659	4	PE_2611606	2796	5
TO_1721000	660	4	PR_4106902	3168	5

Fonte: Autoria própria.

4.2. Experimentos Computacionais

Objetivando produzir uma análise mais robusta dos resultados, que contemple a avaliação do valor mínimo da Fobj do PEUPA, até a aplicação de Testes de Hipóteses, cada um dos seis algoritmos - GRASP, HEC e metaheurísticas -, foi aplicado 20 vezes em cada uma das 30 instâncias, sendo armazenado o valor da Função Objetivo (variância) obtido em cada execução. A partir desses valores, foram calculados o menor valor da função objetivo (variância) e a mediana correspondente, permitindo realizar o conjunto de análises que serão apresentadas a seguir.

De forma a garantir uma comparação equitativa entre algoritmos GRASP, BRKGA, ILS, VNS e LNS, foi adotado o mesmo critério de parada, qual seja, uma abordagem escalonada do tempo máximo de processamento, que depende do tamanho do problema. Assim sendo, neste trabalho as instâncias foram agrupadas em três classes – consideradas: pequenas, médias e grandes - e para cada grupo foi atribuído um tempo máximo de processamento, conforme apresentado na Tabela 4.

Tabela 4 - Classes de instâncias e respectivos tempos máximos de execução

Classe	Faixa de NUPA	Instâncias	Tempo máximo (min.)
Pequenas	$400 - 550$	10	3
Médias	$551 - 1300$	11	6
Grandes	> 1300	9	10

Fonte: Autoria própria.

No caso da HEC, foram adotados dois critérios: (i) tempo máximo de 30 minutos e (ii) se não houver melhora no valor da função objetivo relacionado aos pontos x^k e x^{k+1} produzidos entre duas iterações seguidas -, o mesmo utilizado nos experimentos reportados em Guignard e Ahlatcioglu (2021). Por fim, no caso das formulações F1 e F2, foi estabelecido um tempo máximo de processamento de 60 minutos. Considerando esse limite de tempo, só foi possível obter uma solução viável para a

formulação F1 nas 22 primeiras instâncias. No caso da formulação F2, foi produzida uma solução para todas as 30 instâncias.

4.2.1. Calibração de parâmetros

Em relação ao GRASP, os valores de seus parâmetros foram definidos a partir de experimentos prévios de calibração - com o *irace* (LÓPEZ-IBÁÑEZ *et al*., 2016), disponível no pacote irace do R-, realizados considerando a aplicação do GRASP nas seis instâncias apresentadas na Tabela 5, também selecionadas do conjunto de 152 instâncias inicialmente construído.

O *irace*[9] (*Iterated Racing*) consiste em uma ferramenta que tem sido amplamente utilizada para calibração automática de parâmetros de algoritmos, sendo projetado para encontrar boas configurações de parâmetros, que possibilitem maximizar o desempenho de algoritmos de otimização, especialmente aqueles baseados em metaheurísticas.

Tabela 5 - Informações sobre as seis instâncias utilizadas no experimento de calibração

Instância	**NUPA**	**L**	**Inst**	**NUPA**	**L**
RS_4318705	427	2	PE_2609600	640	4
RJ_3302700	470	3	PR_4119905	669	4
PR_4104808	576	3	PB_2504009	784	5

Fonte: Autoria própria.

O algoritmo GRASP foi executado, no ambiente do irace[10], tomando, como ponto de partida, as 432 combinações (quíntuplas) dos parâmetros utilizados no procedimento de construção e na busca local 2. Os possíveis valores adotados para cada um desses parâmetros, utilizados na construção das combinações, são apresentados na Tabela 6.

[9] Utiliza técnica de "*racing*", que permite avaliar, progressivamente, configurações de parâmetros em uma série de instâncias de teste, eliminando dos testes aquelas que demonstram baixo desempenho, fato que implica economia de tempo computacional.

[10] Para a execução do irace o seui parâmetro *maxExperiments*, que regula o número máximo de experimentos (avaliações), foi definido em 3000.

Tabela 6 - GRASP – Parâmetros de Calibração

Parâmetro	Valores
α (construção)	$0.025, 0.05, 0.10$
n_a (construção)	$5, 10, 20$
MaxTentativas (BL2)	$10, 25, 50$
MaxMelhorias (BL2)	$50, 100, 200, 300$
n_e (BL2)	$10, 25, 50, 75$

Fonte: Autoria própria.

A partir da avaliação dos valores médios da função objetivo do problema (variância), obtidos a partir de várias execuções GRASP para as 6 instâncias, com diferentes combinações dos parâmetros acima, o irace retornou seguinte combinação vencedora: α =0,05, n_a=10, n_e=50, *MaxMelhorias*=200 e *MaxTentativas*=50.

Por fim, no caso dos algoritmos relacionados às metaheurísticas BRKGA, ILS, VNS e VNS, a Tabela 7 traz os valores adotados para os seus respectivos parâmetros de entrada e que tiveram, por base, o que está proposto em Chaves *et al*. (2024). Diferentemente do GRASP, cujos parâmetros foram calibrados especificamente para o PEUPA por meio do irace, os algoritmos BRKGA, ILS, VNS e LNS utilizaram os valores de parâmetros originalmente propostos em Chaves *et al*. (2024). Essa decisão foi motivada pelo fato dessas metaheurísticas compartilharem, no contexto do OCA, os mesmos componentes (perturbação, mistura e busca local - apresentados na Tabela 2), já previamente avaliados pelos autores em experimentos realizados com diferentes problemas de otimização combinatória. Dessa forma, optou-se por preservar as configurações recomendadas na literatura, garantindo consistência metodológica entre os algoritmos derivados do framework.

Tabela 7 - Parâmetros utilizados nos quatro algoritmos relacionados às metaheurísticas

Parâmetro	Definição	BRKGA	ILS	VNS	LNS
p	Tamanho da população	1200	-	-	-
p_e	Conjunto de elite	$0{,}10p$	-	-	-
p_m	Mutantes	$0{,}20p$	-	-	-
ρ	Probabilidade de cruzamento	0,80	-	-	-
β_{min}	Taxa mínima de perturbação	-	0,05	0,10	0,10
β_{max}	Taxa máxima de perturbação	-	0,15	0,50	0,20

T_0	Temperatura Inicial	-	-	-	1000
α_r	Taxa de resfriamento	-	-	-	0,90
$k_{\max}$	Número de vizinhanças	-	-	10	-

Nota: Os parâmetros T_0e α_r são utilizados exclusivamente no algoritmo LNS, devido à adoção de um mecanismo de aceitação inspirado em Simulated Annealing.
Fonte: Autoria própria.

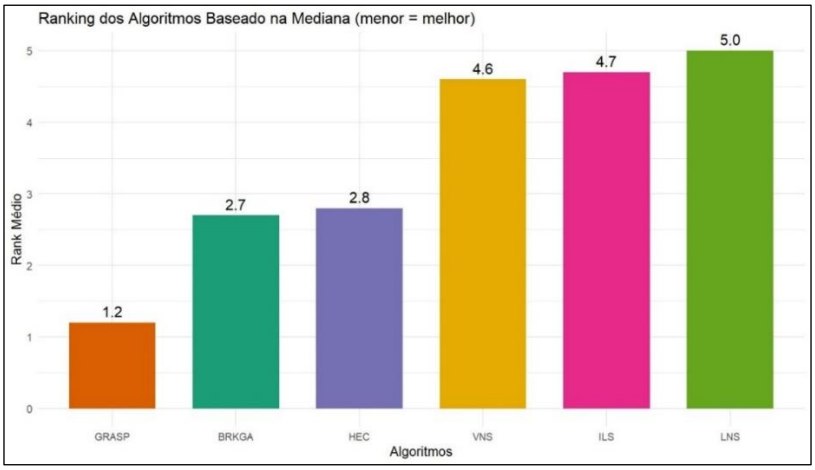
4.2.2. Resultados e Análises

Considerando os valores medianos da função objetivo, obtidos a partir das 20 execuções realizadas com os algoritmos GRASP, HEC, BRKGA, ILS, VNS e LNS executados com as 30 instâncias, foi aplicado o Teste de Friedman (Hollander *et al*., 2014). Este teste - não paramétrico - permite comparar um conjunto de algoritmos, considerando seus rankings[11]. Após a sua aplicação, foi obtido um *p*-valor inferior $2{,}2 \times 10^{-16}$; fato que corrobora com a existência de diferenças estatisticamente significativas entre os algoritmos. Todavia, este teste só indica que há alguma diferença entre os algoritmos - mas não diz entre quais.

Assim sendo, de forma a complementar esta etapa de avaliação, foi aplicado o Teste de Wilcoxon (HOLLANDER *et al*., 2014) e (GIBBONS; CHAKRABORTI, 2020) pareado com *Correção de Bonferroni*, que permite fazer comparações, par a par (total de 15), entre os seis algoritmos, para descobrir quais pares diferem significativamente. Em particular, o GRASP apresentou desempenho significativamente melhor (valores menores da função objetivo) do que os demais algoritmos e o BRKGA também teve desempenho significativamente melhor do que os algoritmos ILS, VNS, LNS. Resumindo a análise, a Figura 5 traz os ranks calculados para cada um dos algoritmos. Quanto o menor o valor dessa medida, melhor a sua classificação/performance em relação aos demais algoritmos. O algoritmo GRASP ocupa a primeira posição, seguido pelo BRKGA e a HEC que apresentam resultados semelhantes.

[11] O cálculo do rank de cada algoritmo foi realizado a partir da análise de seu desempenho observado em cada instância, de forma individual. Isto é, para cada instância, ordenamos os algoritmos do melhor para o pior, tendo, por base, o valor da mediana da função objetivo (variância) obtida nas 20 execuções.

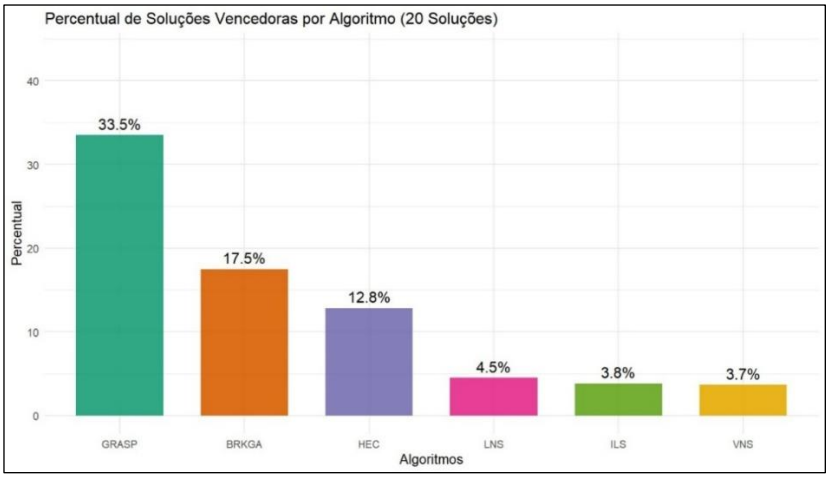
Figura 5 - Análise dos Ranks dos Algoritmos (menor valor indica melhor resultado)



Fonte: Autoria própria.

Dando sequência às análises, a Figura 6 traz os percentuais relacionados aos totais de soluções vencedoras de cada um dos algoritmos, isto é, o número de vezes que cada algoritmo produziu o menor valor função de objetivo nas 20 execuções (comparado ao menor valor obtido em relação a todos os algoritmos e todas as 20 execuções), para cada uma das 30 instâncias (máximo de 600). A partir desta figura, observa-se razoável vantagem do GRASP em relação aos demais algoritmos, com percentual da ordem de 34% de soluções vencendoras, seguido pelos algoritmos BRKGA e HEC, respectivamente, com percentuais da ordem 18% e 13%. Os algoritmos LNS, ILS e VNS tiveram desempenho inferior aos demais, com percentuais abaixo de 5%.

Figura 6 - Soluções vencedoras

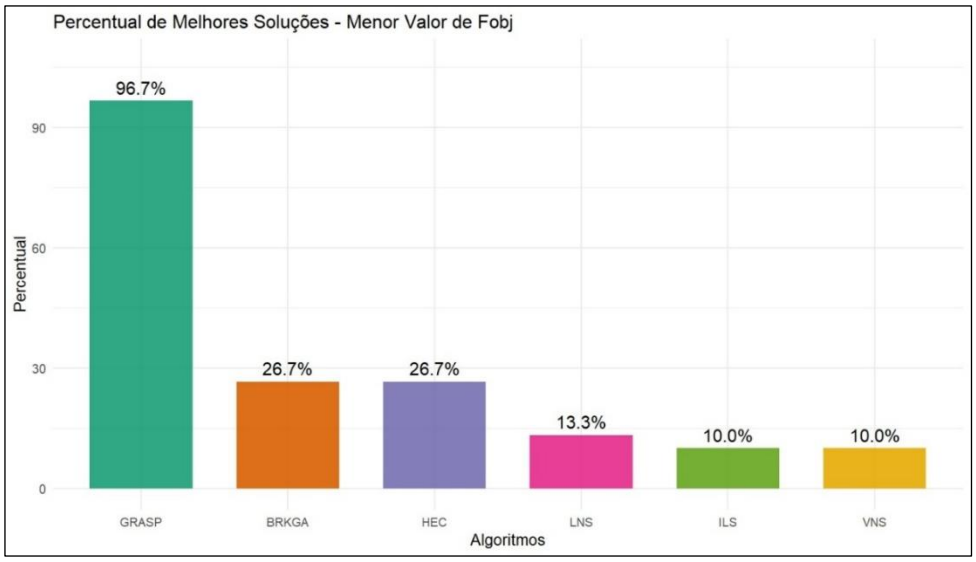


Fonte: Autoria própria.

A Figura 7 traz uma análise similar àquela proposta na Figura 6, considerando, particularmente, a comparação da melhor solução produzida por cada algoritmo nas - 20 execuções - para cada uma das instâncias, com a melhor solução obtida por instância. Novamente, observa-se performance superior do GRASP frentes aos demais algoritmos, produzindo cerca de 97% de soluções melhores, seguido novamente pelos algoritmos BRKGA e HEC, com cerca de 1/3 de melhores soluções em relaçao ao GRASP.

O gráfico da Figura 8 traz os percentuais de melhores soluções produzidas pelos algoritmos e pelas formulações F1 e F2, observando que só foi possível obter soluções para a formulação F2, dentro do tempo de 60 minutos, para as 22 primeiras instâncias. Assim sendo, para fins de comparação entre as formulações e os algoritmos, foi tomado o menor conjunto - 22 instâncias. Neste cenário de avaliação, observou-se, novamente, superioridade do GRASP em relação às demais abordagens. O algoritmo BRKGA, a HEC e a formulação F1 ficaram empatados em 2º lugar, com cerca de 37% das melhores soluções, seguidas pela metaheurística LNS com percentual da ordem de 18%. A formulação F2 teve a pior performance, com percentual da ordem de 9%.

Figura 7 - Melhores Soluções - Algoritmos



Fonte: Autoria própria.

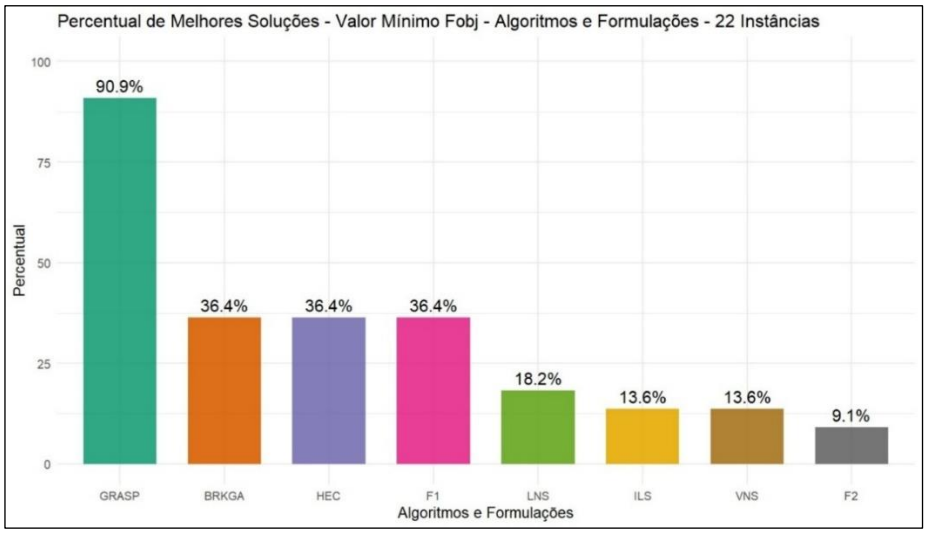
Cabe destacar que, dentro do tempo de 60 minutos, a formulação F1 produziu o ótimo global apenas em 2 das 30 instâncias e a formulação F2 produziu o ótimo

global apenas em 3 das 22 instâncias (no caso de F1: RJ_3300407 e MT_5107602; no caso de F2: SC_4208203, AL_2700300 e RS_4316907)

Por fim, o gráfico na Figura 9 traz os percentuais de melhores soluções produzidas pelos algoritmos e pela formulação F1. Neste último, também é possível observar superioridade do GRASP em relação às demais abordagens, incluindo o algoritmo BRKGA, formulação F1 e a HEC que ficaram novamente empatados em 2º lugar, com percentual de melhores soluções da ordem de 27%, seguidas pela metaheurística LNS com percentual da ordem de 13%.

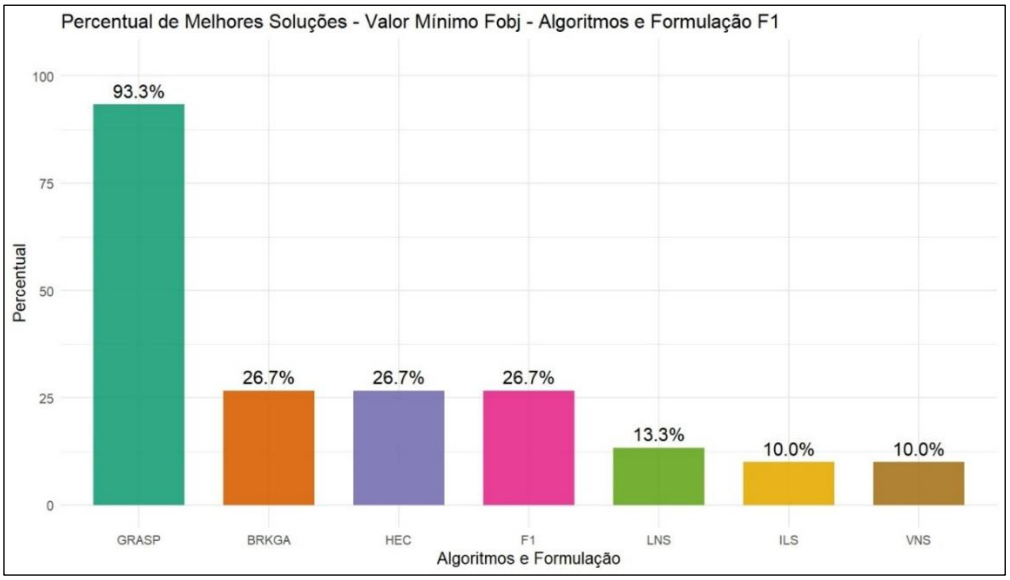
Outra questão relevante em relação ao comportamento de um algoritmo diz respeito à sua estabilidade, ou seja, quando aplicado diversas vezes, em uma mesma instância, espera-se que ocorra baixa variabilidade entre tais resultados. Neste sentido, de forma a realizar tal avaliação, foi realizado um experimento complementar com cinco instâncias escolhidas (contemplando as três classes quanto ao porte) dentre as 30 disponíveis na Tabela 3. Neste experimento, o GRASP foi aplicado 100 vezes em cada uma dessas instâncias, sendo calculado o coeficiente de variação (cv) - razão entre o desvio padrão e a média dos valores das 100 variâncias obtidas. A partir da Tabela 8, que traz os cv's obtidos em valores percentuais para as 5 instâncias, observa-se que o GRASP apresentou, em geral, cv's inferiores a 0.03%, a exceção da instância PE_2604106 com um *cv* da ordem 0.2%; fato que indica a estabilidade do algoritmo.

Figura 8 - Melhores Soluções – Algoritmos e Formulações



Fonte: Autoria própria.

Figura 9 - Melhores Soluções – Algoritmos e Formulação F1



Fonte: Autoria própria.

Tabela 8 - Coeficiente de variação por instância

Base	AC_1200401	AL_2700300	MT_5107602	PE_2604106	RJ_3301009
cv%	0,0174	0,0000	0,0210	0,1857	0,0099

Fonte: Autoria própria.

5 CONCLUSÕES E DESDOBRAMENTOS

O presente estudo apresentou e avaliou diferentes estratégias de resolução para o Problema de Estratificação de Unidades Primárias de Amostragem (PEUPA), uma aplicação de elevada relevância na área de amostragem estatística, cuja formulação pode ser associada a um Problema de Agrupamento Capacitado de difícil solução. Foram desenvolvidas e analisadas abordagens heurísticas e exatas, destacando-se: (i) um algoritmo baseado na metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP); (ii) uma adaptação da heurística da *Envoltória Convexa* (HEC); e (iii) quatro metaheurísticas — *Variable Neighborhood Search* (VNS), *Iterated Local Search* (ILS), *Biased Random-Key Genetic Algorithm* (BRKGA) e *Large Neighborhood Search* (LNS) — implementadas a partir do uso do Otimizador de Chaves Aleatórias (OCA), conforme proposto em Chaves *et al*. (2024). Complementarmente, o PEUPA foi também abordado mediante duas Formulações Matemáticas: uma de Programação Quadrática Inteira (F1) e outra de Programação

Linear Inteira Mista (F2), sendo esta última obtida pela linearização de F1, ambas submetidas ao solver GUROBI nos experimentos computacionais.

No caso do GRASP, tanto o procedimento construtivo quanto as duas estratégias de busca local desenvolvidas foram projetadas para preservar a restrição de capacidade associada ao número mínimo de UPAs por estrato. Já para os quatro algoritmos baseados nas metaheurísticas BRKGA, ILS, LNS e VNS, tendo em vista que os procedimentos do OCA não foram originalmente concebidos para lidar com problemas restritos, tornou-se necessário o emprego de uma função de penalidade que assegurasse a viabilidade das soluções obtidas.

A análise dos resultados experimentais apresentados na Seção 4 evidencia que o algoritmo GRASP se destacou como uma alternativa robusta e eficiente para a resolução do PEUPA, considerando os testes estatísticos aplicados e os percentuais expressivos de soluções de melhor qualidade e de soluções vencedoras obtidas em comparação com as demais abordagens, conforme ilustrado nos gráficos das Figuras 5, 6, 7, 8 e 9. Adicionalmente, a estabilidade do GRASP foi comprovada por meio de experimentos complementares realizados com um conjunto de cinco instâncias — de portes variados — selecionadas dentre as 30 instâncias consideradas nos experimentos principais, resultando em coeficientes de variação significativamente baixos.

Para garantir uma avaliação robusta e comparável dos métodos propostos, cada um dos seis algoritmos — GRASP, HEC, BRKGA, ILS, LNS e VNS — foi executado 20 vezes em 30 instâncias do PEUPA, registrando-se os valores da função objetivo (variância). Essa repetição permitiu analisar medidas como o valor mínimo e a mediana da variância, e aplicar testes estatísticos para comparação de desempenho. Adotou-se um critério de parada proporcional ao porte das instâncias (pequenas, médias e grandes), com tempos máximos entre 3 e 10 minutos, além de limites específicos para a HEC (GUIGNARD; AHLATCIOGLU, 2021) e para as formulações F1 e F2 (60 minutos). A calibração do GRASP, realizada com o *irace* (LÓPEZ-IBÁÑEZ *et al*., 2016), resultou em parâmetros que otimizaram seu desempenho em instâncias representativas.

A análise estatística, conduzida com o Teste de Friedman (HOLLANDER *et al*., 2014), indicou diferenças significativas entre os algoritmos ($p < 2.2 \times 10^{-16}$), sendo

complementada pelo Teste de Wilcoxon pareado com Correção de Bonferroni (HOLLANDER *et al*., 2014; GIBBONS; CHAKRABORTI, 2020). Os resultados apontaram o GRASP como o método de melhor desempenho, seguido por BRKGA e HEC, com destaque para sua estabilidade — coeficientes de variação inferiores a 0,03% — e maior percentual de soluções vencedoras -, confirmando sua robustez e eficiência na resolução do PEUPA.

Como desdobramentos futuros, pretende-se avaliar a integração da atual versão do GRASP com um procedimento de reconexão por caminhos (path-relinking) conforme Resende e Ribeiro (2016), bem como a incorporação de estratégias de busca local mais sofisticadas. Além disso, vislumbra-se o desenvolvimento de uma versão paralelizada do algoritmo, de modo a explorar seu potencial em arquiteturas de processamento de alto desempenho, ampliando a aplicabilidade e a eficiência da abordagem proposta para instâncias de maior dimensão.

AGRADECIMENTOS

Ao CNPq - Projeto Universal 405044/2021-6.

REFERÊNCIAS

ALBIERI, S.; DIAS, A. J. R.(orgs.). **40 anos da unidade de métodos estatísticos do IBGE:** alguns passos. Rio de Janeiro: IBGE, Coordenação de Métodos e Qualidade, 2017. 216 p. (Documentos para Disseminação. Memória Institucional, 22). ISBN 978-85-240-4430-4.

BATISTA, A.S.; FRANÇA, K.C.B.; BERDET, M.; PINTO, M.A.B. Metropolização, homicídios e segurança pública na área metropolitana de Brasília: o município de Águas Lindas de Goiás. **Sociedade & Estado,** Brasília, v. 31, n. 2, p. 433-457, 2016.

BISPO DOS SANTOS, J. A importância das informações estatísticas do Censo do IBGE 2022 para a gestão das políticas públicas no município de Irará (Bahia); **RECIMA21 - Revista Científica Multidisciplinar,** v. 6, n. 3, 2025.

BRITO, J. A. M.; BRITO, L. R. Algoritmos VNS e genéticos aplicados ao problema de agrupamento com soma mínima de distâncias. In: **ANAIS DO XL SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL.** João Pessoa: Sociedade Brasileira de Pesquisa Operacional, 2008. p. 1150–1161.

BRITO, J. A. M.; MONTENEGRO, F. M. T.; FREITAS, M. P. S. Algoritmos de otimização aplicados à estratificação da amostra mestra*. In:* **Anais da III Escola de Amostragem e Metodologia de Pesquisa** - ESAMP, Juiz de Fora, 2011.

BOLFARINE, H.; BUSSAB, W. O. **Elementos de amostragem**. 1. ed. São Paulo: Blucher, 2005. 290 p.

CHAOVALITWONGSE, W. A.; ANDROULAKIS, I. P.; PARDALOS, P. M. Quadratic integer programming: complexity and equivalent forms. In: FLOUDAS, C.; PARDALOS, P. (eds.). **Encyclopedia of Optimization.** Boston: Springer, 2008.

CHAVES, A. A.; RESENDE, M. G. C.; SCHUETZ, M. J. A.; BRUBAKER, J. K.; KATZGRABER, H. G.; ARRUDA, E. F.; SILVA, R. M. A. **A random-key optimizer for combinatorial optimization**. arXiv preprint arXiv:2411.04293, 2024.

COCHRAN, W. G. **Sampling Techniques**. *3. ed. New York: John Wiley & Sons*, 1977.

CONFORTI, M.; CORNUÉJOLS, G.; ZAMBELLI, G**. Integer programming**. Cham: Springer International Publishing, 2014. 787 p.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R.L.; STEIN, C. **Introduction to Algorithms.** 4. ed. Cambridge (MA): The MIT Press, 2022.

DEPARTAMENTO ADMINISTRATIVO NACIONAL DE ESTADÍSTICA (DANE). **Metodología general gran encuesta integrada de hogares (GEIH).** Documento metodológico. Bogotá: DANE, 2016.

FRANK, M.; WOLFE, P. An algorithm for quadratic programming**. Naval Research Logistics Quarterly**, v. 3, n. 1-2, p. 95-110, 1956.

GARCIA, R.P.; DE LIMA, B.S.L.P.; LEMONGE, A.C.C.; JACOB, B.P. An enhanced surrogate-assisted differential evolution for constrained optimization problems. **Soft Computing**, v. 27, p. 6391-6414, 2023.

GASPAR-CUNHA, A.; TAKAHASHI, R. H. C.; ANTUNES, C. H. **Manual de computação evolutiva e meta-heurística.** Belo Horizonte: UFMG, 2013.

GENDREAU, M.; POTVIN, J.-Y. **Handbook of metaheuristics.** 2. ed. New York: Springer, 2010.

GIBBONS, J. D.; CHAKRABORTI, S. **Nonparametric statistical inference.** Statistics: A Series of Textbooks and Monographs. 6. ed. Boca Raton: Chapman and Hall/CRC, 2020.

GUIGNARD, M.; AHLATCIOGLU, A. The convex hull heuristic for nonlinear integer programming problems with linear constraints and application to quadratic 0-1 problems. **Journal of Heuristics**, v. 27, n. 1, p. 251–26, 2021.

HANSEN, P.; JAUMARD, B. Cluster analysis and mathematical programming. **Mathematical Programming,** v. 79, p. 191-215, 1997.

HOLLANDER, M.; WOLFE, D. A.; CHICKEN, E. **Nonparametric Statistical Methods.** 3rd ed. Hoboken: John Wiley & Sons, 2014.

IBGE. **Amostra Mestra Para o Sistema Integrado de Pesquisas Domiciliares:** IBGE, 2007. (Textos para Discussão - Diretoria de Pesquisas, n. 23).

IBGE. **Pesquisa Nacional por Amostra de Domicílios Contínua Notas técnicas Versão 1.15**: IBGE, 2023. (Notas técnicas, n. 1.15).

JIAO, L. C.; LI, L.; SHANG, R. H.; LIU, F.; STOLKIN, R. A novel selection evolutionary strategy for constrained optimization. **Information Sciences,** Amsterdam, v. 239, n. 1, p. 122-141, 2013. DOI: 10.1016/j.ins.2013.03.002.

LEE, J.; LEYFFER, S. (eds.). **Mixed integer nonlinear programming.** v. 154. IMA Volumes in Mathematics and its Applications. New York: Springer, 2012.

LEVIN, M. Sh. Capacitated Clustering Problem. **Journal of Communications Technology and Electronics,** v. 69, n. 1, p. 118-127, 2024.

LOHR, S. L. **Sampling - Design and Analysis**. New York: Chapman and Hall/CRC, 2021.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; PÉREZ CÁCERES, L.; BIRATTARI, M.; STÜTZLE, T. G. The *irace* package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, Amsterdam, v. 3, p. 43-58, 2016.

MALAGUTI, J. G.; ALVES, P. Amostra mestra do Sistema Integrado de Pesquisas Domiciliares Nacional: revisão e discussão das propostas de atualização. **Ciência & Saúde Coletiva,** Rio de Janeiro, v. 29, e03712024, 2024. DOI: 10.1590/1413-812320242911.03712024.

MARTINEZ-GAVARA, A.; LANDA-SILVA, D.; CAMPOS, V.; MARTÍ, R.. Randomized heuristics for the capacitated clustering problem. **Information Sciences**, 417:154–168, 2017.

MARTÍ, R.I; PARDALOS, P.M.; RESENDE, M.G.C. (eds.). **Handbook of Heuristics.** Cham: Springer, 2018. ISBN 978-3-319-07123-7 (impresso); ISBN 978-3-319-07124-4 (e-book).

MENDES, B. G. A investigação da saúde nos censos demográficos do Brasil: possibilidades de análise, vantagens e limitações. **Boletim do Instituto de Saúde** – BIS, São Paulo, v. 16, n. 2, p. 6–14, 2015.

MONTENEGRO, F. M. T.; TORREÃO, J. R. A.; MACULAN, N. Microcanonical optimization algorithm for the Euclidean Steiner problem in $R^n$with application to phylogenetic inference. **Physical Review E,** v. 68, n. 5, 2003.

MUIÑOS, R. **Muestra maestra urbana de viviendas de la República Argentina MMUVRA 2011.** [S.l.], 2018. (Reunión del Grupo de Trabajo sobre Encuestas a Hogares de la Conferencia Estadística de las Américas). Disponível em: https://biblioteca.indec.gob.ar/bases/minde/2mi600.pdf. Acesso em: jun. 2025.

MULVEY, J. M.; BECK, M.P. Solving capacitated clustering problems. **European Journal of Operational Research**, 18:339–348, 1984.

NASCIMENTO, M. C.; TOLEDO, F.M.; DE CARVALHO, A.C. Investigation of a new GRASP-based clustering algorithm applied to biological data. **Computers & Operations Research**, v. 37, n. 8, p. 1381-1388, 2010.

NEGREIROS, M.; MACULAN, N.; PALHANO, W.C.; Muritiba, A.E.F.; BATISTA, P.L.F.. Capacitated Clustering Models to Real-Life Applications. *In:* **IntechOpen**: IntechOpen, 2022. Disponível em: 10.5992/intechopen.1000213. Acesso em: 15 out. 2025.

RESENDE, M.G.C.; RIBEIRO, C.C. **Optimization by GRASP:** Greedy Randomized Adaptive Search Procedures. New York: Springer, 2016.

SERPA, D. R. **Abordagens heurísticas para problemas de agrupamentos.** Dissertação (Mestrado) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2011.

TOSI, D.; KOKAJ, R.; ROCCETTI, M. 15 years of big data: a systematic literature review. **Journal of Big Data,** v. 11, art. 73, 2024.

VON HOHENBALKEN, B. Simplicial decomposition in nonlinear programming algorithms. **Mathematical Programming,** v. 13, p. 49-68, 1977.

WOLSEY, L. A. **Integer Programming.** New York: Wiley-Interscience, 1998.

YAGI, P. A.; QUIROZ, E. A. P.; LENGUA, M. A. C. A systematic literature review on quadratic programming. *In*: **PROCEEDINGS OF THE 7TH INTERNATIONAL CONGRESS ON INFORMATION AND COMMUNICATION TECHNOLOGY (ICICT 2022).** Singapore: Springer Nature, 2022. p. 739–747.

ZHOU, Q.; BENLIC, U.; WU, Q.; HAO, J. Heuristic search to the capacitated clustering problem. **European Journal of Operational Research,** v. 275, p. 123-139, 2018.

Biografia do(s) autor(es)

José André de Moura Brito
Possui bacharelado em Matemática pela UFRJ, Mestrado e Doutorado em Engenharia de Sistemas e Computação (Otimização) pela COPPE/UFRJ e Pós-Doutorado em Otimização na Universidade Federal Fluminense. Atualmente é professor da Escola Nacional de Ciências e Estatística (ENCE/IBGE), onde leciona disciplinas na graduação. Também atua como pesquisador colaborador na Pós-Graduação do Instituto de Computação da Universidade Federal Fluminense. Tem experiência nas áreas de Otimização, Computação e Estatística, atuando principalmente nos seguintes temas: Metaheurísticas, Programação Inteira, Otimização Combinatória, Programação Não Diferenciável e Suavização, Análise de Algoritmos, Análise de Agrupamentos e Amostragem.

Gustavo Silva Semaan
Professor Associado da Universidade Federal Rural do Rio de Janeiro no Instituto Três Rios (ITR/UFRRJ) onde atua desde 2025. Foi professor da Universidade Federal Fluminense (UFF) (2014-2025). Coordenador de Curso da Computação no INF-UFF (2023-2025). Foi professor daAnhanguera / UNIPLI (2010-2013), UniAcademia / CESJF(2008-2010). Pós-doutorado realizado no Laboratório de Inteligência Computacional (LabIC), no Instituto de Computação (IC) da UFF. Doutor e Mestre em Computação pelo IC-UFF. Pós-graduado em Engenharia de Produção pela UniAmerica (2024). Pós-graduado em Ciência de Dados pela UniAmerica (2025). Bacharel em Sistemas de Informação pela Faculdade Metodista Granbery. Técnico em Informática Industrial pela Universidade Federal de Juiz de Fora (CTU-UFJF). Duas vezes indicado ao Prêmio de Excelência em Docência UFF (19 e 22). Professor homenageado em todas as instituições que foi docente: UFF, Anhanguera(UNIPLI), Centro Universitário UniAcademia (CES-JF). Paraninfo ou patrono de todas as turmas da Licenciatura em Computação durante o tempo que foi docente na UFF (8 turmas: 15, 16, 17, 18, 19, 23, 24 e duas turmas em 25). Homenagem: Membro Honorário do Diretório Acadêmico Alan Turing em 22 (INFES/UFF). Projetos de Pesquisa(proponente): FOPESQ/UFF (21-23) e (17-19); Projetos PIBIC (ENCE/IBGE) (início 25),(24-25) e (20-21); FAPERJ IC (17-18) e (14-15); FAPERJ INST(16-18); FOPIN IC (15-16); PIBIC UFF(14-15). Extensão(proponente): Edital de Fluxo Contínuo EExt/UFRRJ (início 26), FAPERJ Jovens Talentos (19-20) e (20-21). Projetos de Pesquisa(colaborador): Projeto CNPq Universal (início 25),(22-25),(2010) e (2007); CNPq CT-INFO (2007), e FAPERJ Pensa Rio (07-10). Membro do Laboratório de Inteligência Computacional (LabIC) desde 2006. Atua com análise e desenvolvimento de sistemas desde 2001, e como professor no magistério superior desde 2008.

Artigo recebido em: 30/10/2025 e aceito para publicação em: 24/05/2026
DOI: https://doi.org/10.14488/1676-1901.v26i2.5713