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

ALGORITMO BRKGA APLICADO AO PROBLEMA DOS K GRUPOS COM CARDINALIDADE FIXA

José André de Moura Brito
ENCE/IBGE
Rua André Cavalcanti, 106, sala 503-A, Bairro de Fátima, Rio de Janeiro - RJ
jose.m.brito@ibge.gov.br

Gustavo Silva Semaan
INFES/UFF
Estr. João Jasbick, s/n - Dezessete, Santo Antônio de Pádua - RJ
gustavosemaan@id.uff.br

Augusto Cesar Fadel
IBGE
Avenida República do Chile, 500, 10o. andar, Centro, Rio de Janeiro - RJ
augustofadel@gmail.com

RESUMO

Este artigo propõe um algoritmo baseado na metaheurística brkga (Biased random-key genetic algorithms), aplicado à resolução do problema dos K grupos com cardinalidade fixa. Neste problema, deve-se definir K grupos, cada um contendo exatamente M_c objetos $(c = 1, \ldots, K)$, selecionados a partir de um conjunto X de N objetos. Adicionalmente, a soma das similaridades (função objetivo do problema) entre todos os pares de objetos alocados a cada um dos grupos deve ser maximizada. Em experimentos realizados com 34 instâncias da literatura, variando-se os valores de K e M_c, observou-se, mediante conjunto de análises e aplicação de teste de hipótese, a eficácia do algoritmo BRKGA, que produziu soluções de boa qualidade, frente a dois algoritmos da literatura e uma formulação de programação inteira.

PALAVRAS CHAVE. Agrupamento, Metaheurísticas, BRKGA.

Tópicos (MH, PM, OC)

ABSTRACT

This article proposes an algorithm based on the brkga metaheuristic applied to resolve the problem of K clusters with fixed cardinality. In this problem, we must define K clusters, each one containing exactly M_c objects $(c = 1, \cdots, K)$, selected from a set X of N objects. Additionally, the sum of similarities (objective function of the problem) between all the object pairs allocated to each cluster must be maximized. In experiments carried out with 34 instances from the literature, varying the values of K and M_c, it was observed, through a set of analysis and application of hypothesis test, the effectiveness of the BRKGA algorithm, which produced solutions of good quality, against two algorithms from the literature and an integer programming formulation.

KEYWORDS. Clustering. Metaheuristic. BRKGA.

Paper topics (MH, PM, OC)

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

1. Introdução

Existem, na literatura, diversas aplicações reais em Estatística, Marketing, Genética, Astronomia, Psiquiatria, Arqueologia etc [Everitt et al., 2011], cuja modelagem e resolução remetem a um problema de agrupamento. Em linhas gerais, nesse problema [Johnson e Wichern, 2018], deve-se segmentar uma base de dados, composta por n objetos, em k grupos homogêneos, de acordo com uma métrica definida a priori - distância euclidiana, manhattan etc. Por sua vez, a construção de tais grupos é realizada, em geral, mediante a aplicação de algoritmos clássicos, denotados por não-hierárquicos (k-médias, etc) ou hierárquicos (ligação simples, completa etc) [Hair et al., 2018].

Todavia, para determinadas aplicações reais associadas a problemas de agrupamento, a utilização de tais algoritmos nem sempre é possível, devido à existência de restrições específicas da aplicação, caracterizando um problema de agrupamento capacitado (PAC). Por exemplo, quando há a exigência de cardinalidade mínima e/ou máxima em relação ao número de objetos por grupo ou um valor mínimo ou máximo em relação à soma de um dos atributos (variáveis) dos objetos por grupo, conforme os problemas abordados em [Negreiros e Palhano, 2006], [Oliveira e Lorena, 2017] e [Fadel et al., 2021], sendo necessário, nestes casos, a aplicação de algoritmos baseados em metaheurísticas.

Em particular, neste trabalho, aborda-se o problema dos K grupos com cardinalidade fixa [Gonçalves e Lourenço, 2019], um PAC onde os objetos devem ser alocados a K grupos, de forma que o total de objetos alocados seja menor ou igual ao total de objetos da base de dados e a soma das similaridades entre todos os pares de objetos, dentro dos K grupos, seja maximizada. Neste trabalho, para resolver tal problema, desenvolveu-se um algoritmo baseado na metaheurística BRKGA.

Além da introdução, o presente trabalho está dividido em mais 4 seções, quais sejam: Na seção dois é apresentada a descrição detalhada desse problema e a revisão da literatura. Em seguida, a seção três traz os conceitos gerais da metaheurística BRKGA e a seção quatro apresenta a descrição do algoritmo proposto. Por fim, na seção cinco são apresentados os resultados computacionais e análises obtidos a partir da aplicação do algoritmo proposto, considerando a sua comparação com dois algoritmos da literatura e uma formulação de programação linear inteira.

2. Problema dos K grupos com Cardinalidade Fixa - PKGCF

Considere um conjunto X (associado à base de dados) formado por N objetos x_i, com l atributos (quantitativos, qualitativos ou mistos), tal que $x_i = (x_i^1, x_i^2, ..., x_i^l); i = 1, \ldots, N$. Neste problema, deve-se determinar K grupos G_c $(c = 1, \ldots, K)$ disjuntos, constituídos por, exatamente, M_c objetos $x_i \in X$, de forma que $\sum_{c=1}^{K} M_c \leq N$. Considerando esta restrição, os objetos devem ser alocados aos grupos de forma que a soma das similaridades s_{ij} entre todos os pares de objetos $x_i, x_j \in X$, dentro de cada um dos grupos, seja maximizada, ou seja:

$$Maximizar \sum_{c=1}^{K} \sum_{\forall x_i, x_j \in G_c} s_{ij} \tag{1}$$

Os valores de s_{ij} são calculados a partir dos l atributos de x_i e x_j, utilizando-se o coeficiente de Jaccard, também considerado em [Cavique, 2004]. Outros tipos de coeficientes são apresentados, por exemplo, em [Hansen e Jaumard, 1997; Johnson e Wichern, 2018].

O PKGCF, que corresponde a um problema de otimização combinatória NP-difícil e tem o problema de k-cluster como um caso particular, conforme [Billionnet, 2005], foi inicialmente abordado por [Gonçalves e Lourenço, 2009], considerando a aplicação de uma formulação de

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

programação linear inteira (PLI). Adicionalmente, de acordo com [Cavique, 2004], possui relação com o problema de clique de peso máximo; além de outros problemas de agrupamento computacionalmente difíceis, conforme [Brito et al., 2022; Semaan, 2013; Bruglieri et al., 2006].

Há várias aplicações reais associadas ao PKGCF, dentre as quais, destacam-se: segmentação de consumidores, marketing, categorização de documentos e análise de dados. Por exemplo, considerando uma estratégia de marketing [Cavique, 2004], grandes supermercados definem diferentes grupos de produtos que são organizados juntos, em uma mesma seção (prateleiras) do supermercado. Neste caso, cada um dos grupos é constituído por um número fixo de produtos (objetos), e a similaridade entre cada par de produtos é baseada na frequência em que tais produtos são comprados em conjunto. Outra aplicação importante do PKGCF diz respeito à área financeira, onde N consumidores devem ser agrupados em K portifólios com M_c consumidores cada, de forma a maximizar a similaridade entre eles, baseando-se nos atributos associados aos seus perfis.

Assim como em outros problemas de agrupamento de difícil solução [Brito et al., 2022; Fadel et al., 2021; Oliveira e Lorena, 2017; Negreiros e Palhano, 2006], são propostos, na literatura, alguns algoritmos heurísticos para a resolução do PKGCF. O trabalho de [Gonçalves et al., 2010] traz uma heurística que combina um procedimento de construção guloso e dois procedimentos de busca local. Em [Gonçalves et al., 2012] é apresentado um algoritmo genético que incorpora uma estratégia elitista. Adicionalmente, em [Gonçalves e Lourenço, 2019] são propostas seis formulações de PLI para a resolução deste problema, que diferem quanto ao tipo e ao número de restrições, que tornam uma formulação melhor do que a outra, no que concerne aos limitantes produzidos e à quantidade de instâncias resolvidas. Por fim, em [Ghirardi, 2021] é proposta uma matheurística, tendo por base a utilização de duas formulações apresentadas em [Gonçalves e Lourenço, 2019] e um procedimento de busca local.

A seguir, apresenta-se a formulação $F2$ - proposta em [Gonçalves e Lourenço, 2019] e adotada nos experimentos realizados neste trabalho, devido à sua eficácia. De acordo com os autores, a sua aplicação possibilitou resolver um número maior de instâncias, quando comparada às demais formulações.

$$Max \sum_{c=1}^{K} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} s_{ij} y_{ij}^{c} \tag{2}$$

$$\sum_{c=1}^{K} x_{ic} \leq 1 \ , \quad i = 1, \ldots, N \tag{3}$$

$$\sum_{i=1}^{N} x_{ic} = M_c \ , \quad c = 1, \ldots, K \tag{4}$$

$$y_{ij}^{c} \geq x_{ic} + x_{jc} - 1 \ , \quad 1 \leq i < j \leq N; c = 1, \ldots, K \tag{5}$$

$$\sum_{i=1}^{j-1} y_{ij}^{c} + \sum_{i=j+1}^{N} y_{ji}^{c} = (M_c - 1) x_{jc} \ , \quad j = 1, \ldots, N; c = 1, \ldots, K \tag{6}$$

$$x_{ic} \in \{0, 1\} \ , \ i = 1, \ldots, N; c = 1, \ldots, K \tag{7}$$

$$0 \leq y_{ij}^{c} \leq 1, \quad 1 \leq i < j \leq N; c = 1, \ldots, K \tag{8}$$

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

Nesta formulação, a função objetivo a ser maximizada em (2) corresponde à soma das similaridades entre todos pares de objetos alocados a cada um dos K grupos. A variável binária x_{ic} assume valor 1 se o objeto i é alocado ao grupo G_c e a variável y_{ij}^c assume valor 1 se os objetos i e j são alocados a um mesmo grupo. A restrição (3) garante que cada objeto será alocado a, no máximo, um dos grupos. A restrição (4) garante que cada grupo tem, exatamente, M_c objetos alocados. A restrição (5) faz y_{ij}^c assumir valor 1 sempre que $x_{ic} = 1$ e $x_{jc} = 1$ e a restrição (6) força $y_{ij}^c = 1$ para $M_c - 1$ variáveis y_{ij}^c, para j e c fixos.

3. Metaheurística BRKGA

A metaheurística BRKGA (Biased Random-Key Genetic Algorithm) [Gonçalves e Resende, 2011], tem sido aplicada com sucesso em diversos problemas de otimização [Fadel et al., 2021; Oliveira e Lorena, 2017; Festa, 2013]. Nessa metaheurística, a população é definida por p cromossomos - vetores de chaves aleatórias, denotados por $u_r(r = 1, \ldots, p)$, com m componentes que têm valores reais gerados segundo $U\,[0, 1]$. Em cada geração do BRKGA aplica-se, na população atual, um procedimento denominado decodificador (específico do problema abordado) e os operadores de seleção, cruzamento e mutação.

O decodificador ($D(.)$), quando aplicado em cada vetor u_r, produz uma solução viável s_r associada, utilizada para o cálculo da função objetivo $f(.)$ do problema. Em seguida, considerando os valores de $f(s_1), \ldots, f(s_p)$ obtidos a partir das m soluções s_r decodificadas, aplica-se o operador de seleção, de forma a rankear tais soluções, possibilitando que a população (vetores de chaves) seja dividida em um conjunto elite C_e - formado pelos p_e vetores de chaves aleatórias (correspondente a um percentual de p), associados às melhores soluções, segundo o valor de $f(.)$ e um conjunto não-elite C_{ne}, formado pelos $p - p_e$ vetores restantes, tal que $p_e < p - p_e$.

A população utilizada em cada geração é composta por: (i) os p_e vetores do conjunto C_e definido na geração anterior; (ii) p_m vetores mutantes, correspondentes a vetores de chaves aleatórias gerados de forma análoga aos vetores da população inicial (1^a geração) e (iii) $p - p_e - p_m$ vetores filhos, obtidos a partir da aplicação do procedimento de cruzamento uniforme, onde cada um desses vetores é produzido selecionando-se, para o cruzamento, um vetor v_1 de C_e e um vetor v_2 de C_{ne}. Os critérios de parada adotados no BRKGA são o número máximo de gerações, número de gerações sem melhoria ou tempo máximo de execução. O Quadro 1 traz um pseudocodigo que resume todos os passos do BRKGA.

Algoritmo 1: Pseudocódigo do BRKGA

```
Gerar a população inicial P_0 composta por p vetores de chaves aleatórias
g ← 0
enquanto (g < MaxGeracoes) faça
    Aplicar D(.) em cada u_r ∈ P_g, produzindo as soluções s_r, r = 1, ..., p
    Calcular f(s_r), r = 1, ..., p, e ordenar, decrescentemente, os vetores de P_g de acordo
      com o valor da função objetivo - problema de maximização
    Selecionar os p_e vetores de chaves aleatórias correspondentes às melhores soluções
      ordenadas de P_g e copiar esses vetores para a população P_{g+1}
    Gerar p_m vetores mutantes analogamente ao passo 1 e adicionar a P_{g+1}
    Cruzar os p_e vetores de C_e com os (p − p_e) vetores de C_ne, ambos de P_g (atual),
      produzindo (p − p_e − p_m) vetores u_r e adicioná-los a P_{g+1}
    g ← g + 1
fim
```

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

4. Algoritmo BRKGA - Representação, Decodificação e Cálculo da Função Objetivo

No BRKGA desenvolvido para o PKGCF, cada cromossomoso é definido como um vetor u_r com N posições (n^o de objetos de X) preenchidas com valores reais gerados segundo uma uniforme $[0, 1]$. O decodificador é aplicado em duas etapas: na primeira, cada um dos N objetos de X é, inicialmente, alocado a um dos K grupos $G_1, \ldots, G_c, \ldots, G_K$, dividindo-se o intervalo $[0, 1]$ em K subintervalos I_c da forma:

$$I_1 = [0, \frac{1}{K}), \ldots, I_{K-1} = [\frac{K-2}{K}, \frac{K-1}{K}), I_K = [\frac{K-1}{K}, 1]$$

Em seguida, toma-se cada $u_r^i \in u_r$ e verifica-se a qual intervalo I_c $(c = 1, \ldots, K)$ este valor pertence; onde c corresponde ao grupo G_c ao qual objeto x_i será alocado. Assim, produz-se um vetor v com N entradas, cada uma com valor entre 1 e K, que indica o grupo G_c ao qual objeto x_i está alocado. Em seguida, a partir de v, calcula-se a frequência absoluta de cada um dos valores entre 1 e K, produzindo-se um vetor $q = (q_1, \ldots, q_c, \ldots, q_K)$, tal que $q_c = |G_c|,\ c = 1, \ldots, K$.

Após esta etapa, pode ser observada a ocorrência de alguns grupos em que $q_c > M_c$ e/ou $q_c < M_c$ $(c = 1, \ldots, K)$. Assim sendo, na 2^a etapa foi elaborado um procedimento heurístico, que realiza a atualização de v, realocando os objetos entre os grupos, de forma que, simultaneamente, a restrição de cardinalidade seja atendida e haja o maior acréscimo possível na função objetivo.

Algoritmo 2: Decodificador - 2^a Etapa

```
Q_e = ∅
para c ∈ {1, ..., K} | q_c > M_c faça
    Q = ∅
    para cada x_i ∈ G_c faça
        s_i = Σ_{∀x_j ∈ G_c | x_j ≠ x_i} s_ij
        Q = Q ∪ {x_i}
    fim
    Atribuir 0 às (q_c − M_c) posições de v associadas aos objetos x_i ∈ Q com os (q_c − M_c)
      menores valores s_i correspondentes.
    Adicionar em Q_e os índices i tais que v_i = 0
fim
para c ∈ {1, ..., K} | q_c < M_c faça
    Q = ∅
    para cada x_i ∈ Q_e faça
        s_i = Σ_{∀x_j ∈ G_c | x_j ≠ x_i} s_ij
        Q = Q ∪ {x_i}
    fim
    Atribuir o valor c às (M_c − q_c) posições de v associadas aos objetos x_i ∈ Q com os
      (M_c − q_c) maiores valores s_i correspondentes.
    Remover de Q_e os índices i correspondentes às posições de v que foram preenchidas
      com valor c no passo anterior
fim
```

Na 2^a etapa, conforme o pseudo-código acima, toma-se cada grupo G_c onde $q_c > M_c$ e removem-se os $(q_c - M_c)$ objetos com as menores somas de similaridades em relação aos demais objetos de G_c. Tais objetos são adicionados a um conjunto Q_e, que contém os objetos definidos como "excedentes"e as posições de v (correspondentes a tais objetos) são preenchidas com 0. Em seguida, em cada grupo onde $q_c < M_c$ são utilizados os objetos de Q_e, tendo como critério de

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

alocação os objetos que são mais similares em relação ao grupo, de forma a produzir o maior acréscimo possível à função objetivo. Isso implica que, algumas das posições de v preenchidas com zero no início desta etapa, serão preenchidas com valores correspondentes aos números (rótulos) dos grupos em que $q_c < M_c$. A seguir apresenta-se um exemplo da aplicação do decodificador.

Exemplo: Supondo $N = 10, K = 3, M_1 = 3, M_2 = 2, M_3 = 3$, u_r=(0.15,0.45,0.71,0.09, 0.49,0.23,0.14,0.31,0.89,0.05) e os intervalos $[0, \frac{1}{3}), [\frac{1}{3}, \frac{2}{3}), [\frac{2}{3}, 1] \Rightarrow v = (1, 2, 3, 1, 2, 1, 1, 1, 3, 1)$, sendo $q = (6, 2, 2)$ e $G_1 = \{x_1, x_4, x_6, x_7, x_8, x_{10}\}$, $G_2 = \{x_2, x_5\}$, $G_3 = \{x_3, x_9\}$. Neste caso: $q_1 > M_1$, $q_2 = M_2$ e $q_3 < M_3$. De forma a ilustrar a aplicação do 2^a etapa do decodificador, além dessas informações, será considerada a matriz $S_{10\times10}$ representada na Figura 1 abaixo, cujas entradas s_{ij} contêm os valores das similaridades entre todo os pares de objetos x_i e x_j.

Figura 1: Matriz de similaridades

	1	2	3	4	5	6	7	8	9	10
1	0.000	0.332	0.187	0.025	0.250	0.248	0.591	0.169	0.245	0.637
2	0.332	0.000	0.144	0.357	0.081	0.084	0.260	0.501	0.086	0.306
3	0.187	0.144	0.000	0.213	0.063	0.060	0.404	0.356	0.058	0.450
4	0.025	0.357	0.213	0.000	0.276	0.273	0.616	0.144	0.271	0.663
5	0.250	0.081	0.063	0.276	0.000	0.003	0.341	0.419	0.005	0.387
6	0.248	0.084	0.060	0.273	0.003	0.000	0.343	0.417	0.002	0.390
7	0.591	0.260	0.404	0.616	0.341	0.343	0.000	0.760	0.346	0.046
8	0.169	0.501	0.356	0.144	0.419	0.417	0.760	0.000	0.414	0.806
9	0.245	0.086	0.058	0.271	0.005	0.002	0.346	0.414	0.000	0.392
10	0.637	0.306	0.450	0.663	0.387	0.390	0.046	0.806	0.392	0.000

Neste exemplo, o grupo G_1 tem cardinalidade superior àquela previamente definida. Assim sendo, ao aplicar os passos 1-10 (Algoritmo 2), calcula-se $s_1 = s_{14} + s_{16} + s_{17} + s_{18} + s_{1,10} = 1.670$, $s_4 = 1.721, s_6 = 1.671$, $s_7 = 2.356$, $s_8 = 2.296$, $s_{10} = 2.542$, sendo os três menores valores correspondentes aos objetos x_1, x_4 e x_6. Logo, $v = (\mathbf{0}, 2, 3, \mathbf{0}, 2, \mathbf{0}, 1, 1, 3, 1)$ e $Q_e = \{1, 4, 6\}$. Aplicando-se os passos 11-19 e considerando o grupo G_3 (com cardinalide inferior àquela definida), temos $s_1 = s_{13} + s_{19} = 0.432$, $s_4 = s_{43} + s_{49} = 0.484$ e $s_6 = s_{63} + s_{69} = 0.062$. Como s_4 tem o maior valor, o objeto x_4 será alocado a grupo G_3, logo $v = (0, 2, 3, \mathbf{3}, 2, 0, 1, 1, 3, 1)$, o que implica $G_1 = \{7, 8, 10\}$, $G_2 = \{2, 5\}$ e $G_3 = \{3, 4, 9\}$, atendendo a restrição de cardinalidade por grupo.

Observa-se que as atualizações efetuadas nos grupos G_c que não safisfazem à restrição de cardinalidade ($q_c \neq M_c$) produzem um novo vetor v que não corresponde ao vetor u_r inicial. Assim sendo, de forma garantir tal correspondência, aplica-se uma decodificação reversa. Mais especificamente, para cada $v_i \neq 0$ toma-se um valor real gerado segundo uniforme no intervalo $I_{v_i} = [\frac{v_i-1}{K}, \frac{v_i}{K})$ e atribui-se tal valor à posição correspondente de u_r. Por exemplo, no vetor v atualizado acima, $v_4 = 3$; logo, atribui-se à 4^a posição de u_r um valor gerado a partir de $I_3 = [\frac{2}{3}, 1)$. As posições de u_r correspondentes às posições de v com valor zero não são atualizadas, sendo tal vetor considerado na geração seguinte. A adoção de tal estratégia tem, por objetivo, dar maior diversidade às soluções decodificadas - grupos produzidos.

5. Resultados Computacionais

Esta seção traz os resultados computacionais obtidos a partir da aplicação do algoritmo BRKGA, da formulação apresentada na seção 2 e de dois algoritmos da literatura, quais sejam: o algoritmo (LS) proposto por [Gonçalves et al., 2010] - baseado em dois procedimentos de busca local e o algoritmo genético (EGH) que adota uma estratégia elitista, proposto em [Gonçalves et al., 2012]. Os três algoritmos e a formulação[1] foram implementados em linguagem R[2]. Todos os expe-

[1]Na formulação utilizou-se o pacote gurobi do R - com rotinas do solver GUROBI - versão 10.0 - lic. acadêmica.
[2]Ambiente RStudio - versão 2023.03.0+386.

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

rimentos foram realizados em um computador com 32GB de memória RAM e processador AMD FX-6300 3.5 GHz e Windows 8 (64 bits).

Para avaliar os três algoritmos e a formulação, foram utilizadas 34 instâncias do problema do k cluster, as mesmas consideradas em [Gonçalves e Lourenço, 2019], sendo cada instância representada por um grafo G (sua lista de arestas), com densidade d e número de vértices correspondente ao número de objetos N considerado no PKGCF.

Em seguida, construiu-se para cada uma das instâncias, uma matriz $S = [s_{ij}]_{N \times N}$, cujas entradas têm valores reais, gerados segundo uma uniforme [0,1] e que correspondem aos pesos das arestas do grafo G associado. No contexto do PKGCF, esses valores correspondem às similaridades entre os objetos de X. Adicionalmente, antes de aplicar os três algoritmos e a formulação, foram definidos, a partir das 34 instâncias, 148 problemas mediante combinações de diferentes valores para os parâmetros K e M_c. A Tabela 1 a seguir traz informações sobre as instâncias utilizadas nos experimentos, quais sejam: total de instâncias, número de objetos N (vértices do grafo), densidade d e as combinações desses parâmetros; totalizando 148 problemas (instâncias x combinações de K e M_c). Por exemplo, há três instâncias com número de objetos $N = 40$ e densidade de 50% e duas instâncias com número de objetos $N = 80$ e densidade de 75%.

Tabela 1: Informações sobre as Instâncias Utilizadas

N^o de Inst.	N	d	Valores de K e M_c adotados para cada instância	N^o de Probs
3	40	25	$\{K = 2, M_c = (4, 16); M_c = (10, 10); M_c = (15, 15); M_c = (5, 25)\}$,	
3	40	50	$\{K = 3, M_c = (10, 10, 10)\}; M_c = (5, 10, 15)\}$,	63
3	40	75	$\{K = 4, M_c = (5, 5, 10, 10)\}$	
1	80	25	$\{K = 2, M_c = (20, 20); M_c = (20, 40)\}, \{K = 3, M_c = (10, 20, 30)\}$,	
2	80	50	$\{K = 4, M_c = (5, 10, 20, 30)\}$,	30
2	80	75	$\{K = 5, M_c = (5, 10, 15, 20, 20)\}; M_c = (10, 10, 10, 10, 10)\}$	
1	100	25	$\{K = 3, M_c = (20, 20, 20)\}, \{K = 4, M_c = (10, 10, 20, 40)\}$,	
2	100	50	$\{K = 5, M_c = (10, 10, 10, 20, 20)\}, \{K = 6, M_c = (5, 10, 10, 15, 20, 20)\}$	20
2	100	75		
1	120	25	$\{K = 3, M_c = (10, 30, 50)\}, \{K = 4, M_c = (10, 20, 20, 40)\}$,	
2	120	50	$\{K = 5, M_c = (5, 10, 20, 20, 40)\}$	15
2	120	75		
1	140	25	$\{K = 3, M_c = (20, 40, 60)\}, \{K = 4, M_c = (10, 20, 40, 40)\}$	
2	140	50		10
2	140	75		
1	160	25	$\{K = 2, M_c = (40, 80)\}, \{K = 3, M_c = (20, 40, 60)\}$	
2	160	50		10
2	160	75		

5.1. Experimentos de Calibração

É fato conhecido na literatura que a perfomance de uma metaheurística, traduzida pela qualidade de seus resultados, está intrinsecamente associada aos valores adotados em seu conjunto de parâmetros. No caso do BRKGA proposto neste trabalho, os valores de tais parâmetros foram definidos, tendo por base, as recomendações feitas em [Gonçalves e Resende, 2011] e a realização de um experimento preliminar de calibração, usando os parâmetros e valores da Tabela 2.

No caso do algoritmo EGH, os parâmetros tamanho da população e número de gerações, definidos inicialmente por 20 e 100, foram multiplicados por 10, sendo esses os valores adotados em todos os experimentos realizados neste artigo. No caso do algoritmo LS, seus procedimentos de construção e busca local foram executados 50 vezes (inicialmente uma execução apenas), sendo considerada como melhor solução aquela obtida ao final das 50 iterações. Tais alterações tiveram, por objetivo, proporcionar uma comparação mais igualitária entre os três algoritmos.

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

Tabela 2: BRKGA - Parâmetros considerados nos experimentos de Calibração

Parâmetro	**Descrição**	**Valores**
p	Tamanho da População	250, 500, 1000
g_{Max}	Número Máximo de Gerações	50, 75, 100
p_e	Tamanho da população elite	$0.1p$, $0.2p$, $0.3p$
p_m	Tamanho da população mutante	$0.1p$, $0.2p$, $0.3p$
p_c	Probabilidade de Cruzamento	0.6, 0.7, 0.8

Para realizar a calibração, foram considerados os seis problemas apresentados na Tabela 3 a seguir, sendo o algoritmo BRKGA aplicado 10 vezes em cada um desses problemas, considerando cada uma das 243 combinações dos cinco parâmetros da Tabela 2. A partir dos resultados das 10 execuções (obtidos para cada combinação) foi calculada a média dos valores da função objetivo (equação 1). Em seguida, tomou-se, para cada um dos problemas, as 5 combinações correspondentes às 5 maiores médias, utilizando-se a combinação comum com maior valor da média associado, obtendo-se, assim, o seguinte conjunto de parâmetros: p=500, g_{Max}=75, $p_e = 0.1p = 50$, $p_m = 0.3p = 150$, $p_c = 0.7$.

Tabela 3: Problemas utilizados nos experimentos de calibração

Instância	**Valores de K e M_c**
$N = 40$ e $d = 75$	$\{K = 3, M_c = (5, 10, 15)\}$
$N = 80$ e $d = 50$	$\{K = 4, M_c = (4, 5, 10, 20, 30)\}$
$N = 100$ e $d = 75$	$\{K = 3, M_c = (20, 20, 20)\}$
$N = 120$ e $d = 50$	$\{K = 5, M_c = (5, 10, 20, 20, 40)\}$
$N = 140$ e $d = 75$	$\{K = 4, M_c = (10, 20, 40, 40)\}$
$N = 160$ e $d = 25$	$\{K = 3, M_c = (20, 40, 60)\}$

5.2. Resultados e Análises

Considerando os valores $p, g_{Max}, p_e, p_m, p_c$ definidos na calibração, de forma a avaliar a eficácia do BRKGA frente aos algoritmos LS e EGH, cada um desses algoritmos foi aplicado 10 vêzes em cada um dos 148 problemas da Tabela 1, sendo calculada, por problema, a média e a melhor solução (valor da função objetivo) das 10 execuções. A análise dos três algoritmos teve, por base, a aplicação de teste de hipótese para validação estatística dos resultados. Mais especificamente, utilizou-se o teste t de *Student* [Triola, 2013] para verificar se existe diferença significativa entre o BRKGA e os algoritmos LS e EGH, no que concerne à qualidade das soluções produzidas - valor de função objetivo. No teste t, a hipótese nula (H_0) afirma que as médias de duas populações são iguais, enquanto a hipótese alternativa (H_1) afirma que as médias são diferentes.

Considerando a média e o desvio-padrão dos resultados da função objetivo do PKGCF, pode-se aplicar o teste *t* (bilateral) para verificar se a diferença entre o BRKGA e os algoritmos EGH e LS é significativa para determinado problema, considerando um nível de significância de 5%. Se a diferença é significativa e o BRKGA apresenta melhor (ou pior) resultado, então diz-se que ele é significativamente melhor (ou pior) para aquele problema; caso contrário (p-valor $> 5\%$), diz-se que os algoritmos apresentam performance equivalente (ou similar). A partir da aplicação desse teste, foram obtidos os resultados apresentados na Tabela 4, sendo observada grande superioridade do BRKGA em relação aos algoritmos EGH e LS, quanto ao percentual de soluções significativamente melhores.

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

Tabela 4: Resultados de Aplicação do Teste de Hipótese

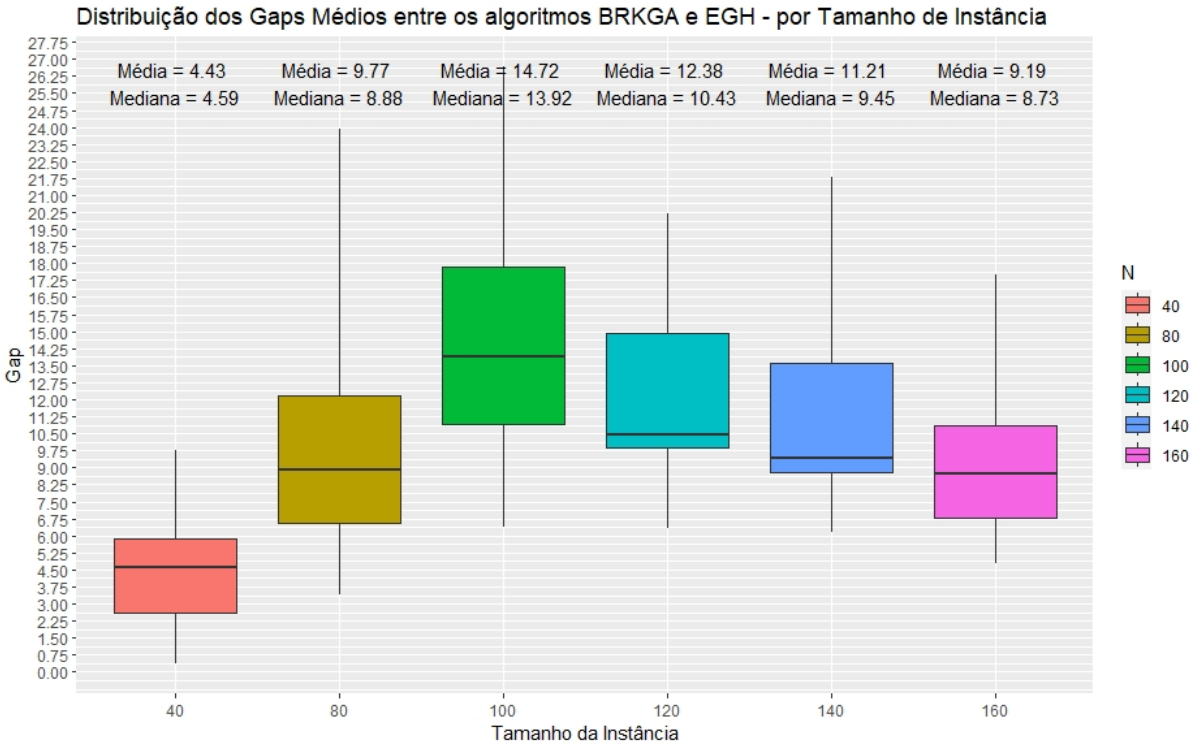
Algoritmos	Melhor	Equivalente	Pior
BRKGA × EGH	99.3%	0.7%	0.0%
BRKGA × LS	87.2%	12.8%	0.0%

Em sequência à aplicação do teste de hipótese, calculou-se o *gap* relativo por problema (instância x K x M_c), entre as médias dos valores de função objetivo obtidos nas 10 execuções (para o i-ésimo problema) produzidas pelo algoritmo BRKGA e pelos os algoritmos EGH e LS, utilizando-se a equação (9), sendo M^i_{BRKGA} e $M^i_{EGH,LS}$ correspondentes às médias do BRKGA e do algoritmo EGH ou LS. A partir deste cálculo, foram construídos os boxplots apresentados nas Figuras 2 e 3, que trazem a distribuição dos *gaps* por algoritmo x tamanho das instâncias (N).

$$100 \times (M^i_{BRKGA} - M^i_{EGH,LS})/M^i_{EGH,LS} \; ; \; i = 1, \ldots, 148 \tag{9}$$

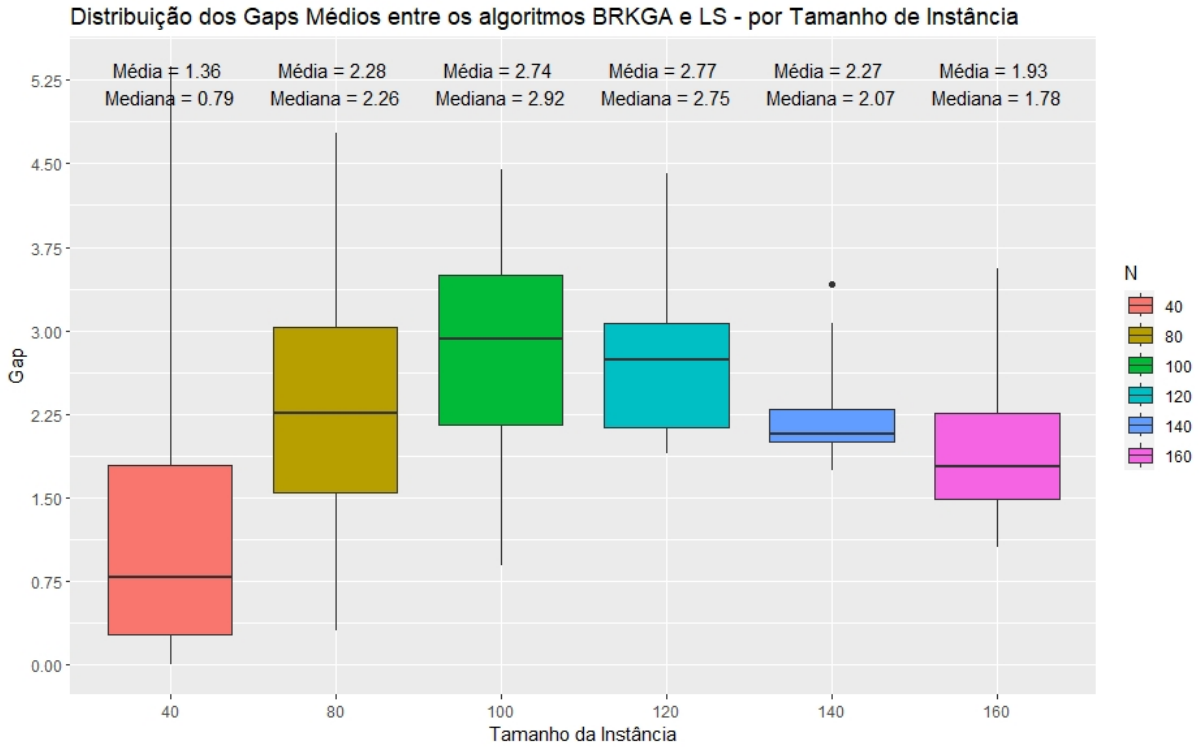
Uma análise desses gráficos mostra que o algoritmo BRKGA tem melhores resultados em relação ao algoritmo EGH, com a média dos gaps variando entre 4.43 e 14.72 e a mediana variando entre 4.59 e 13.92. No caso do algoritmo LS, a média dos gaps variou entre 1.36 e 2.77 e a mediana variou entre 0.79 e 2.92. Em ambos os casos, as maiores diferenças foram observadas para as instâncias com tamanhos de 100 e 120.

Figura 2: Boxplot com Gaps entre os Algoritmos BRKGA e EGH



DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

Figura 3: Boxplot com Gaps entre os Algoritmos BRKGA e LS



Adicionalmente, a Tabela 5 traz os percentuais de melhores soluções produzidas pelos 3 algoritmos, considerando o maior valor de função objetivo obtido na 10 execuções, além do valor de função objetivo da formulação, produzido no tempo máximo de 3 horas. Analisando os percentuais dessa tabela, observa-se, novamente, a boa performance do BRKGA, com 93% de melhores soluções, seguido pela formulação com 43% e pelo algoritmo LS com cerca de 22%. Cabe ressaltar, ainda, que a formulação produziu o ótimo global em cerca de 43% dos casos (63 problemas) e, dentro deste percentual, o BRKGA produziu o ótimo global em cerca de 84% dos casos (53 dos 63 problemas)

Tabela 5: Percentuais de Melhores Soluções - Algoritmos (10 execuções) e Formulação (Tempo de 3 horas)

Algoritmos	Maior Valor de Fobj
BRKGA	93.2%
EGH	5.4%
LS	21.6%
Formulação	43.2%

Por fim, a Tabela 6 traz estatísticas em relação às médias dos tempos das 10 execuções dos algoritmos BRKGA, EGH e LS. Observa-se que, tanto em termos médios ($\overline{x}$) quanto em medianos (Med), o BRKGA demanda maior tempo computacional para produzir soluções de qualidade superior àquelas dos algoritmos EGH e LS. No caso da formulação, a menos dos problemas associados às instâncias de tamanho 40, onde as estatísticas (tempos em segundos) foram $min = 5.6$; $q_1 = 26.8$; $Med = 81.1$; $\overline{x} = 330.3$; $q_3 = 384.3$ e $Max = 2575.8$; a partir das instâncias de tamanho 80, todas as estatísticas foram de 10800 segundos (tempo máximo de execução da formulação).

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

Tabela 6: Estatísticas dos Tempos de Processamento (segundos) Algoritmos x Tamanho de Instâncias*

	N=40			N=80			N=100		
	BRKGA	EGH	LS	BRKGA	EGH	LS	BRKGA	EGH	LS
Min	16.5	8.3	0.8	46.6	10.3	2.9	68.9	14.4	5.3
q_1	19.5	9.0	0.8	57.2	13.7	3.3	74.1	18.2	5.5
Med	25.5	10,1	0.9	67.9	16.6	3.5	81.3	19.6	5.8
$\overline{x}$	24.7	10.6	0,9	64.7	15.8	3.5	80.9	19.1	5.8
q_3	28.7	11.5	0.9	72.1	18.7	3.7	87.2	20.6	6.1
Max.	37.1	14.8	1.2	74.2	20.7	4	92.2	22.7	6.2
	N=120			N=140			N=160		
	BRKGA	EGH	LS	BRKGA	EGH	LS	BRKGA	EGH	LS
Min	80.9	20.6	8.2	88.8	24.2	10.4	91.9	27.1	17.1
q_1	81.5	20.7	8.6	89.1	24.5	10.5	92.8	27.3	17.4
Med	88.7	20.7	8.8	94.0	25.6	11.2	96.8	29.2	17.9
$\overline{x}$	89.0	21.7	8.8	93.8	25.6	11.3	96.5	29.2	18.2
q_3	96.8	23.3	9.1	98.4	26.7	12	100.3	31.0	18.2
Max.	97.3	24.8	9.5	98.6	26.9	12.2	100.5	31.5	22.2

* Estatísticas considerando todos os problemas associados a cada uma das instâncias, conforme Tabela 1.

O presente artigo trouxe a proposta de um algoritmo BRKGA para o PKGCF, que foi aplicado em 148 problemas teste, derivados de 34 instâncias da literatura, variando-se os parâmetros K e M_c. Experimentos realizados com os 148 problemas e a aplicação de teste de hipótese, mostraram um percentual elevado de soluções significativamente melhores do BRKGA em relação a dois algoritmos da literatura e uma formulação de PLI; indicando que o algoritmo BRKGA é uma boa alternativa à resolução desse problema. Em trabalhos futuros, serão implementadas versões do BRKGA incorporando novos decodificadores e a inclusão de busca local baseada nos métodos *first* and *best improvement* [Resende e Ribeiro, 2016].

Referências

Billionnet, A. (2005). Different formulations for solving the heaviest k-subgraph problem. *Information Systems and Operational Research*, 43:171–186.

Brito, J., , Semaan, G., e Fadel, A. (2022). The effective brkga algorithm for the k-medoids clustering problem. *RAIRO Operations Research*, 56(4):3137–3153.

Bruglieri, M., Ehrgott, M., Hamacher, H., e Maffioli, F. (2006). An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Applied Mathematics*, 154:1344–1357.

Cavique, L. (2004). Graph-based structures for the market baskets analysis. *Investigação Operacional*, 24:233–246.

Everitt, B. S., Landau, S., Leese, M., e Stahl, D. (2011). *Cluster Analysis*. wiley, 5th edition.

Fadel, A., Ochi, L., Brito, J., e Semaan, G. (2021). Microaggregation heuristic applied to statistical disclosure control. *Information Sciences*, 548:37–55.

Festa, P. (2013). A biased random-key genetic algorithm for data clustering. *SI:BIOCOMP,Math. Biosci*, 245(1):76–85.

Ghirardi, M. (2021). *Matheuristics for the K clusters with fixed cardinality problem*. PhD thesis, Politecnico di Torino.

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852

Gonçalves, G. e Lourenço, L. (2009). A strengthened mixed-integer linear for- mulation for the k clusters problem with fixed cardinality. In *MATH'09: Proceedings of the 14th WSEAS International Conference on Applied Mathematics*, p. 229–232. World Scientific and Engineering Academy and Society (WSEAS).

Gonçalves, G. e Lourenço, L. (2019). Mathematical formulations for the k clusters with fixed cardinality problem. *Computers & Industrial Engineering*, 135:593–600.

Gonçalves, G., Lourenço, L., e Pato, M. (2010). A two local search for the k clsuters with fixed cardinality problem. In *Advances in Mathematical and Computational Methods, Proceeding of the 12th WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering (MACMESE '10, University of Algarve, Faro, Portugal)*, p. 88–92. World Scientific and Engineering Academy and Society (WSEAS).

Gonçalves, G., Lourenço, L., e Pato, M. (2012). An elitist genetic heuristic for the k-clusters with fixed cardinality problem. In *Proceedings of the 14th WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems (MAMECTIS'12), Ana Julia Viamonte (ed.), Porto, June 2012, Mathematical Models and Methods in Modern Science*, p. 160–165. Wseas Llc Staff.

Gonçalves, J. e Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17:487–5251.

Hair, J., Black, W., Babin, B., e Anderson, R. (2018). *Multivariate Data Analysis*. Cengage Learning, 8th edition.

Hansen, P. e Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical Programming*, 79:191–215.

Johnson, R. e Wichern, D. (2018). *Applied Multivariate Statistical Analysis (Classic Version)*. Pearson, 6th edition.

Negreiros, M. e Palhano, A. (2006). The capacitated centred clustering problem. *Computers and Operations Research*, 33(6):1639–1663.

Oliveira, A., R. amd Chave e Lorena, L. (2017). A comparison of two hybrid methods for constrained clustering problems. *Applied Soft Computing*, 54:256–266.

Resende, M. e Ribeiro, C. (2016). *Optimization by GRASP - Greedy Randomized Adaptive Search Procedures*. Springer.

Semaan, G. (2013). *Algoritmos para o Problema de Agrupamento Automático*. PhD thesis, Federal Fluminense University.

Triola, M. (2013). *Introdução a Estatística - Atualização da Tecnologia*. LTC, 11^a Edição.

Agradecimentos: Ao CNPq - Projeto Universal 405044/2021-6.

DOI: 10.59254/sbpo-2023-174852
http://live-proceedings.pantheonsite.io/p/174852