Heuristics Applied to Minimization of the Maximum-Diameter Clustering Problem

José André de Moura Brito, Augusto Cesar Fadel, Gustavo Silva Semaan, Flávio Marcelo Tavares Montenegro

***Abstract*— This paper introduces two heuristic algorithms for the Maximum-Diameter Clustering Problem (MDCP), based on the Biased Random-Key Genetic Algorithm (BRKGA) and the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristics. This problem consists of finding *k* clusters that minimize the largest within-cluster distance (diameter) among all clusters. The MDCP is classified as NP-hard and presents increased difficulty when attempting to determine the optimal solution for any instance. The results obtained in the experiments using 50 well-known instances indicate a good performance of proposed heuristics, that outperformed both three algorithms and an integer programming model from the literature.**

***Index Terms*—Clustering, Diameter, Metaheuristics, Integer Programming**

I. Introdução

Atualmente existe um vasto conjunto de aplicações reais que podem ser abordadas utilizando a análise de agrupamentos [1]. Tais aplicações estão associadas aos mais diversos domínios, tais como: Biologia, Engenharias, Estatística, Medicina, Computação, entre outros. Em linhas gerais, a análise de agrupamentos é uma técnica de análise multivariada que incorpora um conjunto de algoritmos, os quais têm, por finalidade, a construção de *k* grupos (clusters) a partir de uma base de dados constituída por *n* objetos (registros) com *q* variáveis. De forma a efetuar a alocação dos objetos aos grupos e avaliar a sua homogeneidade, utiliza-se uma função objetivo baseada em uma métrica, ou seja, uma medida de distância.

Face às variadas aplicações desta técnica e à complexidade envolvida na resolução dessas aplicações, quanto à obtenção de soluções de boa qualidade, encontram-se na literatura diversos algoritmos, classificados, basicamente, como não hierárquicos e hierárquicos [1]. A escolha da métrica e a definição da função objetivo impactam, diretamente, a alocação dos objetos aos grupos.

J.A.M. Brito, Escola Nacional de Ciências Estatísticas, Rio de Janeiro, RJ, Brasil (e-mail: jambrito@gmail.com).
A. Fadel, Instituto Brasileiro de Geografia e Estatística, Rio de Janeiro, RJ, Brasil (e-mail: augustofadel@gmail.com)
G.S. Semaan, Instituto do Noroeste Fluminense de Educação Superior da Universidade Federal Fluminense, Santo Antônio de Pádua, RJ, Brasil (e-mail: gustavosemaan@id.uff.br).
F.M.T. Montenegro, Escola Nacional de Ciências Estatísticas, Rio de Janeiro, RJ, Brasil (e-mail: fmtmontenegro@gmail.com).

Além disso, ao definir a função objetivo, fica definido, consequentemente, o problema de agrupamento que será abordado. Neste sentido, em [2-5] são apresentados exemplos de funções objetivo e de formulações matemáticas adotadas em alguns desses problemas. Entretanto, independentemente da métrica e da função consideradas, problemas de agrupamento são, em geral, de difícil solução computacional, conforme [3], [4-9] e [10], sendo classificados como problemas NP-difíceis. Isso implica, por sua vez, que à medida que o número de objetos a serem agrupados aumenta, mais difícil torna-se a obtenção da solução ótima global para esses problemas, seja através da aplicação de métodos exatos ou mesmo heurísticas. Neste trabalho, foi considerada, em particular, a função objetivo associada ao diâmetro, caracterizando o problema de agrupamento com diâmetro mínimo, doravante denotado por PADM. Uma revisão sistemática da literatura mostra que há poucos trabalhos que trazem a proposta de algoritmos para a resolução deste problema, sendo a maioria baseada na aplicação de procedimentos simples e que produzem, no máximo, soluções correspondentes a ótimos locais de baixa qualidade. Ainda neste sentido, até o presente momento, o algoritmo relativamente mais eficaz encontrado para o PADM foi o algoritmo GRASP proposto em [11].

Os novos algoritmos propostos neste trabalho têm como diferenciais: (i) o uso de procedimentos de construção, cruzamento e busca local que garantem a produção de ótimos de melhor qualidade (muitas vezes ótimos globais, às custas de baixo tempo computacional), quando comparados às soluções produzidas pelos demais algoritmos da literatura, em particular, o algoritmo proposto em [11] que, até o presente momento, relatava os melhores resultados; (ii) a robustez e estabilidade do algoritmo baseado no BRKGA, comprovadas por experimentos computacionais da seção IV e (iii) a proposta de um novo algoritmo que teve por base uma metaheurística nunca aplicada ao PADM. A seguir, fazemos um breve relato dos trabalhos chave encontrados durante o processo de revisão.

O trabalho de [2] apresenta uma formulação de programação inteira para o problema. Além da formulação, o autor propôs um algoritmo heurístico que produz o ótimo global quando o número de grupos *k* é igual 2. Ainda neste sentido, foi provado, em [12], que este problema é NP-difícil quando k≥3. Em [13], foi proposto um algoritmo *branch and bound* específico para o problema. Todavia, conforme relatado em [11], tal algoritmo não apresenta uma boa

performance para problemas com número de objetos a partir da ordem de centenas.

Em [14], os autores propuseram um limitante superior para o PADM a partir do estudo do algoritmo hierárquico clássico de ligação completa. Em [15], os autores adotaram o algoritmo hierárquico clássico de ligação completa para resolver o PADM em aplicações reais, mais especificamente relacionadas à psicologia, e propuseram uma metodologia para definir o número de grupos k com base na convergência das soluções obtidas, a fim de produzir soluções finais mais robustas. Em [16], é apresentado um framework geral para a resolução do PADM em que consideram duas questões. Na primeira, subconjuntos de objetos são utilizados para obter *lower bounds* para a busca de soluções ótimas do problema original, enquanto a segunda apoia-se na existência de um subconjunto de objetos da instância que possui o mesmo valor no diâmetro que o do problema original.

Em função da complexidade desse problema, e objetivando produzir soluções de boa qualidade, às expensas de baixo custo computacional, são apresentados, neste artigo, dois algoritmos heurísticos, baseados, respectivamente, nos conceitos das metaheurísticas GRASP [17][18] (*Greedy Randomized Adaptive Search Procedure*) e BRKGA *(Biased Random-Key Genetic Algorithm*) [19].

Além da introdução, este artigo está dividido em mais quatro seções. A seção dois traz uma descrição do PADM, sendo apresentada, inclusive, a formulação que foi proposta em [2] para esse problema. Complementando a seção, apresenta-se uma breve descrição dos três algoritmos da literatura considerados neste trabalho para fins de comparação com as duas heurísticas propostas, quais sejam: o algoritmo GRASP proposto por [11], o algoritmo clássico de Ligação Completa (do inglês, *Complete Linkage*) [1] e o algoritmo FPF (*Furthest Point First*) apresentado em [20]. Doravante, para fins de referência neste artigo, esses três algoritmos serão denominados, respectivamente, por: GRASPDM2, CL e FPF.

A seção três apresenta as metaheurísticas GRASP e BRKGA, seguidas da descrição dos dois algoritmos heurísticos propostos, denominados, respectivamente, GRASPDM1 e BRKGADM. Na seção quatro, são apresentadas algumas informações sobre as instâncias utilizadas, os resultados obtidos e análises realizadas a partir dos experimentos computacionais, considerando a aplicação dos cinco algoritmos e da formulação apresentada na seção dois. Analisando os resultados dos experimentos realizados, observou-se que os algoritmos propostos têm desempenho superior aos demais algoritmos da literatura, quanto à qualidade e à quantidade das soluções produzidas. Finalizando, a seção cinco traz as conclusões.

II. Descrição do Problema

No PADM, fornecidos um conjunto X formado por n objetos $X=\{x_1,x_2,...,x_i,...,x_n\}$, com q variáveis (quantitativas e/ou qualitativas), tal que $x_i=(x_{i1},x_{i2},...,x_{iq})$, e uma matriz $D=[d_{ij}]_{nxn}$ que contém as distâncias entre todos os objetos x_i e x_j de X, tomados dois a dois, e definido o número de grupos igual k, deve-se alocar os n objetos aos k grupos, denotados por $G_1,...,G_h,...,G_k$, de forma a minimizar o diâmetro em relação aos grupos.

Mais especificamente, determina-se, para cada grupo G_h $(h=1,...,k)$, a maior distância d_{ij} (entre dois objetos x_i e x_j mais afastados dentro do grupo), o que corresponde ao diâmetro do grupo G_h, denotado por:

$$z_h = max_{\forall x_i,x_j \in G_h} d_{ij} \quad (h = 1, \dots, k) \tag{1}$$

Em seguida, calcula-se o máximo de z_h $(h=1,...,k)$ obtendo-se Z, que corresponde ao maior diâmetro dentre os k grupos. Assim sendo, no PADM busca-se minimizar Z tal que

$$Z = max_{h=1,...,k} z_h = max_{h=1,...,k}(max_{\forall i<j \in G_h} d_{ij}) \tag{2}$$

Adicionalmente à minimização de Z, assim como em outros problemas clássicos de agrupamento, devem ser cumpridas as restrições clássicas (3-5), sejam elas:

$$G_1 \cup G_2 \cup .. \cup G_k = X \tag{3}$$

$$G_l \cap G_s = \varnothing \ (l<s,\ l,s=1,...,k) \tag{4}$$

$$|G_l| \geq 1 \ (l=1,...,k) \tag{5}$$

A restrição (3) garante que união dos grupos produz o conjunto de dados original X (nenhum objeto deixa de ser alocado aos grupos). A restrição (4) garante que, cada objeto $x_i \in X$ $(i=1,...,n)$ será alocado, exatamente, a um grupo e a restrição (5) garante que cada grupo terá, pelo menos, um objeto alocado. A Fig. 1 traz exemplo de duas soluções para o PADM, considerando o número de objetos igual sete (n=7) e o de grupos igual a dois (k=2). Em cada solução, a linha tracejada corresponde ao valor de Z. Assim sendo, a **Solução 2** é melhor do que a **Solução 1.**

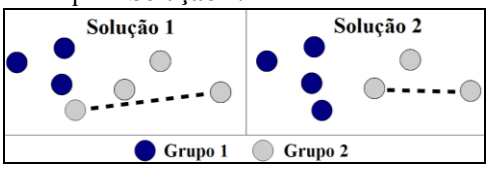


Fig. 1. Exemplo de Duas Soluções para o PADM

Em [2], são apresentadas várias formulações para problemas de agrupamento, em particular, uma formulação para o PADM. Nesta formulação, os parâmetros de entrada são o número de objetos (n), o número de grupos (k) e a matriz $D=[dij]_{nxn}$ que contém as distâncias entre os n objetos tomados dois a dois.

Além disso, define-se x_{ih} $(i=1,...,n;\ h=1,...,k)$ como a variável de decisão binária que assume valor 1 se o objeto i é alocado ao grupo G_h; e 0 caso contrário.

Minimizar Z (6)

Sujeito a

$$d_{ij}\, x_{ih} + d_{ij} x_{jh} - Z \leq d_{ij} \quad i=1,...,n\text{-}1; j=i+1,...,n; h=1,...,k \tag{7}$$

$$x_{i1} + ... + x_{ih} + ... + x_{ik} = 1 \quad i=1,...,n \tag{8}$$

$$x_{ih} \in \{0,1\};\ Z \geq 0 \quad i=1,...,n;\ h=1,...,k \tag{9}$$

A função objetivo minimizada em (6) corresponde ao maior diâmetro. A restrição (7) garante que os diâmetros dos grupos são menores ou iguais ao maior diâmetro Z. A restrição (8) garante que cada objeto i deve ser alocado a exatamente um dos grupos G_h $(h=1,...,k)$. A restrição (9) indica a

integralidade das variáveis x_{ih} e reforça que Z assume valores não negativos.

Esta formulação tem o número de variáveis igual a $(n.k+1)$ e o número de restrições igual a $k.(n^2-n)+n+1$, ou seja, pode ter um número substancial de variáveis e restrições à medida que n aumenta. Assim sendo, pensando no ótimo global, conforme comentado em [2], a aplicação de tal formulação para este problema é factível, apenas, para valores pequenos de n (ordem de dezenas).

Em [2], o autor também propôs uma heurística que produz o ótimo global quando k=2. Basicamente, em cada passo, essa heurística tenta alocar os dois objetos mais distantes a grupos distintos. A partir dessa heurística, em [11] os autores propuseram um algoritmo GRASP que constrói uma solução inicial a partir do conjunto X, aplicando, recursivamente, a heurística proposta em [2], de forma a produzir k' grupos, sendo $k'=2.k$. No primeiro passo, a heurística é aplicada dividindo os n objetos da base de dados em dois grupos. Nos passos seguintes, a heurística é aplicada no grupo com maior diâmetro, e assim sucessivamente, até que sejam produzidos k' grupos.

Em seguida, a partir da solução composta pelos k' grupos, são aplicados, nesta ordem, procedimentos de construção e de busca local, de forma a produzir os k grupos finais. Em linhas gerais, o procedimento de construção efetua a união de alguns dos k' grupos até produzir k grupos. O critério de junção leva em conta o acréscimo mínimo nos diâmetros dos k grupos, e na busca local são efetuadas diversas realocações dos objetos entre os k grupos (obtidos na fase de construção), de forma a reduzir o diâmetro máximo.

O algoritmo de Ligação Completa [1] (*Complete Linkage*) ou algoritmo do diâmetro, pertence à classe dos métodos hierárquicos aglomerativos. Tem esta denominação, porque todos os objetos, em cada grupo, são conectados uns com os outros tendo por base a distância máxima. Desta forma, a distância interna do grupo se iguala ao diâmetro máximo. No passo inicial deste algoritmo, cada objeto corresponde a um grupo, sendo efetuadas, em cada passo, junções entre os objetos ou entre um objeto e grupos (definidos em passos anteriores) já formados, em função da distância máxima, definindo novos grupos. A solução deste tipo de algoritmo pode ser representada graficamente por um dendograma [1]. Em [20] os autores apresentam o algoritmo denominado FPF (*Furthest Point First*) que trabalha, selecionando do conjunto X, em cada iteração, um objeto denominado "cabeça" (centroide do grupo), que define um novo grupo. O algoritmo FPF (Algoritmo 1) tem duas etapas: Na etapa de inicialização, o algoritmo seleciona aleatoriamente de X um objeto x_i, sendo tal objeto definido como cabeça do grupo 1. Ainda nesta etapa, todos os $(n-1)$ objetos restantes são inicialmente alocados ao grupo 1. Na segunda etapa, durante $(k-1)$ iterações, são definidos os demais cabeças (centroides) dos grupos $2,...,k$ e realizadas realocações dos objetos ao grupo mais próximo, considerando a distância entre o objeto e o cabeça do grupo.

Algoritmo 1 – Algoritmo FPF

Etapa de Inicialização (conjunto X)
1: Selecione $x_i \in X$ aleatoriamente e defina x_i como cabeça de G_1
2: Adicione x_i a C (Conjunto com objetos cabeça, centroides)
3: Aloque todos os objetos ($X \backslash C$) a G_1
Etapa de Alocação
4: **Para** $j \leftarrow 2$ **até** k **Faça**
5: Selecione de $X \backslash C$ o objeto x_s mais afastado dos cabeças de C
6: Adicione x_s em C
7: Defina x_s como cabeça de G_j
8: Aloque a G_j cada objeto x_i que esteja mais próximo do objeto cabeça de G_j do que o cabeça do seu grupo atual
9: **Fim-Para**

III. Metodologia de Resolução

Esta seção traz uma breve descrição das metaheurísticas BRKGA e GRASP e dos algoritmos implementados para o PADM, a partir do estudo dessas metaheurísticas.

A. Metaheurística BRKGA

Na metaheurística BRKGA [19][21] cada solução que compõe a população (cromossomos) é representada por um vetor u (vetor de chaves aleatórias) com n valores reais, gerados segundo uma distribuição uniforme [0,1].

A população utilizada em todas gerações de um algoritmo BRKGA é constituída por um conjunto de p vetores de chaves aleatórias. Além disso, em cada geração do BRKGA aplica-se, em cada um desses vetores, um procedimento denominado decodificador (específico para o problema), que transforma cada vetor u em uma solução viável s para o problema de otimização para o qual a função objetivo deve ser computada.

Após a aplicação do decodificador e o cálculo da função objetivo considerando as p soluções viáveis, a população é particionada em dois conjuntos, a saber: um pequeno conjunto formado por p_e soluções elite, correspondentes às melhores soluções segundo o valor da função objetivo, e um conjunto formado por $(p-p_e)$ soluções não-elite, sendo $p_e<p-p_e$.

Para a atualização da população entre duas gerações seguidas utiliza-se uma estratégia de elitismo, ou seja, todos os p_e cromossomos do conjunto elite em uma geração g são copiados para a população da geração $g+1$, produzindo soluções viáveis cada vez melhores durante as m gerações. Os $(p-p_e)$ cromossomos que complementam a população da geração seguinte são produzidos aplicando-se procedimentos de mutação e cruzamento. Em cada geração a população é composta por: p_e cromossomos associados ao conjunto elite (geração g), p_m cromossomos mutantes e por $(p-p_e-p_m)$ cromossomos filhos, que são produzidos mediante aplicação do cruzamento uniforme [22]. Os critérios de parada são: (i) número máximo de gerações, (ii) tempo máximo de execução

ou (iii) número máximo de gerações sem melhoria no valor da função objetivo.

B. Metaheurística GRASP

A metaheurística GRASP [17][18] corresponde a um método iterativo que combina procedimentos de construção e busca local. Em linhas gerais, durante t iterações, aplica-se o procedimento de construção, seguido do procedimento de busca local. O objetivo da construção é produzir soluções viáveis s_o, que podem ser melhoradas mediante a aplicação do procedimento de busca local, produzindo soluções s^* tais que $f(s^*)<f(s_o)$ (considerando a função objetivo $f(.)$ de um problema de minimização). Ainda neste sentido, a melhor solução s^* produzida após as t iterações do GRASP será a solução adotada para o problema de otimização em questão.

C. Algoritmo BRKGA para o PADM

Considerando o algoritmo proposto para o PADM (BRKGADM), a partir do estudo do BRKGA, cada cromossomo foi definido como um vetor u com n posições (nº de objetos do problema) preenchidas com valores gerados segundo uma distribuição uniforme [0,1], sendo cada posição de u (índice) correspondente ao índice do objeto x_i.

O decodificador implementado atribui a um vetor w, os n valores de u ordenados crescentemente. Em um passo posterior, os valores de w são pesquisados em u, retornando-se as posições que esses valores ocupam em u a um terceiro vetor y. As k primeiras posições de y contêm índices associados aos objetos que serão utilizados para a formação dos k grupos iniciais $(G_1, G_2,..., G_k)$, ou seja, o índice na 1ª posição de y (objeto ao qual foi atribuído o menor valor de u) será alocado ao grupo G_1, e assim sucessivamente, com a alocação do objeto na k-ésima posição de y ao grupo G_k.

Os demais objetos, cujos índices estão entre as posições $(k+1)$ e n de y, são alocados, cada um, ao grupo G_h $(h=1,...,k)$ onde ocorrer o menor incremento do diâmetro. Mais especificamente, para cada objeto x_i $(i=k+1,...,n)$ de y, determina-se, a partir da matriz D (contendo as distâncias entre todos os objetos de X tomadas dois a dois), a sua maior distância $(d_{max}\, x_{ih}$ $h=1,...,k)$ em relação aos demais objetos já alocados a cada um dos grupos. Em seguida, aloca-se o objeto x_i ao grupo h^* tal que $h^*=argmin_h\, (d_{max}\, x_{ih})$.

A Fig.2 ilustra a definição dos grupos iniciais, considerando $n=8$, $k=2$ e um vetor u hipotético. Após a aplicação desse procedimento, considerando os vetores u e w a seguir, os dois grupos iniciais são definidos pelos objetos associados às duas primeiras posições de y, ou seja: $G_1=\{7\}$ e $G_2=\{3\}$.

A Fig. 3 ilustra (de cima para baixo e da esquerda para direita) a etapa de alocação dos $(n-k)$ objetos aos k grupos inicialmente definidos. Novamente considerando n=8 e k=2, os quadrados em cinza indicam os objetos que ainda não foram alocados a nenhum grupo, os quadrados em preto e branco indicam os objetos já alocados aos grupos 1 e 2, respectivamente, o quadrado hachurado corresponde ao próximo objeto x_i (de acordo com ordem da decodificação de y) que será alocado a um dos grupos e as setas tracejadas indicam, para G_h, qual a maior distância de x_i ao grupo. O segmento de reta cinza corresponde ao diâmetro máximo Z. No caso do PADM, o cruzamento possibilita a troca dos k primeiros objetos que definirão, inicialmente, os grupos, além de alterar a ordem de alocação dos $(n-k)$ objetos aos grupos $G_1, G_2,..., G_k$.

Índices (número dos objetos)

Vetor	1	2	3	4	5	6	7	8
u	0.34	0.71	0.12	0.28	0.83	0.69	0.02	0.54
w	0.02	0.12	0.28	0.34	0.54	0.69	0.71	0.83
y	7	3	4	1	8	6	2	5

Fig. 2. Procedimento de aplicação do decodificador

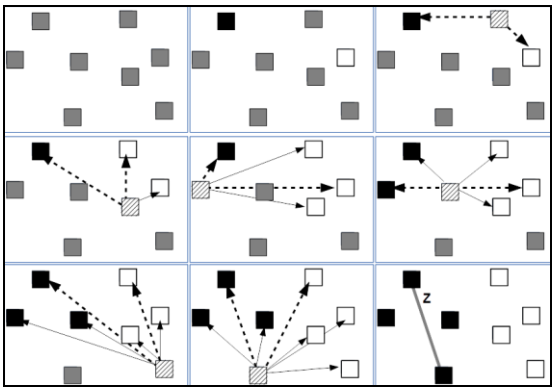


Fig. 3. Etapa de Alocação dos $(n-k)$ objetos aos grupos

D. Algoritmo GRASP para o PADM

No que concerne ao GRASP proposto para o PADM, o procedimento de construção foi desenvolvido para operar em duas etapas, quais sejam: **(1)** são selecionados k objetos dentre os n objetos da base de dados (conjunto X), sendo cada um desses objetos alocados a um grupo G_h $(h=1,...,k)$. Doravante, denotaremos cada um desses objetos como centroide do grupo. A ideia é alocar os k objetos mais afastados, entre si, a grupos distintos. **(2)** Alocar os $(n-k)$ objetos restantes aos k grupos a partir da definição de uma lista restrita de candidatos.

Para efetuar a alocação dos k objetos, definindo-se os centroides dos grupos, toma-se, inicialmente por base, um vetor $d=(d_1,...,d_i,...,d_n)$, onde cada componente d_i de d contém a maior distância do i-ésimo objeto x_i em relação aos demais $(n-1)$ objetos de X . Em seguida, cria-se um vetor b que contém os elementos do vetor d ordenados decrescentemente. Finalmente, os valores de b são pesquisados em d, retornando-se, em um vetor a, as posições que tais valores ocupam em b. O vetor a, por sua vez, será utilizado como dado de entrada pelo procedimento de construção em todas t iterações do algoritmo GRASPMD. Por exemplo, supondo n=8 e que os vetores d e b produzam o vetor a=(2,5,1,7,4,3,8,6), temos que 2 é o objeto com a maior distância em relação aos demais objetos de X, 5 é o objeto com a segunda maior distância em relação aos demais, e assim sucessivamente.

Considerando os valores nas n' primeiras posições de a $(n'=0.7n)$, define-se o vetor $s=(a_1,a_2,....,a_{n'})$. Em cada iteração, o procedimento de construção seleciona k elementos do vetor s, sendo cada um desses elementos associados a um

grupo G_h *(h=1,...,k)*.

O algoritmo 2 tem os passos associados à primeira etapa de aplicação do procedimento de construção, considerando o vetor *s* e a matriz *D* (com distâncias d_{ij} entre todos os objetos x_i e $x_j \in X$).

Algoritmo 2 – GRASP – 1ª Etapa de Construção(*D,s,n'*)

```
1: G[1]←selecione(s,1)
2: s←s\G[1]
3: n'←n'-1; h←1
4: Enquanto (h<k)  Faça
5:    M←0
6:    Para i←1 Até n' Faça
7:       Para j ←1 Até h Faça
8:          Se (d[s[i]][G[j]] > M) Então
9:             w←i;
10:            M← d[s[i]][G[j]]
11:         Fim-se
12:      Fim-Para
13:   Fim-Para
14:   h←h+1
15:   G[h] ← s[w]; n'←n'-1; s←s\G[h]
16: Fim-Enquanto
17: Retorna(G)
```

Na linha 1 deste algoritmo, é selecionado de s, aleatoriamente, o objeto (1º centroide) que definirá o primeiro grupo (G_1) e, nas linhas 2 e 3, ocorre a atualização do vetor *s* e do seu número de elementos, excluindo o objeto selecionado na linha 1. Entre as linhas 5 e 15, ocorre a seleção de *k-1* objetos de *s* que serão os centroides dos grupos G_2, G_3,...,G_k. Assim sendo, na primeira iteração, considerando as linhas 6 até 13, o próximo objeto a ser escolhido de *s*, e que será utilizado para definir o grupo 2, é aquele que está mais distante do centroide do grupo G_1. Após a execução dessas linhas, temos a atualização (linha 14) do número de grupos e a definição do grupo G_2 (linha 15). Generalizando, na *h*-ésima iteração, considerando as linhas 5 a 15, define-se o objeto de *s* que será alocado ao *h*-ésimo grupo, considerando a condição explicitada na linha 8.

Definidos os *k* centroides dos grupos, os *n-k* objetos restantes em *X* são alocados aos grupos $G_1,...,G_k$, utilizando o algoritmo 3. Na linha 1 do algoritmo 3, a lista de candidatos (*LC*) é definida, inicialmente, com todos os objetos de *X*, excluindo-se os *k* objetos utilizados como centroides dos grupos G_h (*h=1,...,k*). Em cada iteração desse algoritmo (linhas 4 a 22) um objeto da *LC* é alocado ao grupo G_h que esteja mais próximo (menor distância). Mais especificamente, entre as linhas 6 e 15, determina-se a distância máxima de cada objeto $x \in LC$ a cada um dos grupos G_h, ou seja, dentre os objetos já alocados a cada um dos grupos, verifica-se qual objeto está mais afastado de *x*.

Assim sendo, após a execução das linhas 9-11, o vetor *dmax* tem a maior distância de *x* a cada um dos grupos G_h. Na linha 13, associa-se a *g[t]* o menor valor de *dmax*, que corresponde ao menor incremento que pode ser observado ao alocar *x* ao grupo G_h. Na linha 14, *H[x]* armazena o número do grupo ao qual *x* deve ser alocado. Uma vez realizados os passos entre 6 e 15, no passo 16 determina-se o menor e o maior acréscimos associados à alocação de objetos da *LC* na solução em construção. No passo 17, define-se a *LRC* a partir de um subconjunto de objetos da *LC* e, no passo 18, é selecionado um objeto da *LRC* para ser alocado a um dos grupos, atualizando-se o diâmetro do grupo e a lista de candidatos *LC* (passos 19-21). Após a aplicação do algoritmo 3, produz-se uma solução com os grupos (alocação dos *n* objetos de *X*) e o diâmetro máximo.

Algoritmo 3 – GRASP – 2ª Etapa de Construção(*G,X,n*)

```
1: LC←X\{∪Gh}
2: i←0
3: Z[h] ←∞ (h=1,...k)
4: Enquanto (i<n-k) Faça
5:    i←i+1
6:    Para t←1 Até (n-k-i+1) Faça
7:       max ← -∞
8:       x←LC[t]
9:       Para h←1 Até k Faça
10:         dmax[h] ←maximo ∀y∈Gh(d[x][y])
11:      Fim-Para
12:      h*← argmin h (dmax[h]).
13:      g[x] ←dmax[h*]
14:      H[x] ← h*
15:   Fim-Para
16:   gmin←minimo(g); gmax←maximo(g)
17:   LRC←{t∈LC|g[t] ⩽ gmin+0.2(gmax - gmin)}
18:   x←selecione(LRC,1)
19:   Se (g[x]>Z[H[x]]) Então Z[H[x]]← g[x]
20:   G[H[x]]←G[H[x]] ∪x
21:   LC←LC-x
22: Fim-Enquanto
23: Retorna(G)
```

Algoritmo 4 – GRASP – Busca Local(*G,Z*)

```
1: reducao← VERDADEIRO
2: Enquanto (redução= VERDADEIRO) Faça
3:    reducao←FALSO
4:    w←argmax h=1,...,k (Z[h])
5:    ZMAX←Z[w]
6:    Atribua a yi e yj, respectivamente os pontos xi , xj ∈ G[w]
      associados ao diâmetro ZMAX de G[w]
7:    Para h∈{1,...,k}\w Faça
8:       Se (alocação de yi ou yj a G[h] produz Z[h] < ZMAX) Então
9:          ZMAX←Z[h]; h*←h
10:         xk ← (yi ou yj) que produz maior decréscimo em Z[h*]
11:         reducao←VERDADEIRO
12:      Fim-se
13:   Fim-Para
14:   Se (reducao=VERDADEIRO) Então
15:      Alocar xk a G[h*] e atualizar o grupo G[w]
16:      Atualizar o vetor Z
17:   Fim-se
18: Fim-Enquanto
```

Com o objetivo de reduzir o diâmetro máximo, foi aplicada a busca local apresentada no algoritmo 4. Em linhas gerais, este algoritmo determina, em cada iteração, o grupo atual com maior diâmetro e verifica se a alocação de um dos dois objetos x_i e x_j desse grupo (que determinam esse diâmetro maior d_{ij}) a um dos *k-1* grupos restantes produz um novo conjunto de grupos G_h *(h=1,...,k),* tal que o maior diâmetro dentre os diâmetros dos *k* grupos seja inferior ao maior diâmetro atual. Enquanto ocorrer (linha 2) redução no

diâmetro, novas iterações são realizadas, identificando-se o grupo com maior diâmetro e avaliando-se a alocação dos objetos associados a este grupo a outros grupos. Nas linhas 4, 5 e 6, determina-se o número do grupo G_w atual com maior diâmetro e os dois objetos que determinam este diâmetro. Entre as linhas 7-13, é avaliado se a alocação de um desses objetos a outros grupos produz redução no diâmetro máximo. A alocação que produz a maior redução no diâmetro é considerada nos passos 14-17, e efetua-se a atualização dos grupos que tiveram realocação de objetos, mais especificamente, os grupos G_w e G_{h^*}

IV. Resultados Computacionais

A presente seção traz resultados relativos à aplicação dos algoritmos BRKGADM e GRASPDM1, descritos na seção III, da formulação da seção II e dos algoritmos CL e FPF.

Em relação ao algoritmo GRASPDM2 (correspondente ao algoritmo GRASP-II proposto em [11]), os seus autores não disponibilizaram a sua versão implementada. Assim sendo, conforme explicado mais à frente, foi considerado um experimento específico, onde os resultados obtidos para um conjunto de instâncias utilizadas em [11] foram comparados com os resultados produzidos pelos algoritmos BRKGADM, GRASPDM1, CL e FPF. Os algoritmos BRKGADM, GRASPDM1 e FPF foram implementados em linguagem R (www.r-project.org), o algoritmo CL está disponível em R, no pacote *cluster* (função hclust) e a formulação foi implementada pelos autores utilizando a versão 14.0 do *solver* de otimização LINGO (https://www.lindo.com).

Todos os experimentos relacionados aos algoritmos e à formulação foram realizados em um computador com 24GB de memória RAM e dotado de processador de 3.40 GHz (*I7*).

De forma a avaliar os algoritmos BRKGADM e GRASPDM1 frente aos três algoritmos da literatura e à formulação, no que diz respeito à qualidade das soluções produzidas, foram considerados dois experimentos, a saber: **(Experimento - I)** os algoritmos BRKGADM, GRASPDM1, FPF, CL e a formulação foram aplicados nas 29 instâncias listadas na Tabela I. Esta tabela traz o nome da instância, seu número de objetos (n) e a origem (O). Quanto à origem, essas instâncias são distribuídas em 8 grupos, a saber: (1) literatura, sendo citadas e utilizadas, por exemplo, em [4] e [6]; (2) site do IBGE *(www.ibge.gov.br)* e da universidade da Califórnia (*archive.ics.uci.edu/ml/*); (3) gerada artificialmente no *software* R; (4) disponível no pacote MASS no R; (5) utilizadas em [23]; (6) e (7) disponíveis, respectivamente, nos pacotes *cluster.dataset* e *dataset* do R e (8) disponível em https://cs.joensuu.fi/sipu/datasets/;

(Experimento - II) os algoritmos BRKGADM, GRASPDM1, FPF e CL foram aplicados em 21 instâncias listadas na tabela II. Esta tabela traz o nome da instância (formato *xxx_y*, sendo *xxx* correspondente ao número de objetos, variando entre 300 e 1.000, e y correspondente ao número de grupos (k) que foi considerado ao aplicar o algoritmo GRASPDM2 nessas instâncias). Estas 21 instâncias, gentilmente cedidas pelos autores de [11], foram geradas artificialmente de acordo com procedimento descrito em [11]. Os resultados produzidos pelos algoritmos BRKGADM, GRASPDM1, FPF e CL, para estas instâncias, foram comparados com os resultados produzidos pelo algoritmo GRASPDM2, apresentados na tabela 1 (table 1, pg 518), disponível em [11]. Em todas as instâncias das Tabelas I e II, foi considerada a distância euclidiana para a construção da matriz de distâncias D (vide seção II), que é utilizada no cálculo da função objetivo do problema.

TABELA I
INSTÂNCIAS UTILIZADAS – EXPERIMENTO I

Instância	n	O	Instância	n	O
200DATA (2)*	200	1	idh2013 (1)	187	2
2face (2)	200	1	indochina combat (4)	72	6
Aggregation (2)	788	8	iris (4)	150	1
Airqualit (2)	153	7	maronna (2)	200	1
BreastB (1213)	49	5	moreshapes (2)	489	1
broken-ring (2)	800	1	normal300 (2)	300	3
Chart (60)	600	1	numbers2 (2)	540	1
concrete_data (9)	1.030	2	Parkinsons (23)	195	2
cpu (8)	209	4	pib100 (1)	100	2
DBLCA (661)	141	5	ruspini (2)	75	1
face (2)	296	1	vowel2 (2)	528	1
Faithful (2)	272	7	wholesale (6)	440	2
Forestfires (7)	517	2	wine (13)	178	1
gauss9 (2)	900	1	yeast (7)	1.484	1
Haberman (4)	306	2			

*Valores entre parênteses após nome da instância indicam o número de atributos

TABELA II
INSTÂNCIAS UTILIZADAS – EXPERIMENTO II

Instâncias						
400 3	500 3	600 3	700 3	800 3	900 3	1000 3
400 4	500 4	600 4	700 4	800 4	900 4	1000 4
400_5	500_5	600_5	700_5	800_5	900_5	1000_5

Em relação ao experimento I, os resultados do BRKGADM e GRASPDM1 foram comparados com os da formulação (teve tempo máximo de execução de 2 horas) e com os dos algoritmos CL e FPF, considerando as 29 instâncias e k=2,3,4,5. Ou seja, foram produzidas 116 soluções (nº de instâncias x nº de grupos). No experimento II, adotou-se, para cada uma das 21 instâncias, o mesmo valor de k apresentado na Tabela II.

Os parâmetros utilizados do BRKGADM e GRASPDM1 foram definidos a partir de experimentos preliminares em um subconjunto de 5 instâncias (dentre as 29), sendo testadas várias combinações dos seus parâmetros. Assim sendo, os valores finais utilizados no algoritmo BRKGADM foram: p=100 (tamanho da população); p_m=20; p_e=25 (número de cromossomos mutantes e elite); ρ_e=0.7 (probabilidade de cruzamento) e m=50 (número de gerações). De igual forma, após a realização de experimentos preliminares em relação ao GRASPDM1, o número de iterações t (aplicação da construção e busca local) foi definido igual a 75 e o valor de α=0.2.

Para fins de análise e construção dos gráficos e tabelas a seguir, foi considerada como solução "vencedora" a melhor solução produzida (ótimo local ou global) por, pelos menos, um dos algoritmos ou pela formulação.

Uma análise da Fig. 4 mostra que o BRKGADM teve uma performance superior em relação aos demais algoritmos e à formulação, seguido pelo algoritmo GRASPDM1. Mais

especificamente, os percentuais de soluções vencedoras do BRKGADM para *k*=2, 3, 4 e 5 foram, respectivamente, de: 100%, 97%, 97% e 83%. No caso do GRASPDM1, estes percentuais foram de: 100%, 86%, 72% e 45%.

A tabela III traz os *gaps* médios (%) para cada um dos algoritmos e número de grupos, tomando por base as 29 instâncias e a expressão: $(z_{ij} - zv_j)/ zv_j$, sendo zv_j o valor do menor diâmetro (função objetivo) associado à solução vencedora em relação à *j*-ésima instância e z_{ij} o valor do menor diâmetro (fobj) produzido pelo *i*-ésimo algoritmo (ou formulação) para *j*-ésima instância. A partir desta tabela, observa-se que BRKGADM teve *gaps* médios bem pequenos, inferiores a 0,5%, independentemente do número de grupos.

TABELA III
EXPERIMENTO I – GAPS MÉDIOS EM RELAÇÃO À SOLUÇÃO VENCEDORA

	k=2	k=3	k=4	k=5
Formulação	0,00%	7,9%	27,0%	39,7%
BRKGADM	0,00%	0,4%	0,0%	0,2%
GRASPDM1	0,00%	1,6%	1,8%	8,8%
FPF	2,5%	9,1%	15,0%	32,0%
CL	6,9%	14,0%	25,4%	35,7%

Concluindo a análise dos resultados do Experimento I, a tabela IV traz o percentual de soluções ótimas globais produzidas dentro do intervalo de 2 horas pela formulação e pelos dois melhores algoritmos (BRKGADM e GRASPMDM1). Em particular, o BRKGADM produziu o mesmo percentual de ótimos globais que a formulação, a menos do número de grupos igual a 4. O tempo de processamento do BRKGADM variou entre 2 segundos (menor instância) e 1,1 minutos (maior instância). No caso do algoritmo GRASPDM1, os tempos variaram entre 3 segundos e 5 minutos. A Tabela V traz, para as instâncias reportadas na Tabela II, os percentuais de soluções vencedoras e os *gaps* médios, obtidos com a aplicação dos algoritmos BRKGADM, GRASPDM1, GRASPDM2, FPF e CL. Novamente, pode-se observar que o BRKGADM teve performance superior aos demais algoritmos quanto ao total de soluções vencedoras e *gaps* médios. Destaca-se, também, uma vantagem do algoritmo GRASPDM1 em relação ao GRASPM2, tanto em relação ao percentual de soluções vencedoras quanto ao *gap* médio.

De forma a avaliar a robustez das soluções produzidas pelo algoritmo BRKGADM, o algoritmo com melhores soluções, realizou-se um experimento com 7 instâncias dos experimentos I (k=3,4,5) e II. O BRKGADM foi aplicado 100 vezes nestas instâncias e, a partir das 100 soluções (valores de *Z*) produzidas, foram calculadas as estatísticas resumo apresentadas na tabela VI. Nesta tabela, a partir da coluna três, temos os valores: mínimo (*min*), médio (*med*) e máximo (*max*) em relação a *Z*, além do coeficiente de variação das soluções.

TABELA IV
EXPERIMENTO I – PERCENTUAIS DE ÓTIMOS GLOBAIS

	k=2	k=3	k=4	k=5
Formulação	89,7%	62,1%	37,9%	13,8%
BRKGADM	89,7%	62,1%	34,5%	13,8%
GRASPDM1	89,7%	48,3%	31,0%	6,9%

TABELA V
EXPERIMENTO II
PERCENTUAIS DE SOLUÇÕES VENCEDORAS E GAPS MÉDIOS

Algoritmo	%Vencedoras	*Gap*
BRKGADM	90,5%	0,5%
GRASPDM1	81,0%	1,1%
GRASPDM2	66,7%	4,3%
FPF	0,0%	29,5%
CL	9,5%	20,0%

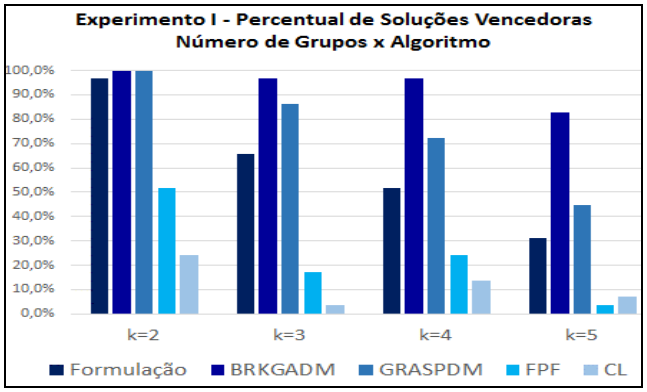


Fig. 4. Gráfico com os percentuais de soluções vencedoras.

TABELA VI
ESTATÍSTICAS RESUMO DAS 100 EXECUÇÕES DO BRKGADM

Instância	k	min	med	max	cv%
DBLCA	3	51,968	51,968	51,968	0,00
DBLCA	4	49,035	49,035	49,035	0,00
DBLCA	5	47,848	47,848	47,848	0,00
maronna	3	3,058	3,058	3,058	0,00
maronna	4	1,988	1,988	1,988	0,00
maronna	5	1,859	1,859	1,859	0,00
moreshapes	3	2,435	2,435	2,435	0,00
moreshapes	4	2,282	2,283	2,312	0,14
moreshapes	5	1,555	1,555	1,555	0,00
Haberman	3	5,128	5,128	5,128	0,00
Haberman	4	4,427	4,448	4,554	0,71
Haberman	5	4,182	4,182	4,182	0,00
broken-ring	3	2,908	2,908	2,908	0,00
broken-ring	4	2,119	2,119	2,119	0,00
broken-ring	5	1,924	1,932	1,935	0,27
C3600	4	0,545	0,545	0,545	0,00
C3900	3	0,104	0,104	0,107	0,64

Em todos os casos, o valor mínimo obtido para Z (*min*) foi igual à solução produzida pelo algoritmo BRKGADM nos experimentos I e II. Além disso, em 13 dos 17 casos, o coeficiente de variação (*cv*) foi igual a zero e, nos demais casos, foi inferior a 1%. Isso significa que, na maioria das vezes, o algoritmo produz a mesma solução e que esta corresponde à melhor solução, uma vez que a solução média, na maioria dos casos, foi igual à solução mínima.

V. CONCLUSÕES

O presente artigo trouxe a proposta de dois algoritmos heurísticos para o PADM baseados nas metaheurísticas GRASP e BRKGA. Foram realizados experimentos computacionais com esses algoritmos e mais três algoritmos de literatura, além de uma formulação de programação inteira em dois conjuntos de instâncias de porte variado quanto ao número de objetos. Tendo como critérios de avaliação o número total de melhores soluções e o *gap* médio, o algoritmo BRKGADM foi o que teve a melhor performance frente aos demais algoritmos, seguido pelo GRASPDM. Estes

algoritmos produziram, também, soluções melhores que o algoritmo GRASP proposto em [11]. Além disso, no experimento adicional com o BRKGADM, para avaliar a robustez, observou-se que o algoritmo foi robusto e produziu soluções de boa qualidade. Em trabalhos futuros, pretende-se desenvolver novos decodificadores para o BRKGADM e novos procedimentos de construção e busca local para o GRASPDM1, bem como avaliar a aplicação desses algoritmos em mais instâncias.

REFERÊNCIAS

[1] J.F, Hair, W.C, Babin, J.B, Anderson, R.E. and Black, W.C., *Multivariate Data Analysis*, 8th Edition, 2018.

[2] M.R., Rao, "Cluster Analysis and Mathematical Programming," Journal of American Statistical Association, vol. 66, pp. 622-626, 1971.

[3] P., Hansen and B., Jaumard, "Cluster Analysis and Mathematical Programming," Mathematical Programming, vol. 79, pp. 191-215, 1997.

[4] M.D. Cruz, "O Problema de Clusterização Automática, " Tese de doutorado, Coppe/UFRJ, Rio Janeiro, Brasil, 2010.

[5] J.A.M., Brito, G.S., Semaan e L.R., Brito, "Resolução do Problema dos k-medoids Via Algoritmo Genético de Chaves Aleatórias Viciadas," Revista Pesquisa Naval, vol. 27, pp.126-142, 2015.

[6] G.S., Semaan, "Algoritmos para o Problema de Agrupamento Automático," Tese de doutorado, IC/UFF, RJ, Brasil, 2013.

[7] G.S., Semaan, A.C., Fadel, J.A.M., Brito e L.S.Ochi, "A Hybrid Heuristic with Hopkins Statistic for the Automatic Clustering Problem," IEEE Latin America Transactions, vol. 17, no. 1, 2019.

[8] Alexeis Joel Ochoa Reyes, A.J.O., Arturo Orellana Garcia, A.O., Mui, Y.L. System for Processing and Analysis of Information Using Clustering Technique. IEEE Latin America Transactions. VOL. 12, NO. 2, 2015.

[9] C. D. Guerrero, D. Salcedo and, H. Lamos "A Clustering Approach to Reduce the Available Band width Estimation Error" IEEE LATIN AMERICA TRANSACTIONS, VOL. 11, NO. 3, 2013.

[10] Thomas, J.C.R., Penãs, M.S., Cofre M.M, Carralero, N.D. "Performance Analysis of Clustering Internal Validation Indexes with Asymmetric Clusters" IEEE LATIN AMERICA TRANSACTIONS, VOL. 17, NO. 5, 2019.

[11] J.A., Fiorucci, F.M.B., Toledo and M.C.V., Nascimento, "Heuristics for minimizing the maximum within-clusters distance", Pesquisa Operacional, vol.32, no. 3, pp: 497-522, 2012.

[12] P. Hansen and M., Delattre, "Complete-link cluster analysis by graph coloring," Journal of the American Statistical Association, vol. 73, pp. 397-403, 1978.

[13] M.J., Brusco and S. Stahl, "Branch and Bound applications in combinatorial data analysis*,"* New York, Springer-Verlag, 2005.

[14] Großwendt, A., Röglin, H. Improved Analysis of Complete-Linkage Clustering, Algorithmica, vol **78**, pp. 1131–1150, 2017.

[15] M. Brusco, D. Steinley. "Model selection for minimum ‐ diameter partitioning" British Journal of Mathematical and Statistical Psychology, vol. 67, pp. 471-495, 2014., pp. 471-495, 2014.

[16] D. Aloise and Contardo, C. (2016). A New Global Optimization Algorithm for Diameter Minimization Clustering. Proceedings of Global Optimization Workshop (GOW16), Portugal.

[17] M.G.C., Resende and C.C., Ribeiro, "Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*,"* 2nd ed., Springer, 2016.

[18] M.G.C., Resende and C.C. Ribeiro. GRASP: Greedy Randomized Adaptive Search Procedure: Advances and extensions. Handbook of 3rd edition, M. Gendreau and J.-Y. Potvin, Eds., Springer, pp. 169-220*, 2019.*

[19] J.R., Gonçalves and M.G.C., Resende, "Biased random-key genetic algorithms for combinatorial optimization," Journal of Heuristics, vol. 17, pp: 487-525, 2011.

[20] Gonzalez, "Clustering to minimize the maximum intercluster distance," Theoretical Computer Science, vol. 38, pp: 293-306, 1985.

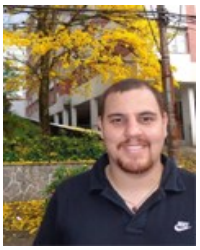
[21] M.G.C., Resende, "Introdução aos Algoritmos Genéticos de Chaves Aleatórias Viciadas," Anais do XLVSPPO, pp: 3680-3691, 2013.

[22] W.M., Spears and K.A., Dejong, "On the virtues of parameterized uniform crossover," in Proceedings of Fourth International Conference on Genetic Algorithms, pp: 230-236, 1991.

[23] M.C.V., Nascimento, F.M.B., Toledo e A.C.P.L.F, de Carvalho,"Investigation of a new GRASP- based clustering algorithm applied to biological data," Computers & Operations Research, vol. 37, pp:1381-1388, 2010.

José André de Moura Brito tem bacharelado em Matemática pela UFRJ (1997), Mestrado e Doutorado em Otimização pela COPPE/UFRJ (1999 e 2004) e Pós-Doutorado em Otimização na UFF (2008). É professor da Escola Nacional de Ciências Estatísticas (ENCE/IBGE).

Augusto Cesar Fadel é bacharel em Estatística pela Escola Nacional de Ciências Estatísticas (ENCE) e tem mestrado em Ciência da Computação na UFF. Atua como estatístico no Instituto Brasileiro de Geografia e Estatística (IBGE), onde desenvolve atividades relacionadas a controle estatístico de sigilo e uso de big data.



Gustavo Silva Semaan é Professor da Universidade Federal Fluminense (UFF). Doutor e Mestre pelo Instituto de Computação da UFF. Bacharel em Sistemas de Informação pela Faculdade Metodista Granbery.

Flávio Marcelo Tavares Montenegro é Professor da Escola Nacional de Ciências Estatísticas (ENCE). Tem mestrado em Física pelo Centro Brasileiro de Pesquisas Físicas (1997) e doutorado em Engenharia de Sistemas e Computação pela COPPE/UFRJ (2001).