BRKGA – UM PACOTE DO R PARA PROBLEMAS DE OTIMIZAÇÃO

José André de M. Brito
Escola Nacional de Ciências Estatísticas – ENCE/IBGE
Rua André Cavalcanti, 106, Sala 503-A, Santa Teresa, Centro, Rio de Janeiro/RJ
jambrito@@gmail.com

Augusto Fadel
Instituto Brasileiro de Geografia e Estatística - IBGE
Avenida República do Chile, 500, 8º andar – Centro – Rio de Janeiro-RJ
augustofadel@gmail.com

Gustavo S. Semaan
Instituto do Noroeste Fluminense de Educação Superior (INFES) - UFF
Av. João Jasbick, s/nº - Aeroporto - Santo Antônio de Pádua -RJ
gustavosemaan@gmail.com

RESUMO

Nos dias atuais, pesquisadores são, frequentemente, confrontados com inúmeras aplicações reais que remetem a problemas de otimização de alta complexidade. Face à complexidade desses problemas, a aplicação de métodos exatos ou de enumeração exaustiva torna-se inviável. De forma contornar essa dificuldade, pode ser considerada a aplicação de algoritmos implementados a partir de heurísticas ou metaheurísticas. Neste sentido, o presente trabalho traz a apresentação de um pacote do R que contém o algoritmo BRKGA (função), que pode ser utilizado, de forma simples, em diversos problemas de otimização. Além dos argumentos comuns de um BRKGA, para utilizar o pacote, o usuário precisa, apenas, definir a função e um decodificador para o seu problema. São apresentados alguns exemplos do uso desse pacote, considerando funções objetivo e decodificadores para vários problemas de otimização, como, por exemplo, o PCV e o problema de agrupamento.

Palavra-chave: Otimização; Metaheurísticas; Pacote; Linguagem R; BRKGA.

ABSTRACT

Nowadays, researchers are often confronted with numerous applications that address problems of high computational complexity. Given the complexity of these problems, the application of exact or exhaustive enumeration methods becomes impracticable. In order to overcome this difficulty, it can be considered the application of algorithms implemented from heuristics or metaheuristics. This work brings R package that contains the BRKGA metaheuristic (function), which can be used simply in several optimization problems. Besides the common arguments of a BRKGA, to use the package, the user just needs to define the function and a decoder for his problem. Two examples of using this package are presented, considering objective functions and decoders for PCV and clustering problem.

Keywords: Optimization; Metaheuristics; Package; R language; BRKGA.

1. INTRODUÇÃO

Quase que diariamente, pesquisadores das mais variadas áreas como, por exemplo, Pesquisa Operacional, Engenharia e Estatística, têm que trabalhar com aplicações reais que remetem a problemas de otimização de alta complexidade computacional, como, por exemplo: problemas de agrupamento [1][2][3], de roteamento [4][5], de localização [6][7] e de amostragem [8].

Em geral, face à complexidade desses problemas, a utilização de formulações matemáticas ou de métodos enumeração exaustiva, torna-se impraticável, à medida que a dimensão do problema (instância) aumenta. Mais especificamente, a resolução de formulações mediante a aplicação de métodos como *Branch and Bound* [9] é impactada, fortemente, pelo substancial número de variáveis e restrições do problema. Ainda neste sentido, mesmo utilizando *solvers* como LINGO, CPLEX, etc, que possuem rotinas sofisticadas para resolução de problemas de programação linear e inteira, e disponibilizando um razoável tempo computacional (da ordem de horas ou dias) para a resolução da formulação, podem ser produzidas, apenas, soluções correspondentes a ótimos locais de baixa qualidade. Em relação à aplicação de um método de enumeração exaustiva, embora, em geral, o conjunto de soluções viáveis seja finito, a sua cardinalidade pode ser de ordem exponencial ou fatorial, o que novamente torna inviável a aplicação desse tipo de método.

Normalmente, de forma contornar essa dificuldade, pode-se considerar a aplicação de algoritmos implementados a partir do estudo e utilização de heurísticas e metaheurísticas [10]. Tais algoritmos produzem soluções de boa qualidade às expensas de um baixo tempo computacional, quando comparado ao tempo de resolução de uma formulação ou da aplicação de um método de enumeração.

Quanto às diversas metaheurísticas disponíveis na literatura, uma possível alternativa consiste na implementação de algoritmos baseados na metaheurística algoritmos genéticos de chaves aleatórias viciadas (*Biased Random Key Genetic Algorithm* - BRKGA), que tem sido aplicada, com êxito, em diversos problemas de otimização [11].

De forma a disponibilizar para toda a comunidade científica uma ferramenta que implemente essa metaheurística, e que seja de fácil adaptação e aplicação a diversos problemas de otimização, o presente trabalho traz a proposta de um pacote desenvolvido em linguagem R (https://www.r-project.org/). Tal pacote contém um conjunto de funções que implementam o algoritmo associado à metaheurística BRKGA. Para utilização do pacote, são necessários conhecimentos básicos do usuário (pesquisador, estudante etc) quanto à metaheurística BRKGA e ao R, que é uma linguagem que, a cada dia, tem mais adeptos e vem sendo amplamente utilizada por pesquisadores e analistas de dados das mais diversas áreas, para o desenvolvimento de diversas aplicações.

Além dos procedimentos e parâmetros comuns ao BRKGA, incorporados em um conjunto de funções do pacote, o usuário só precisa definir a função objetivo e um decodificador para o seu problema. Conforme será visto mais à frente, tal decodificador, quando aplicado em um vetor de chaves aleatórias, produz soluções viáveis para o problema de otimização abordado.

Além da introdução, o presente trabalho está dividido em mais três seções. Na seção dois são apresentados os conceitos da metaheurística BRKGA e o seu funcionamento. A seção três traz a descrição de um conjunto de problemas de otimização que serviram de base para ilustrar o funcionamento do pacote, como, por exemplo, o Problema do Caixeiro

Viajante (PCV) [12] e Problemas de Agrupamento [3]. A seção quatro traz uma explanação do pacote e do seu funcionamento, considerando apresentação das funções objetivo e dos decodificadores associados aos problemas de otimização descritos na seção 3. Ao final dessa seção são apresentadas algumas conclusões.

2. METAHEURÍSTICA BRKGA

A metaheurística BRKGA [11][13] tem semelhanças com um algoritmo genético tradicional [14], no sentido que, termos como população, cromossomos e genes, são comuns às duas metaheurísticas, além de alguns procedimentos como, por exemplo, o cruzamento.

Em relação à forma de representação de cada solução que compõe a população, nessa metaheurística, cada cromossomo é representado por um vetor *u* (também denominado vetor de chaves aleatórias) com n valores reais, gerados segundo uma distribuição uniforme [0,1].

A população utilizada na primeira geração de um algoritmo BRKGA é constituída por um conjunto de *p* vetores de chaves aleatórias. Além disso, em cada geração do BRKGA é aplicado, em cada um desses vetores, um procedimento denominado decodificador, que transforma cada vetor em uma solução viável para o problema de otimização para o qual a função objetivo (aptidão) deve ser computada. Assim sendo, o decodificador é específico para cada problema de otimização ao qual é aplicado o BRKGA.

Após a aplicação do decodificador, e o cálculo da função objetivo associada a cada solução viável, 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 $(p\text{-}p_e)$ soluções não-elite, sendo $p_e<p\text{-}p_e$.

De forma a atualizar a população entre duas gerações seguidas, uma nova geração de cromossomos deve ser produzida. Neste sentido, o BRKGA usa uma estratégia de elitismo, tendo em vista que todos os p_e cromossomos pertencentes ao conjunto elite em uma geração *g* são copiados para a população da geração $g+1$. A adoção dessa estratégia possibilita produzir soluções viáveis cada vez melhores durante as gerações do algoritmo. Os $(p\text{-}p_e)$ cromossomos que complementam a população da geração seguinte são produzidos aplicando procedimentos de mutação e cruzamento.

O procedimento de mutação produz cromossomos chamados de mutantes, que nada mais são do que vetores de chaves aleatórias gerados de forma equivalente aos *p* vetores da população inicial. Em cada geração um pequeno quantitativo (p_m) de vetores mutantes são introduzidos na população e, assim como os demais vetores, esses podem ser decodificados em soluções viáveis para o problema.

De forma a complementar a população na geração $g+1$, além dos cromossomos do conjunto elite e dos cromossomos mutantes, são produzidos, mediante aplicação do cruzamento uniforme [15] $(p\text{-}p_e\text{-}p_m)$ cromossomos filhos. Cada um desses cromossomos é obtido a partir do cruzamento entre um vetor de chaves aleatórias do conjunto elite e outro vetor do conjunto não elite (definidos na geração g). A Figura 1 ilustra a evolução de uma população de uma geração *g* para a geração $g+1$.

Uma vez selecionado um cromossomo c_1 do conjunto elite e um cromossomo c_2 do conjunto não elite, é gerado um vetor auxiliar $(\boldsymbol{v_a})$ com valores reais no intervalo [0,1] (de igual tamanho a c_1 e c_2). Também é definido o valor de $\rho_e>0.5$, que corresponde à probabilidade de um gene de c_1 compor o cromossomo filho c_f. Cada valor de v_a é comparado com ρ_e tal que, se o valor na *i*-ésima posição de v_a for menor ou igual a ρ_e, temos que a

i-ésima posição do cromossomo filho herda o valor da *i*-ésima posição de c_1. Em caso contrário, a *i*-ésima posição do cromossomo filho herda o valor da i-ésima posição de c_2.

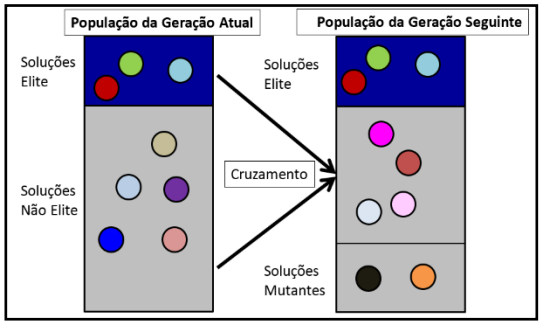


Figura 1 - Funcionamento do BRKGA entre a geração g e a geração g+1
(Baseada em figura utilizada na palestra de Maurício Resende [16]).

Uma vez construída a população da geração seguinte, isto é, quando há *p* cromossomos, distribuídos entre cromossomos elite, mutantes e filhos, aplica-se o decodificador e calcula-se novamente o valor da função objetivo para todos os novos vetores mutantes e filhos. E, dando continuidade ao processo, a população é novamente particionada em um conjunto elite e não elite para iniciar uma nova geração.

A Figura 2 ilustra todos os passos considerados para aplicação do BRKGA. Em particular, no que concerne ao critério de parada, pode ser adotado o número máximo de gerações, um tempo máximo de execução ou número máximo de gerações sem melhoria em relação ao valor da função objetivo.

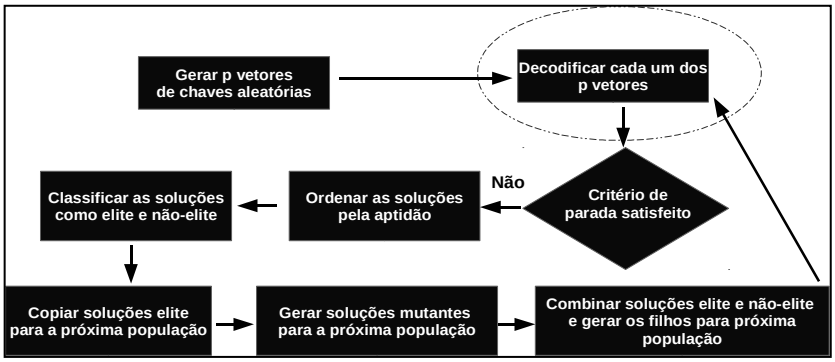


Figura 2 – Passos do BRKGA (Baseada em figura de [11])

De acordo com esta descrição, conclui-se que todos os passos do BRKGA, a menos da função objetivo e da decodificação, independem do problema de otimização abordado. Neste sentido, em [11] são apresentados alguns exemplos de problemas de otimização, comentando o procedimento decodificador implementado no algoritmo BRKGA aplicado à resolução do problema, além dos resultados de vários experimentos com o algoritmo.

3. PROBLEMAS ABORDADOS

A presente seção traz uma descrição sucinta de alguns problemas de otimização de alta complexidade computacional, quais sejam: Problema da Mochila, Problema de Agrupamento com k-Medoids, da Diversidade Máxima, do Caixeiro Viajante, de Alocação Ótima em Amostras Estratificadas, de Agrupamento com Soma Mínima e do Problema de Minimização da Função de Himmelblaus, que corresponde a uma função não linear. Além da descrição, propõe-se para cada um desses problemas, um possível decodificador que está implementado no pacote BRKGA.

3.1 PROBLEMA DA MOCHILA

É um conhecido problema de otimização [17][18] combinatória pertencente à classe NP-difícil e que vem sendo estudado há muito tempo. Segundo [19], provavelmente, tal problema foi comentado pela primeira vez por Dantzig em 1957. A importância desse problema diz respeito ao seu estreito relacionamento com um grande número de outros modelos de programação.

Há diversas aplicações reais associadas a esse problema como, por exemplo, o problema de alocação de recursos, onde há uma ou mais restrições financeiras. O problema da mochila também é estudado em teoria da complexidade, criptografia etc.

O nome é decorrente de uma situação em que é necessário colocar, em uma mochila, um conjunto de objetos (produtos) de diferentes pesos e valores (lucros, por exemplo), de forma a maximizar a soma dos valores dos objetos carregados, não ultrapassando um limite de peso, ou seja, capacidade da mochila. A partir dessa descrição, e considerando um caso particular desse problema, onde há apenas um objeto de cada tipo que pode ser carregado na mochila, esse problema pode ser formulado como:

$$\text{Maximizar } f=v_1x_1+v_2x_2+...+v_nx_n \quad (1)$$

$$\text{s.a } w_1x_1+w_2x_2+...+w_nx_n \leq C \quad (2)$$

$$x_i \in \{0,1\}\ ,\ i=1,...,n$$

Nessa formulação, x_i é uma variável 0-1 que assume valor 1 se o i-ésimo objeto é carregado na mochila. Os termos $(v_1,v_2,...,v_n)$ e $(w_1,w_2,...,w_n)$ correspondem, respectivamente, aos valores dos objetos e aos seus pesos. O termo C corresponde à capacidade da mochila.

3.1.1 DECODIFICADOR PARA PROBLEMA DA MOCHILA

Neste problema, cada vetor de chaves aleatórias u foi definido com n posições correspondentes aos n objetos que podem ser carregados na mochila. Por sua vez, o decodificador proposto aplica, em cada entrada de u, um arredondamento, isto é, round(u_i), i=1,...,n, o que implica geração de um vetor x de 0's e 1's. As posições de x com valores iguais a 1 correspondem aos objetos que serão carregados na mochila.

A fim exemplificar a aplicação desse decodificador, considere uma mochila com capacidade C igual 100, na qual podem ser carregados até 7 objetos. Além disso, considere os vetores w=(10,50,30,10,10,40,30) e v=(40,80,10,10,4,20,60). Utilizando o vetor u=(0.75,0.11,0.20,0.89,0.19,0.67,0.94), aplicando o decodificador e calculando o valor da função objetivo temos: x=(1,0,0,1,0,1,1) $\Rightarrow$ f=130 e $w_1x_1+...+w_7x_7$=90, sendo $C>90$.

Cabe observar que, alguns dos vetores x obtidos a partir da aplicação do decodificador supracitado, podem ser inviáveis, ou seja, a restrição de capacidade (Equação 2) pode não ser cumprida. Neste sentido, de forma a garantir que sejam produzidas soluções viáveis e que as soluções inviáveis sejam descartadas durante as gerações do BRKGA, introduz-se um termo de penalidade [20][21] na função objetivo da Equação 1. Isso implica resolver o seguinte problema:

$$\text{Maximizar } f-P \quad (3)$$

$$x_i \in \{0,1\}$$

$$\text{sendo } P=[máximo(0,\ w_1x_1+w_2x_2+...+w_nx_n - C)]^3$$

3.2 PROBLEMA DOS K-MEDOIDS

Definido um conjunto X formado por n objetos $X=\{x_1,x_2,...,x_n\}$, tendo cada objeto q atributos, deve-se selecionar, desse conjunto, k objetos denominados medoids, que definem o conjunto $M=\{x_{m1},x_{m2},...,\ x_{mk}\}$ [2]. Em seguida, cada medoid é alocado a um grupo G_i ($i=1,...,k$) e os $(n-k)$ objetos restantes são alocados ao grupo G_i (i=1,..., k) cujo medoid esteja mais próximo, segundo alguma medida de distância; por exemplo, a distância euclidiana.

Os objetos de M devem ser escolhidos de forma minimizar a função objetivo definida pela média (por grupo) das distâncias de todos os objetos aos seus respectivos medoids, conforme a equação 4 a seguir:

$$f = \sum_{i=1}^{k} \sum_{\forall x_j \in G_i} \frac{d_{x_{m_i} x_j}}{|G_i|} \tag{4}$$

3.2.1 DECODIFICADOR PARA PROBLEMA DOS K-MEDOIDS

Neste problema, cada vetor u tem n posições correspondentes ao número de objetos que serão agrupados. Em relação ao decodificador, definindo k como o número de grupos, em um primeiro passo, efetua-se a divisão do intervalo [0,1] em k subintervalos da forma: [0,1/k);[1/k,2/k);...; [(k-1)/k,1).

Em seguida, tomando os k intervalos e os valores de u, produz-se um vetor v de n posições com valores entre 1 e k. Mais especificamente, o valor atribuído à i-esima posição de v corresponde ao número do subintervalo no qual o i-ésimo elemento de u foi encontrado. Assim sendo, o vetor v define alocação dos n objetos aos k grupos G_i (i=1,...,k).

Finalmente, para definir os medoids, avalia-se, dentro de cada grupo, qual objeto tem a menor distância média em relação aos demais objetos do grupo, sendo esse o objeto definido como medoid do grupo.

A fim exemplificar esse decodificador, considere um conjunto com 10 objetos (n=10) e 3 grupos (k=3). Utilizando o vetor u=(0.75,0.11,0.20,0.89,0.19,0.23,0.94,0.55,0.69,0.61) e os intervalos [0,1/3);[1/3,2/3);[2/3,1] temos que v=(3,1,1,3,1,1,3,2,3,2), ou seja, G_1={2,3,5,6}, G_2={8,10} e G_3={1,4,7,9}.

Para definir o medoid do grupo 1, por exemplo, é necessário avaliar as seguintes médias: (d_{23}+d_{25}+d_{26})/3; (d_{32}+d_{35}+d_{36})/3;(d_{52}+d_{53}+d_{56})/3 e (d_{62}+d_{63}+d_{65})/3. Supondo que a menor média é a primeira, temos que o objeto 2 será o medoid do grupo. Procedimento análogo deve ser aplicado para determinar os medoids de G_2 e G_3.

3.3 PROBLEMA DA DIVERSIDADE MÁXIMA

Neste problema [22][23], deve-se selecionar, a partir de um conjunto N formado por n elementos, um subconjunto M, formado por m elementos que possuam a maior diversidade possível entre si, sendo tal diversidade avaliada a partir de uma medida distância.

Esse problema tem sido abordado em diversos trabalhos da literatura, com diferentes nomes como, por exemplo, problema da dispersão máxima [24] ou da p-dispersão [25] etc.

Formalizando o problema, considere que x_i e x_j sejam dois elementos quaisquer selecionados de N para compor M e que $x_i=(x_{i1},x_{i2},...,x_{iq})$ e $x_j=(x_{j1},x_{j2},...,x_{jq})$ são os vetores com q atributos de x_i e x_j. Em particular, adotando-se a métrica euclidiana, tem-se que a diversidade d_{ij} entre os elementos x_i e x_j, pode ser definida por:

$$d_{ij} = \sqrt{\sum_{f=1}^{q} (x_{if} - x_{jf})^2} \tag{5}$$

A partir do cálculo das distâncias entre todos os elementos do conjunto *N*, tomados dois a dois, define-se a matriz $D_{nxn}=[d_{ij}]$. Considerando essa matriz, o valor da diversidade do subconjunto *M*, em relação aos seus *m* elementos, pode ser definido por:

$$\mathrm{div}(M) = \sum_{\forall x_i, x_j \in M \subset N, i<j} d_{ij} \quad (6)$$

Assim sendo, busca-se determinar os elementos de *M* que maximizam o valor da equação (6).

3.3.1 DECODIFICADOR PARA PROBLEMA DA DIVERSIDADE MÁXIMA

Cada vetor *u* foi definido com *n* posições correspondentes ao número de elementos do conjunto *N*. Em relação ao decodificador, inicialmente calcula-se a razão r_i para cada elemento $x_i \in N$, mais especificamente, a soma das distâncias entre x_i e os demais elementos de *N* dividida pela soma das distâncias entre todos os elementos de *N* tomados dois a dois.

$$r_i = \sum_{\forall x_j \in N \text{ tq } i \neq j} d_{ij} \,/\, \sum_{\forall x_i, x_j \in N, i<j} d_{ij}\,, \quad \forall x_i \in N \quad (7)$$

Em seguida, efetua-se a multiplicação do vetor *r* (razões r_i dos elementos de *N*) pelo vetor *u*, produzindo vetor *w*. Os índices correspondentes às posições de *w* que contêm os *m* primeiros maiores valores, definem os elementos de *N* que serão selecionados para o conjunto *M*. A fim de exemplificar, suponha *n*=8, *m*=3 (cardinalidades de *N* e *M*), *u*=(0.15,0.31,0.27,0.74,0.89,0.99,0.65,0.71) e *r*=(0.31,0.18,0.77,0.52,0.59,0.81,0.14,0.45). Efetuando o produto $u.w \Rightarrow w$=(0.0465,0.0558,0.2079,0.3848,0.5251,0.8019,0.0910 0.3195).

Avaliando *w*, verificamos que os 3 maiores valores estão nas posições 4, 5 e 6, sendo esses os elementos selecionados para o conjunto $M \Rightarrow div(M)=d_{45}+d_{46}+d_{56}$

3.4 PROBLEMA DO CAIXEIRO VIAJANTE

O Problema do Caixeiro Viajante (PCV) [26] é um dos problemas mais tradicionais e mais estudados na área de Pesquisa Operacional, tendo em vista a sua complexidade, suas inúmeras aplicações práticas e sua relação com outros modelos.

Neste problema, dada uma lista de *n* cidades e uma matriz que contém as distâncias entre cada par de cidades, deve-se determinar qual é a rota de menor comprimento a ser percorrida por um caixeiro (ou vendedor), que parte de uma cidade de origem, e deve visitar todas as demais (*n-1*) cidades e retorna à cidade de origem.

O PCV é um problema pertencente à classe NP-Árdua [26], o que torna proibitiva a aplicação de um método de enumeração exaustiva, onde seria necessário avaliar toda as possíveis rotas para as *n* cidades, ou seja, um total de rota de ordem fatorial, mais especificamente, (n-1)!

3.4.1 DECODIFICADOR PARA PCV

No caso do PCV, cada vetor *u* foi definido com *n* posições correspondentes ao número de cidades que devem ser percorridas. Em relação ao decodificador, os elementos de *u* são ordenados crescentemente e atribuídos a um vetor *w*.

Em seguida, de forma a definir a rota (sequência de percurso das cidades), toma-se o elemento w_1 de *w* e verifica-se em qual posição de *u* esse elemento está. Tal posição corresponde à primeira cidade da rota. De igual forma, para determinar a segunda cidade da rota, procura-se em qual posição de *u* está o elemento de w_2 de *w*, e assim sucessivamente, até compor a rota com as *n* cidades, representada pelo vetor *x*. Em suma, cada vetor solução

obtido a partir da decodificação proposta corresponde a uma permutação na ordem de percurso das *n* cidades.

Considerando o número de cidades igual a 7 (n=7) e utilizando o vetor u=(0.11,0.45,0.63,0.98,0.02,0.58,0.78), temos: w=(0.02,0.11,0.45,0.58,0.63,0.78,0.98) $\Rightarrow$ x=(5,1,2,6,3,7,4,5).

3.5 PROBLEMA DE ALOCAÇÃO ÓTIMA EM AMOSTRAS ESTRATIFICADAS

Considere que uma população U composta por N unidades (pessoas, fazendas, empresas etc) seja dividida em H estratos E_1, E_2,..., E_H constituídos, respectivamente, por N_1, N_2,..., N_H unidades populacionais. Esses estratos [27][28] não se superpõem e, juntos, abrangem a totalidade da população de tal modo que:

$$E_h \cap E_k = \emptyset\,,\ h \neq k \tag{8}$$

$$\cup_{h=1}^{H} E_h = U \tag{9}$$

Em seguida, seleciona-se em cada um dos H estratos, uma amostra (algumas unidades) de tamanho n_h tal que $n_h \leq N_h$. A partir dessas amostras (nos estratos) são levantadas informações e produzidas estimativas (total, média etc) para um conjunto de m variáveis de pesquisa. Supondo que essas variáveis sejam denotadas, respectivamente, por y_1,...,y_j,...,y_m, a variância dessas variáveis em cada um dos estratos é definida por:

$$S_{hj}^2 = \frac{1}{(N_h-1)} \sum_{i \in E_h} (y_{ij} - \overline{Y}_{hj})^2, j = 1,\dots,m, h = 1,\dots,H \tag{10}$$

sendo y_{ij} correspondente ao valor de y_j para i-ésima observação da população e $\overline{Y}_{hj}$ corresponde à média dessa variável no h-ésimo estrato, isto é,

$$\overline{Y}_{hj} = \frac{1}{N_h} Y_{hj}, j = 1,\dots,m, h = 1,\dots,H \tag{11}$$

$$Y_{hj} = \sum_{i \in E_h} y_{ij} \ , \text{h=1,...,H e j=1,...,m} \tag{12}$$

$$Y_j = \sum_{h=1}^{H} \sum_{i \in E_h} y_{ij} = \sum_{h=1}^{H} Y_{hj} \tag{13}$$

Considerando a utilização de amostragem estratificada [28], a variância para cada uma das m variáveis de pesquisa é definida por:

$$V(t_j) = \sum_{h=1}^{H} N_h^2 \left(\frac{1}{n_h} - \frac{1}{N_h}\right) S_{hj}^2, j = 1,\dots,m \tag{14}$$

Como os valores de N_h e S_{hj}^2 são obtidos a priori, a partir da definição dos estratos, o valor da variância em (14) só depende dos tamanhos de amostra n_h que serão alocados aos estratos. A determinação dos tamanhos de amostra caracteriza o problema de alocação ótima.

De acordo com a literatura, uma possível abordagem para esse problema, diz respeito à determinação [8] dos tamanhos de amostra n_h que serão alocados aos estratos, de forma que o tamanho total da amostra n (n_1+...+n_H) seja o menor possível, e que os coeficientes de variação (cv_j) associados às estimativas produzidas para as variáveis de pesquisa não excedam patamares definidos a priori. Isso implica resolver o seguinte problema:

$$\text{Minimizar } \sum_{h=1}^{H} n_h \tag{15}$$

Sujeito a

$$n_{min} \leq n_h \leq N_h \quad \text{(h=1,...,H)} \tag{16}$$

$$\sqrt{V(t_j)}/Y_j \leq cv_j \ \ j = 1,\dots,m \tag{17}$$

$$n_h \in Z_+ \ \text{(h=1,...,H)} \tag{18}$$

Nessa formulação, a função objetivo a ser minimizada (equação 15) corresponde à soma dos tamanhos de amostra alocados aos estratos. A restrição apresentada na equação (16) garante que serão alocadas, pelo menos, n_{min} unidades amostrais a cada um dos estratos e que o número de unidades alocadas não ultrapassará os tamanhos dos estratos.

Já a restrição associada à equação (17) garante que a razão entre o desvio padrão do estimador de total de cada variável de pesquisa e o seu respectivo total será menor ou igual a um coeficiente de variação cv_j (j=1,...,m), (chamado de cv alvo) fixado previamente. Finalmente, a restrição da equação (18) garante que os tamanhos de amostra alocados aos estratos serão inteiros (restrição de integralidade do problema).

3.5.1 DECODIFICADOR PARA PROBLEMA DE ALOCAÇÃO ÓTIMA

Considerando a formulação definida por (15)-(18), cada vetor u foi definido com H componentes $u=(u_1,u_2,...,u_H)$ (número de estratos). A partir de u, foi elaborado um decodificador que produz vetores da forma $a=(n_1,...,n_h,...,n_H)$, que contém os tamanhos de amostra que serão alocados aos H estratos, mais especificamente, cada n_h é obtido a partir da aplicação da seguinte expressão: $n_h=n_{min}+\text{round}(u_h.(N_h-n_{min}))$, sendo n_{min} o tamanho mínimo de amostra alocado a cada estrato, u_h o valor da h-ésima componente de u e N_h o tamanho populacional no h-ésimo estrato.

Para exemplificar, definindo H=3, n_{min}=2, N_1=100, N_2=250, N_3=80 e u= (0.75, 0.19, 0.45), temos a=(76,49,37).

Essa decodificação garante o cumprimento imediato das restrições (16) e (18) da formulação supracitada. Todavia, alguns dos cromossomos decodificados durante as gerações do BRKGA podem levar a vetores $\boldsymbol{a}$ que não são viáveis em relação à restrição (17). Destarte, nesta proposta, a função objetivo f (Equação 15) avaliada durante as gerações do BRKGA incorpora um termo de penalidade P associado à restrição (17):

$$f = \sum_{h=1}^{H} n_h + P \qquad (19)$$

Esse termo é definido a partir do cálculo de um termo R, tal que $R = max_{j=1,...,m}\left(\frac{cv_{a_j}}{cv_j}\right)$. Se $R \leq 1$, ou seja, todos os coeficientes de variação (cv_{a_j}) associados à solução $a=(n_1,...,n_h,...,n_H)$ são menores ou iguais aos coeficientes de variação (cv_j) fixados a priori, define-se P=0; caso contrário, $P=T^R$ *(T=2000).*

A introdução desse fator de penalização na função objetivo original garante a geração e a perpetuação de soluções viáveis durante as gerações do BRKGA, isto é, de soluções que satisfaçam todas as restrições, em particular, restrições da equação (17).

3.6 PROBLEMA DE AGRUPAMENTO COM SOMA MÍNIMA

Considere um conjunto X formado por n objetos, $X=\{x_1,...,x_i,...,x_n\}$ e que cada objeto x_i tenha q atributos, isto é, $x_i=(x_{i1},x_{i2},...,x_{iq})$. Suponha, ainda, que a distância d_{ij} entre dois objetos x_i e x_j quaisquer de X seja definida, conforme equação (20) a seguir.

$$d_{ij} = \sqrt{\sum_{r=1}^{q}(x_{ir} - x_{jr})^2} \qquad (20)$$

No problema de agrupamento com soma mínima [29][1], deve-se alocar os n objetos de X em k grupos, denotados por $G_1,...,G_k$, de forma que a soma total das distâncias d_{ij} entre todos os objetos, tomados dois a dois, dentro de cada um dos grupos, seja mínima. Ou seja, busca-se minimizar a seguinte função objetivo:

$$f = \sum_{l=1}^{k} \sum_{\forall x_i, x_j \in G_l} d_{ij} \quad (21)$$

3.6.1 DECODIFICADOR PARA PROBLEMA DE AGRUPAMENTO COM SOMA MÍNIMA

Assim como problema de agrupamento apresentado em 3.2, neste problema, cada vetor *u* é definido *n* posições correspondentes ao número de objetos que serão agrupados. Em relação ao decodificador, em um primeiro passo, efetua-se a divisão do intervalo [0,1] em *k* (número de grupos) subintervalos {[0,*1/k*); [*1/k*,*2/k*);...;[*(k-1)/k*,1)}.

Em seguida, é criado a partir de *u*, um vetor *v* de *n* posições com valores entre 1 e *k*, sendo o valor atribuído à *i*-ésima posição de *v* correspondente ao número do subintervalo no qual o *i*-ésimo elemento de *u* foi encontrado. Assim sendo, o vetor *v* define alocação dos *n* objetos aos grupos G_i (*i*=1,...,*k*).

A fim exemplificar esse decodificador, considere um conjunto com *n*=10 objetos e 3 grupos (*k*=3). Utilizando o vetor u=(0.75,0.11,0.20,0.89,0.19,0.23,0.94,0.55,0.69,0.61) e os intervalos [0,1/3);[1/3,2/3);[2/3,1) temos que v=(3,1,1,3,1,1,3,2,3,2), ou seja, G_1={2,3,5,6}, G_2={8,10} e G_3={1,4,7,9}.

Feito isso, aplicando a equação (21), calcula-se, dentro de cada um dos grupos, a soma das distâncias entre todos os objetos tomados a dois ⇒

$$f=d_{23}+d_{25}+d_{26}+d_{35}+d_{36}+d_{56}+d_{810}+d_{14}+d_{17}+d_{19}+d_{47}+d_{49}+d_{79}$$

3.7 MINIMIZAÇÃO DA FUNÇÃO DE HIMMELBLAUS

A função de Himmelblaus [30], definida pela equação abaixo, é a uma função multimodal usada para testar a performance de algoritmos de otimização.

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \quad (22)$$

Essa função tem, no domínio definido por -5≤x≤5 e -5≤y≤5, um máximo local em x=-0.270845 e y=-0.923039 (f(x,y)=181.617) e quatro pontos de mínimo local idênticos, quais sejam: f(3.0,2.0) = f(-.805118,3.131312) = f(-3.779310,-3.283186) =f(3.584428,-1.848126)=0.

Para minimização de *f(x,y)*, cada vetor *u* foi definido com duas posições correspondentes às duas variáveis da função e com valores em [0,1]. O decodificador foi definido por $x=-5+u_1*10$ e $y=-5+u_2*10$; o que garante a geração de valores de x e y no intervalo [-5,5].

4. O PACOTE BRKGA

4.1 APRESENTAÇÃO DO PACOTE

A presente seção traz uma explanação quanto à utilização e funcionamento das funções disponíveis no pacote BRKGA, disponível em **github.com/jambrito/BRKGA** para download e instalação ou fazendo a instalação a partir do RStudio, utilizando a linha de comando **devtools::install_github("jambrito/BRKGA")**. Após a instalar esse pacote no *RStudio* (ambiente de desenvolvimento integrado para o R), e consultar seu o *help*, o usuário visualiza as funções apresentadas na Figura 3 a seguir.

O pacote agrega três grupos de funções, quais sejam: **(i)** brkga, popgen e crossover, associadas, diretamente, à aplicação da metaheurística BRKGA; **(ii)** as funções iniciadas pela palavra "**Decoder**" e **(iii)** as funções iniciadas pela palavra "**Fobj**"; que correspondem,

respectivamente, aos decodificadores e às funções objetivo dos problemas de otimização descritos na seção 3.

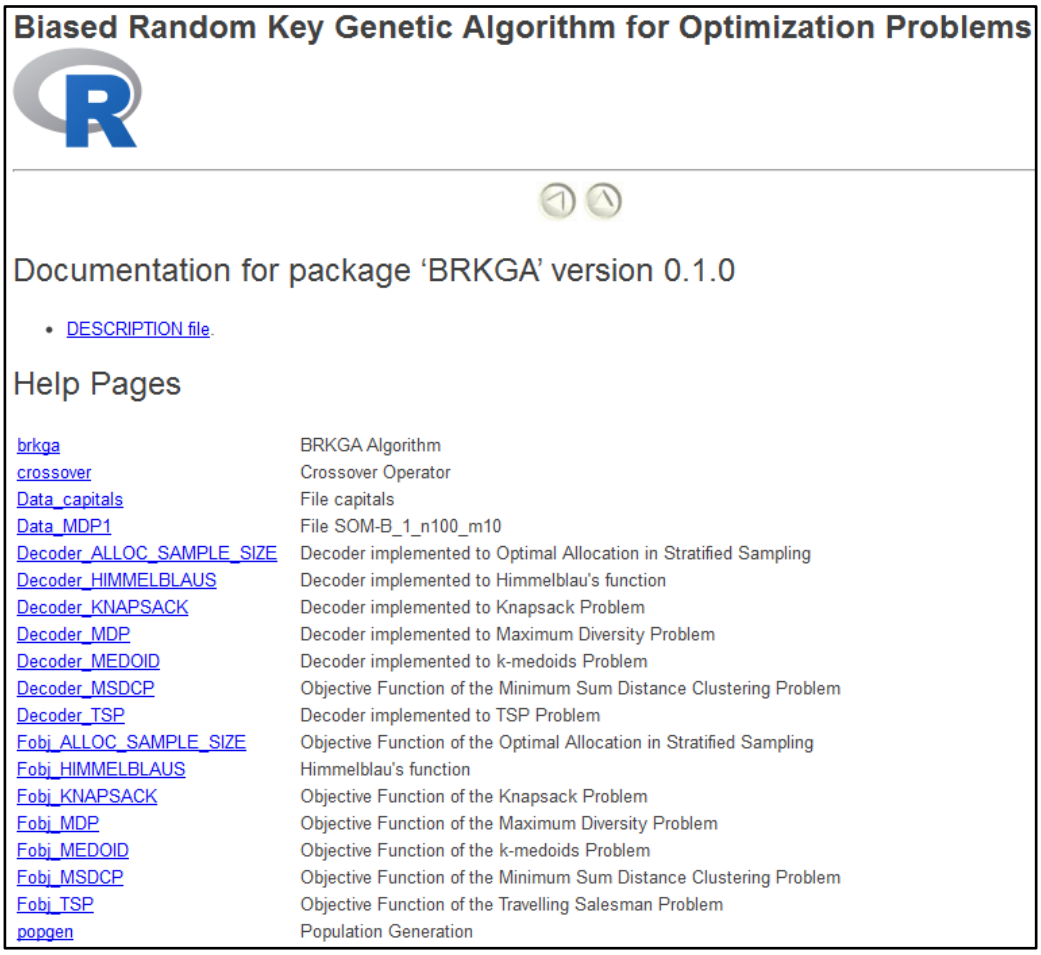


Figura 3 – Help com todas as funções do pacote BRKGA

A função brkga implementa o algoritmo genético de chaves aleatórias viciadas, que pode ser aplicado para resolver diversos problemas de otimização, considerando que o usuário tenha implementado, previamente no R, a função objetivo e o decodificador associados ao problema. Além dos argumentos comuns a um algoritmo baseado na metaheurística BRKGA, o usuário passa como argumentos, para essa função, a função objetivo e a função associada ao decodificador.

Em relação à Figura 4, que traz o *help* da função brkga, o argumento *Data* (uma matriz ou vetor) corresponde ao dado de entrada utilizado pelo algoritmo no problema de otimização que será resolvido. Os argumentos *Fo* e *Dc* estão associados, respectivamente, à função objetivo e ao decodificador implementados para o problema de otimização em questão.

Os argumentos *rc* (probabilidade de cruzamento), *pe* (percentual de soluções elite), *pm* (percentual de soluções mutante), *n* (tamanho dos cromossomos), *p* (tamanho da população), *ng* (número de gerações), *ngw* (número de gerações sem melhora no valor da função objetivo), MaxTime (tempo máximo de processamento) e *MAX* (define se o problema é de maximização ou minimização) são comuns a qualquer algoritmo BRKGA. Ainda neste sentido, os valores iniciais, atribuídos aos argumentos *p*, *ng*, *ngw*, *MaxTime*, *rc*, *pe*, *pm*, foram definidos a partir das recomendações feitas em [11] e [16].

A execução da função brkga retorna, em um objeto do tipo lista, o valor da função objetivo (*fbest*), o vetor solução do problema (*gbest*) e o tempo de processamento em segundos (*cpu_time*).

BRKGA Algorithm

Description

This function applies BRKGA algorithm to a problem considering Objective Function and Decoder defined by user

Usage

```
brkga(Data, Fo, Dc, rc = 0.7, pe = 0.2, pm = 0.2, n, p = 100,
  ng = 2000, ngw = 500, MaxTime = 3600, MAX = FALSE, Exa1 = NULL,
  Exa2 = NULL)
```

Arguments

Data	Vector or Matrix
Fo	Objective function defined by user
Dc	Decoder defined by user
rc	Crossover probability
pe	Percentual of elite chromosomes
pm	Percentual of mutant chromosomes
n	Number of genes in the chromosomes associated with a solution
p	Number of elements (chromosomes) in the population
ng	Number of generations of the brkga algorithm
ngw	Number of generations without improvement
MaxTime	Maximum CPU Time (seconds)
MAX	Argument that determines maximization (TRUE) or minimization problem (FALSE)
Exa1	Extra Argument to Decoder
Exa2	Extra Argument to Objective Function

Details

brkga

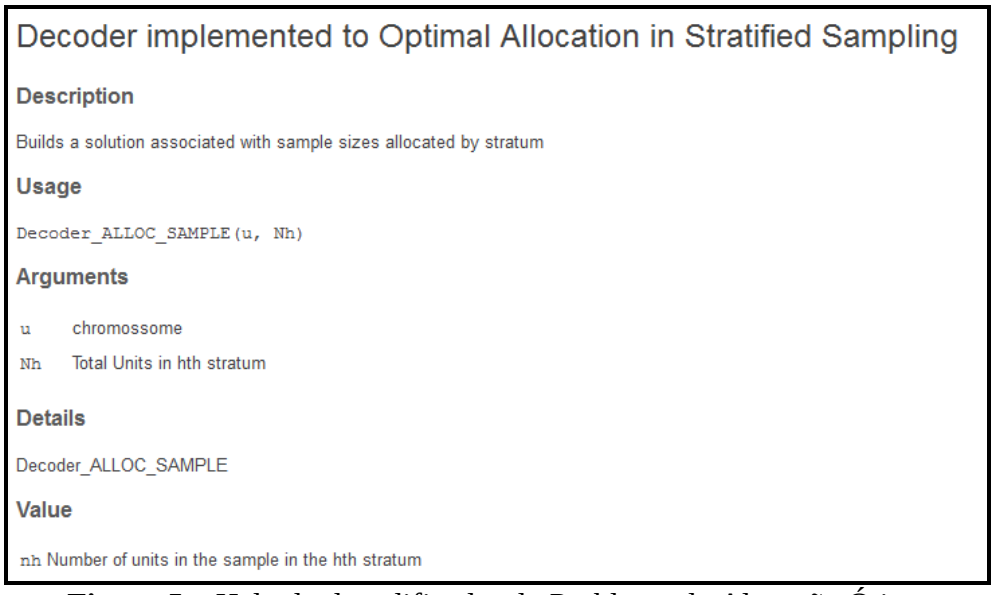
Value

fbest	Best value of Objective Function
gbest	Best Solution
cpu_time	Cpu time in seconds

Figura 4 – Help com a apresentação da função brkga

Complementarmente, *Exa1* e *Exa2* são argumentos adicionais que podem ser utilizados de acordo com a definição da função objetivo e do decodificador. Mais especificamente, caso o usuário defina para o decodificador uma função que tem mais de um argumento, além do vetor de chaves aleatórias *u*, essa informação deve ser especificada no argumento *Exa1*. De igual forma, caso a função objetivo definida pelo usuário tenha, além da entrada de dados e do vetor solução *x*, mais alguma informação, isso deve ser especificado no argumento *Exa2*.

O uso desses argumentos é exemplificado no caso do Problema de Alocação Ótima, mediante as Figuras 5 e 6, que trazem o *help* associado às funções do decodificador e da função objetivo disponíveis no pacote para resolver esse problema. Em relação ao decodificador, o argumento *Nh* está associado ao argumento *Exa1* e em relação à função objetivo o argumento *S* está associado ao argumento *Exa2*.



Decoder implemented to Optimal Allocation in Stratified Sampling

Description

Builds a solution associated with sample sizes allocated by stratum

Usage

```
Decoder_ALLOC_SAMPLE(u, Nh)
```

Arguments

u	chromossome
Nh	Total Units in hth stratum

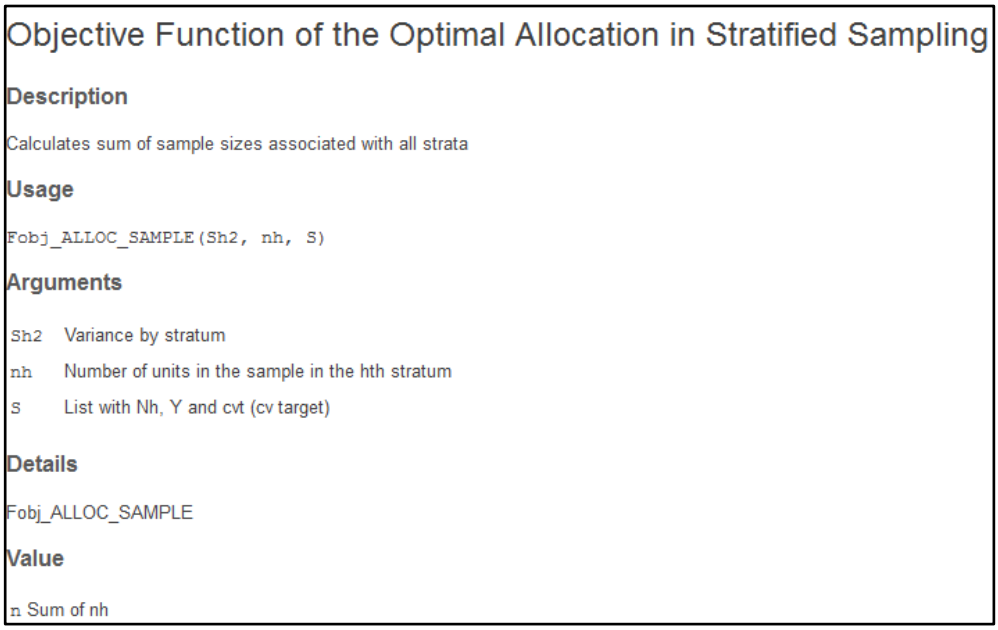
Details

Decoder_ALLOC_SAMPLE

Value

nh Number of units in the sample in the hth stratum

Figura 5 – Help do decodificador do Problema de Alocação Ótima



Objective Function of the Optimal Allocation in Stratified Sampling

Description

Calculates sum of sample sizes associated with all strata

Usage

```
Fobj_ALLOC_SAMPLE(Sh2, nh, S)
```

Arguments

Sh2	Variance by stratum
nh	Number of units in the sample in the hth stratum
S	List with Nh, Y and cvt (cv target)

Details

Fobj_ALLOC_SAMPLE

Value

n Sum of nh

Figura 6 – Help da função objetivo do Problema de Alocação Ótima

São apresentados, a seguir, exemplos de aplicação da função brkga em alguns dos problemas abordados na seção 3. Tais exemplos são parte integrante do *help* dessa função (em **Examples**)

```
# TSP - Travelling Salesman Problem
data(Data_capitals)
D<-as.matrix(dist(Data_capitals))
Ncities<-nrow(Data_capitals)
s<-brkga(Data=D,Fo=Fobj_TSP,Dc=Decoder_TSP,n=Ncities,p=50)
> s
$`fbest`
[1] 46821.12
$gbest
 [1] 21 32 48  5 41 16 22  1  9 46 18  7 44 31 33 15 40 12 20 11 23 13 14 25 39 42  4 26  2 29 24
```

```
[32] 45 35 10 34  3  8 38 36 28 30  6 37 19 27 17 43 47 21
$cpu_time
elapsed
  5.87
```

 # Clustering Problem: k-medoids

```
Distance<-as.matrix(dist(iris[,1:4]))
N<-nrow(iris)
k<-3 #Clusters
s<-
brkga(Data=Distance,Fo=Fobj_MEDOID,Dc=Decoder_MEDOID,n=N,p=50,ng=1000,MaxTime=60,
Exa1=k,Exa2=k)
> s
$`fbest`
[1] 1.880905
$gbest
[1]  3 28 92
$cpu_time
elapsed
 47.43
```

 # Knapsack Problem

```
wi<-c(40,50,30,10,10,40,30)
li<-c(40,80,10,10,4,20,60)
C<-100
Datalw<-cbind(li,wi)
s<-brkga(Data=Datalw,Fo=Fobj_KNAPSACK,Dc=Decoder_KNAPSACK,n=length(wi),p=10,
ng=100,Exa2=C,MAX=TRUE)
```

 # Optimal Allocation in Stratified Sampling

```
Nh<-c(212,84,61)
Sh2<-c(723.1,2693.4,36231.7)
Y<-80548
H<-3
Exa1<-Nh
Exa2<-list(Nh=Nh,Y=Y,cvt=0.1)
s<-brkga(Data=Sh2,Fo=Fobj_ALLOC_SAMPLE,Dc=Decoder_ALLOC_SAMPLE,n=H,p=1000,
Exa1=Exa1,Exa2=Exa2,MAX=FALSE)
```

 # Minimization of Himmelblaus Function

```
nv<-2 #Number of variables
s<-brkga(Data=NULL,Fo=Fobj_HIMMELBLAUS,Dc=Decoder_HIMMELBLAUS,n=nv,p=100)
```

 # MDP Problem

```
data(DataMDP1)
D<-Data_MDP1
N<-dim(D)[1]
M<-10
RDi<-apply(as.matrix(1:N),1,function(i) sum(D[i,]))/sum((D[upper.tri(D)]))
Exa1<-list(M=M,RDi=RDi)
s<-brkga(Data=D,Fo=Fobj_MDP,Dc=Decoder_MDP,n=N,p=75,MaxTime=3,Exa1=Exa1,
MAX=TRUE)
> s$`fbest`
[1] 323
> s$gbest
 [1]   6  17  70   1  32  13  48 100  16  49
```

4.2 CONCLUSÕES

Os exemplos apresentados em 4.1 indicam que o pacote BRKGA constitui-se como uma ferramenta útil para os pesquisadores que pretendem adotar uma abordagem baseada em metaheurísticas, para resolver problemas de otimização com e sem restrições.

Conforme comentado na seção anterior, para utilizar o pacote, o usuário precisa especificar, apenas, a função objetivo e um decodificador, além de entender o funcionamento geral desta metaheurística e ter conhecimentos básicos de R.

Futuramente, pretende-se disponibilizar uma nova versão desse pacote que ofereça o recurso de paralelismo, mediante a utilização de outro pacote do R, denominado *parallel*. O paralelismo pode ser implementado, por exemplo, no cálculo da função objetivo ou na realização do cruzamento.

Finalmente, cabe observar, que os decodificadores propostos para os problemas apresentados na seção 3 representam uma possibilidade de solução, ou seja, o usuário pode tentar elaborar diversos decodificadores para um mesmo problema, e avaliar o melhor deles.

5. REFERÊNCIAS BIBLIOGRÁFICAS

[1] HANSEN, P. and JAUMARD, B. (1997). Cluster Analysis and Mathematical Programming. *Mathematical Programming*, 79:191-215.

[2] KAUFMAN L. and ROUSSEEUW P.J. (1989). *Finding Groups in Data – An Introduction to Cluster Analysis*. Wiley-Interscience Publication.

[3] SEMAAN, G.S. (2013). Algoritmos para o Problema de Agrupamento Automático. (Tese de Doutorado) – Instituto de Computação, Universidade Federal Fluminense, Rio de Janeiro, RJ.

[4] ANGELELLI, E. and SPERANZA, M.G (2002). The Periodic Vehicle Routing Problem with Intermediate Facilities, *European Journal of Operational Research*, 137:233-247, Elsevier.

[5] GOLDEN, L. F. B. and WASIL E. (2004). Very large-scale vehicle routing: new test problems, algorithms and results. Computers & Operations.

[6] DREZNER, Z and HAMACHER. H. W. (2004). Facility location: applications and theory. New York, NY: Springer.

[7] VYGEN, J. (2005). Approximation algorithms for facility location problems. Bonn, Germany: Research Institute for Discrete Mathematics.

[8] BRITO, J. A. M.; SILVA, P.L.N, SEMAAN, G. S. and MACULAN, N. (2015) . Integer Programming Formulations Applied to Optimal Allocation in Stratified Sampling. Survey Methodology, 41, p. 427-442.

[9] WOLSEY, L.A. and NEMHAUSER, G.L. (1999). Integer and Combinatorial Optimization, first edition, Wiley-Interscience.

[10] MARTÍ, R., PARDALOS, P.M and RESENDE, M.G.C. (2018). Handbook of Heuristics. Springer.

[11] GONÇALVES, J.R. and RESENDE, M.G.C (2011). Biased random-key genetic algorithms for combinatorial optimization, *Journal of Heuristics*, 17: 487-525.

[12] APPLEGATE D. L., BIXBY, R. E., CHVÁTAL V. and Cook, W. (2006). The travelling salesman problem: a computational Study. Princeton: Princeton University Press.

[13] RESENDE, M.G.C. Introdução aos Algoritmos Genéticos de Chaves Aleatórias Viciadas. Anais do XLVSBPO, p. 3680-3691, Natal/RN, 2013.

[14] LINDEN, R. (2012). Algoritmos Genéticos. Editora Ciência Moderna. 3a edição.

[15] SPEARS, W.M., DEJONG, K.A. (1991). On the virtues of parameterized uniform crossover. *In*: *Proceedings* of the Fourth International Conference on Genetic Algorithms, p. 230-236.

[16] RESENDE, M.G.C. (2013). Biased random-key genetic algorithms. Talk given at XLV Symposium of the Brazilian Operational Research Society (XLV SBPO) Natal, RN.

[17] GAREY, M. and JOHNSON, D. S (1990). Computer and Intractibility: A guide to the Theory of NPCompleteness, Freeman, San Francisco.

[18] MARTELLO, S. and TOTH, P. (1990). Knapsack Problems - Algorithms and Computer Implementations. Wiley.

[19] GOLDBARG, M.C. e LUNA, H.P.L. (2000). Otimização Combinatória e Programação Linear – Modelos e Algoritmos. Editora Campus.

[20] TSOULOS, L.G. (2009). Solving Constrained Optimization Problems Using a Novel Genetic Algorithm. Applied Mathematics and Computation, 208, p. 273-283.

[21] YENIAY, O. and A. BEYTEPE (2005). Penalty Function Methods for Constrained Optimization with Genetic Algorithms. Mathematical and Computational Applications, 10(1), pp.45-56,

[22] GLOVER, F. HERSH, G. e MILLAN, C. (1977). Selecting subsets of maximum diversity, MS/IS Report no 77-9, University of Colorado at Boulder.

[23] MARTÍ, R., GALLEGO, M., DUARTE, A. and PARDO, E. g. (2013) heuristics and metaheuristics for the maximum diversity problem. Journal of Heuristics 19(4): 591-615.

[24] KUBY, M. J. (1987). Programming models for facility dispersion: The p-dispersion and maximum dispersion problem. *Geographical Analysis*, 19(4):315-329.

[25] ERKUT, E. (1990). The discrete p-dispersion problem. *European Journal of Operational Research* 46(1): 48–60.

[26] CAMPELLO, R.E. e MACULAN, N. (1994). Algoritmos e Heurísticas – Desenvolvimento e Avaliação de Performance. Editora UFF.

[27] COCHRAN, W. G. (1977). *Sampling Techniques*. Third Edition – Wiley.

[28] LOHR, S. (2009). Survey sampling, New York: Addison and Huxley publications, 2nd Ed.

[29] FRIEDMAN, J. H. and MEULMAN J.J. (2004). Clustering objects on subsets of attributes. Journal Royal Statistics Society B, Part 4,66:815-849.

[30] HIMMELBLAU, D. (1972). Applied Nonlinear Programming. McGraw-Hill.