Aplicação da metaheurística BRKGA com heurística de busca local para o problema de agrupamento com restrições

Rudinei Martins de Oliveira
Universidade Federal de São Paulo - UNIFESP
São José dos Campos, SP
rudmart@gmail.com

Antonio Augusto Chaves
Universidade Federal de São Paulo - UNIFESP
São José dos Campos, SP
antonio.chaves@unifesp.br

Luiz Antonio Nogueira Lorena
Instituto Nacional de Pesquisas Espaciais - INPE
São José dos Campos, SP
luiz.lorena@inpe.br

RESUMO

Este artigo propõe um método híbrido que combina o BRKGA com uma heurística de busca local para resolver o problema de agrupamentos com restrições. O problema de agrupamentos consiste em separar um conjunto de objetos em grupos tal que os membros de cada grupo sejam similares entre si. No problema de agrupamentos com restrições, alguns objetos são definidos a priori para estar no mesmo grupo (restrições *must-link*) ou em grupos distintos (restrições *cannot-link*). Este problema é classificado como *NP-hard*. O BRKGA é uma recente metaheurística que codifica uma solução como um vetor de chaves aleatórias e produz uma solução viável através de um algoritmo determinista. Os resultados computacionais considerando dados reais disponíveis na literatura são comparados com uma abordagem de geração de colunas.

PALAVRAS CHAVE. BRKGA, Busca local, problema de agrupamentos, heurística, restrições.

Tópicos (Metaheurísticas)

ABSTRACT

This paper proposes a hybrid method that combines the BRKGA with a local search heuristic to solve the clustering problem with constraints. The clustering problem consists in separating a set of objects into groups such that members of each group are similar to each other. In the clustering problem with constraints, some objects are defined, a priori, to be in the same group (must-link constraints) or in distinct groups (cannot-link constraints). This problem is well known to be NP-hard. The BRKGA is a recent metaheuristic that encodes a solution as a vector of random keys and produces a feasible solution through a deterministic algorithm. Computational results considering real data available in the literature are compared with a column generation approach.

KEYWORDS. BRKGA; Local Search, Clustering Problem, Heuristic, Constraints.

Paper topics (Metaheuristics)

1. Introdução

Problemas de agrupamentos envolvem a análise de grandes volumes de dados podendo ter várias aplicações no dia a dia, tais como, identificar pacientes com quadros clínicos semelhantes, agrupar problemas de saúde pública, encontrar padrões no comportamento dos clientes que fazem compras de produtos semelhantes, verificar o comportamento dos usuários em determinada página da *web*, agrupar ações com as mesmas flutuações de preço, entre outros [Babaki et al., 2014].

Dado um conjunto de dados que apresenta atributos e representa algo do mundo real, o problema de agrupamentos consiste no processo de separar o conjunto de dados em grupos, tal que os membros de cada grupo sejam similares entre si. No problema de agrupamento com restrições alguns objetos são definidos a priori para estar juntos no mesmo grupo (restrições *must-link*) ou separados em grupos distintos (restrições *cannot-link*). Cada restrição está associada a dois objetos. *Must-link* indica que se um objeto está associado a um grupo o outro objeto também deve estar, e *cannot-link* indica que se um objeto está associado a um grupo o outro não deve estar.

A dificuldade ocorre quando alguns atributos não são definidos de forma clara ou geram interpretações equivocadas. A questão a ser resolvida é a maneira adequada de agrupar esses dados.

Para que os dados sejam agrupados é necessário identificar quão próximos ou distantes eles estão. Um dos caminhos seria a criação de uma matriz de distâncias e em seguida determinar a similaridade usando uma métrica. Neste trabalho foi utilizada a distância euclidiana para verificar a similaridade entre os objetos. Os grupos criados são analisados por uma função custo que busca minimizar as distâncias entre elementos do mesmo grupo.

O problema de identificar os dados que são semelhantes e desenvolver métodos para agrupá-los não é uma tarefa fácil, devido ao grande número de soluções. Dessa forma, os métodos podem não conseguir classificar os dados de forma eficiente. Esta classificação pode ser melhorada quando se impõe restrições *must-link* e *cannot-link* na formulação do problema [Wagstaff e Cardie, 2000].

Em geral, o problema de determinar uma solução de agrupamentos que satisfazem todas as restrições é classificado como *NP-hard* [Davidson e Ravi, 2005]. Então, é interessante usar heurísticas para encontrar boas soluções viáveis.

Neste contexto, o objetivo deste trabalho é resolver o problema de agrupamentos com restrições (*must-link* e *cannot-link*). O método híbrido proposto é baseado no Biased Random Key Genetic Algorithm (BRKGA) com uma heurística de busca local. O BRKGA representa as soluções em vetores de chaves aleatórias. Estes vetores são decodificado em soluções reais para um problema específico. Esta característica faz com que o método se torne independente do problema. Assim, o único componente que necessita ser implementado é a função de decodificação. Neste artigo também se aplica uma heurística de busca local para acelerar a convergência do método. Os resultados computacionais mostram que este método híbrido é uma boa alternativa para resolver o problema de agrupamentos com restrições.

O trabalho está organizado da seguinte forma. Seção 2 apresenta uma breve revisão bibliográfica sobre o problema de agrupamentos com restrições. Seção 3 apresenta uma visão geral do método híbrido BRKGA com a heurística de busca local. Seção 4 apresenta os dados e os resultados computacionais. Na Seção 5 as conclusões e considerações finais são relatadas.

2. Revião da literatura

O problema de agrupamento tem sido amplamente estudado. Chang et al. [2009] propõem um algoritmo de separação baseado em um algoritmo genetico com rearranjamento de genes. O método busca remover as degenerações. Um operador de *crossover* que explora a similaridade entre os cromossomos em uma população também é apresentado.

Oliveira et al. [2014] examina heurísticas híbridas para resolução do problema de agrupamentos. As heurísticas híbridas são baseadas na aplicação de uma técnica de geração de colunas para resolver o problema de p-medianas.

Muitos trabalhos investigam a variabilidade e complexidade computacional das restrições no problema de agrupamentos. Kleinberg [2002] desenvolveu um axioma para o problema de agrupamentos. Como resultado desenvolveu o teorema da impossibilidade.

Davidson et al. [2006] resolvem o problema de agrupamento com restrições e estudam o impacto das restriões na solução final. Davidson e Ravi [2007] apresentam algumas combinações de restrições que tornam a solução inviável e ilustram tanto formal como experimentalmente as implicações desses resultados para a utilização das restrições no problema de agrupamentos.

Jiang et al. [2013] introduzem as restrições de pares de pontos elites (elite *must-links* - (EML) e elite *cannot-links* - (ECL)), e desenvolvem um método heurístico para obter as restrições. Além disso, um método chamado *Limit Crossing* é proposto para extrair as restriões EML e ECL.

Dao et al. [2015] apresentam um modelo para o problema de agrupamentos com restrições que tem três critérios: minimizar o diâmetro máximo, maximizar a separação entre os grupos, e minimizar a soma das dissimilaridades. A estrutura é integrada num algoritmo, que utiliza um modelo para minimizar o diâmetro máximo e em seguida, maximiza a separação entre os agrupamentos com as restrições adaptadas.

De acordo com Davidson e Ravi [2005] o problema de agrupamentos com restrições podem ser dividido em dois tipos: aqueles que as restrições ajudam o algoritmo a aprender e os que as restrições orientam a solução. O k-means é um exemplo de algoritmo que não é guiado pelas restrições.

O *k-means* foi amplamente utilizado nos estudos iniciais sobre o problema de agrupamento com restrições e ainda é o mais explorado na literatura. Variações usando os *k-means* são apresentadas em Wagstaff e Cardie [2000] que modificam e adicionam as restrições *must-link* e *cannot-link* no algoritmo denominado de *COP-k-means*.

Outro exemplo de melhorias no *k-means* aparece em Davidson e Ravi [2005]. O algoritmo *k-means* é modificado para que as restrições *must-link* e *cannot-link* sejam identificadas e interpretadas como conjunção ou disjunção de acordo com um limiar estabelecido. Os autores também analisam a viabilidade e a complexidade do problema.

Embora o algoritmo *k-means* seja fácil de implementar, ele não resolve o problema de agrupamentos para um número variado de restrições, assim métodos mais robustos são necessários. Buscando melhorar os resultados para o problema de agrupamentos com restrições Babaki et al. [2014] usa um algoritmo de geração de colunas em um ambiente de programação linear inteira. O processo de geração de colunas é responsável pela identificação dos objetos candidatos que podem ser adicionados ao grupo.

3. *Biased Random Key Genetic Algorithm* (BRKGA)

O *Biased Random Key Genetic Algorithm* (BRKGA) foi proposto por [Gonçalves e Resende, 2011]. É uma variação do *Random Key Genetic Algorithms* (RKGA) [Bean, 1994].

O BRKGA representa um cromossomo como um vetor de chaves aleatórias. Este vetor (cromossomo) é composto por números reais no intervalo $[0, 1]$. Este vetor (também chamado de cromossomo) não é considerado como uma solução do problema. Então, é necessário decodificá-lo na solução do problema que está sendo resolvido. Para cada problema é definido um decodificador específico. O decoder é um algoritmo determinístico que pega as informações do cromossomo e retorna em uma solução do problema. A aptidão (fução objetivo) de cada solução também é calculada pelo decodificador.

O processo de evolução do BRKGA é independente do problema. A população de P vetores de chave aleatórias evolui ao longo das gerações. Em cada geração, a população é ordenada pelo valor da função objetivo. Em seguida, um pequeno grupo com as melhores soluções (p_e) (grupo elite) são copiados sem modificação para a população da próxima geração. Um número (p_m) de vetores de chaves aleatórias randomicamente gerados (mutantes) também são introduzidos nesta população. O restante da população ($P - p_e - p_m$ soluções) é produzido através do cruzamento, combinando os pais dos vetores elites com os não elites na solução corrente.

Neste artigo, o BRKGA foi adaptado para resolver o problema de agrupamento com restrições. O método cria aleatoriamente uma população inicial de vetores de chaves aleatórias. Cada vetor tem n chaves aleatórias, tal que, n é o número de objetos dos dados. A solução do problema é obtida pelos correspondentes valores das chaves aleatórios pelo decodificador. No decodificador o intervalo $[0, 1]$ é dividido por k grupos e os clusters são criados com os objetos que possuem as chaves aleatórias nesse intervalo. As Figuras 1 e 2 mostram um exemplo do processo de decodificação para o problema de agrupamento. Neste exemplo, existem dez objetos. As chaves aleatórias ordenadas correspondem a três grupos ($K = 3$) $\{(1, 5, 7, 9), (2, 3, 4), (6, 8, 10)\}$.

Index	1	2	3	4	5	6	7	8	9	10
Random keys	0.12	0.37	0.66	0.56	0.04	0.97	0.23	0.75	0.15	0.89

Figura 1: Exemplo de chaves aleatórias.

Index	1	2	3	4	5	6	7	8	9	10
Random keys	0.12	0.37	0.66	0.56	0.04	0.97	0.23	0.75	0.15	0.89
Clusters	1	2	2	2	1	3	1	3	1	3

Figura 2: Exemplo de chaves aleatórias depois do decodificador.

Depois de decodificar a solução, a aptidão é calculada através da função objetivo proposta em Babaki et al. [2014]. A função objetivo (Z) minimiza a distância (d) entre os elementos do mesmo grupo, dividido pelo número de elementos de cada grupo ($|C|$) , como na Equação 1.

$$Z = \frac{\sum_{x_1, x_2 \in C} d^2(x_1, x_2)}{|C|} \tag{1}$$

De tal forma que, $x_1 \neq x_2$ pertencem ao grupo C e $|C|$ é a cardinalidade do grupo. Cada par de dois pontos em C é incluído na soma uma vez, sem repetição. Por exemplo, o cálculo das distâncias dos objetos na Figura 2 é: $d(5, 1)$, $d(5, 9)$, $d(5, 7)$, $d(1, 9)$, $d(1, 7)$ e $d(9, 7)$ no grupo 1 e $d(2, 4)$, $d(2, 3)$ e $d(4, 3)$ no grupo 2 e $d(8, 10)$, $d(8, 6)$ e $d(10, 6)$ no grupo 3. O cálculo de Z é feito simplesmente dividindo a soma das distâncias de cada grupo pelo número de elementos deste grupo.

Em seguida, é verificado se a solução satisfaz a restrição definida. Quando falha, é aplicada uma penalidade. Assim, a aptidão é a soma da função objetivo acrescido da penalidade. A penalidade é um termo que aumenta proporcionalmente com a quantidade de inviabilidades obtidas em cada solução analisada (Equação 2).

$$\text{aptidão} = Z + \underbrace{(\mu * n * infeasibility)}_{penalidade} \tag{2}$$

Tal que μ é um valor alto e n é o número de objetos e a inviabilidade é o número de restrições não satisfeitas.

Para acelerar o processo de convergência do BRKGA, implementamos uma heurística de busca local, que é aplicada a toda a decendência gerada pelo BRKGA. A busca local é aplicada na solução decodificada. A solução encontrada pela busca local não é transferida para o vetor de chaves aleatórias. Isto preserva a diversidade do BRKGA. O Algoritmo 1 apresenta a heurística de busca local proposta neste trabalho. A busca local move o objeto de um grupo para os outros. Cada

objeto é movido para um conjunto diferente. A solução vizinha (S') recebe a solução corrente (S) e realiza movimentos para os outros grupos. Quando a solução vizinha (S') é melhor do que a solução corrente, a solução atual (S) recebe a solução vizinha (S') e a melhor solução (S^*) é armazenada. Esta heurística é executada enquanto a solução é melhorada.

Algoritmo 1: Busca local (S)

```
1  enquanto (Melhora S ) faça
2      para (cada objeto) faça
3          para (cada grupo) faça
4              S' ← S;
5              S' ← Move (um objeto para o outro grupo);
6              se S' é melhor que S então
7                  S ← S';
8                  armazena (a melhor solução S*);
9              fim
10         fim
11     fim
12 fim
13 return (S*)
```

A Figura 3 apresenta o fluxograma do BRKGA com busca local proposto neste trabalho.

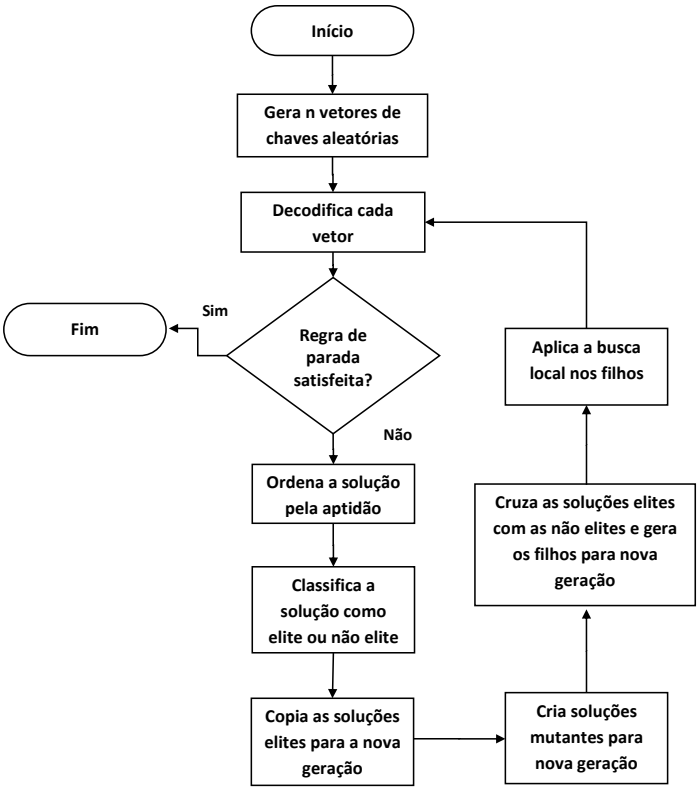

Figura 3: Fluxograma do BRKGA e heurística de busca local adaptada de Gonçalves e Resende [2011].

4. resultados Computacionais

Esta seção apresenta os dados utilizados e os resultados computacionais. Foram considerados um total de cinco conjunto de dados: Wine, Soybean, BreastA and DLBCLB (Tabela 1). Os dados de Iris, wine e Soybean foram obtidos no UCI *machine learning repository* [Lichman, 2013]. Os dados de BreastA e DLBCLB são do programa de repositório de câncer (`http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi`).

Os dados de Iris são compostos por 150 objetos (flores) separados em quatro atributos: larguras e comprimentos da sépala e da pétala e são divididas em 3 grupos: iris virgínica, iris setosa e iris versicolor.

Os dados de Wine consistem de 178 objetos e são o resultados de uma análise química de vinhos produzidos na mesma região da Itália, são derivados de três diferentes cultivares. Possui 13 atributos, para 3 tipos de vinhos.

Os dados de Soybean tem 47 objetos, com 35 Atributos e quatro grupos. Os dados de BreastA possui 98 objetos de tumor de mama com 1213 atributos e é dividido em 3 grupos com 11, 51 e 36 objetos, que levam em consideração o receptor estrogênico positivo e negativo.

Os dados de DLBCLA correspondem a exemplos de linfomas de grandes células B e são compostos por 141 objetos. O DLBCLA apresenta 661 atributos separados por 3 grupos: fosforilação oxidativa (49 objetos), respostas das células B (50 objetos) e respostas de hospedeiros (42 objetos).

Tabela 1: Característica dos dados.

Nome	Objetos	Dimensão	Quantidade de grupos
Iris	150	4	3
Wine	178	13	3
Soybean	47	35	4
BreastA	98	1213	3
DLBCLB	180	661	3

As restrições foram geradas de acordo com Wagstaff e Cardie [2000] e Babaki et al. [2014]. Elas foram obtidas selecionando dois objetos aleatórios e, em seguida, é verificado se eles estão no mesmo grupo. Se os objetos estão no mesmo grupo, a restrição é classificada como *must-link*, caso contrário, *cannot-link*.

Os experimentos foram realizados em um PC Intel Core i5 de 2,6 GHz e 4 GB de RAM. Os parâmetros usados para o método híbrido é apresentado na Tabela 2 e foram obtidos empiricamente com sugestões de Gonçalves e Resende [2011] e Chaves et al. [2016].

Tabela 2: Parâmetros usados nos experimentos.

Parâmetros	Significado	Valor
p	Número de indivíduos da população	1000
Gen	Número de gerações	1000
p_e	Fração do conjunto elite da população	0.2
p_m	Fração do conjunto de mutantes para ser introduzido na população a cada geração	0.2
ρ_e	probabilidade de que descendentes herdam um alelo do pai da população elite	0.6

As tabelas de 3 a 7 apresentam os resultados computacionais obtidos pelo BRKGA com o algoritmo de busca local (BRKGA+BL) proposto neste artigo. Em geral foram realizadas 50 execuções para cada conjunto de restrições. Nestas tabelas, o número de restrições é a primeira coluna (restr). A segunda coluna apresenta os resultados do método de geração de colunas de

Babaki et al. [2014] (GC). A terceira coluna mostra os melhores resultados do método BRKGA+BL em 50 execuções. A quarta coluna apresenta as porcentagens de quantas vezes o BRKGA+BL satisfez todas as restrições (%) nas 50 execuções. Finalmente, a quinta coluna mostra o tempo de CPU do BRKGA+BL em segundos (Tempo) para encontrar a melhor solução. As soluções em negrito indicam quando o BRKGA+BL foi melhor e o asterisco (*) indica uma ótima solução encontrada. Todos os dados foram agrupados de acordo com a Tabela 1.

O método Geração de Colunas (GC) [Babaki et al., 2014] está disponível em `http://dtai.cs.kuleuven.be/CP4IM/cccg/`. O GC foi iniciado com as soluções da primeira execução do BRKGA+BL e estas soluções na maioria dos casos eram viáveis, mas poderia ocorrer de serem inviáveis. O GC é executou por um tempo máximo de 3600 segundos. Pode-se observar que o GC não conseguiu encontrar soluçõess viáveis para todos os conjuntos de restrições.

Analisando os resultados do BRKGA+BL nas Tabelas de 3 a 7, observa-se que, em geral, o método proposto encontrou bons resultados para todos os dados analisados, quando comparados com os resultados do GC [Babaki et al., 2014] e, adicionalmente, quando o método de GC não encontra a solução ótima o BRKGA+BL encontra as melhores soluções viáveis em menos tempo.

Tabela 3 apresenta os resultados para os dados de Iris. É possível observar que para as restrições 2-200 o BRKGA+BL encontrou boas soluções viáveis. Para as restrições 250-500 o BRKGA+BL obteve as melhores soluções. Além disso, para as restrições 2, 50 e 500, o BRKGA+BL encontrou soluções viáveis em todas as execuções. O GC não foi capaz de encontrar soluções para as restrições 100, 150 e 200.

Tabela 3: Resultados para os dados de Iris.

restr	GC	BRKGA+BL	%	Tempo(s)
2	78.9408	78.9408	100	46.09
50	212.6546	**85.5174**	100	155.85
100	-	**91.5376**	6	155.69
150	-	**147.4134**	8	232.79
200	-	**90.8079**	4	143.14
250	88.6659*	88.6659*	2	137.12
300	88.9516*	88.9516*	6	146.16
350	88.9516*	88.9516*	18	251.86
400	88.9516*	88.9516*	80	336.98
450	89.3868*	89.3868*	90	6.27
500	89.3868*	89.3868*	100	72.33

Como aconteceu com os dados de Iris, para os dados do Wine (Tabela 4) o GC não encontrou uma solução viável para as restrições 100-300. O BRKGA+BL encontrou soluções viáveis para todos os conjuntos de restrições. Para as restrições 400-500 as soluções são ótimas. Para a restrição 2 o BRKGA+BL encontrou soluções viáveis em todas as execuções.

Para dados de Soybean (Tabela 5), ambos GC e BRKGA proposto resolveram facilmente todos os conjuntos de restrições. Todas as soluções encontradas foram ótimas.

Para os dados de BreastA, novamente o GC teve dificuldades em resolver as restrições 100 e 150. O BRKGA+BL encontrou soluções viáveis para todas as restrições. Para as restrições 200-500 as soluções são ótimas. Além disso, para as restrições 2 e 50, o método proposto encontrou as soluções viáveis em todas as execuções.

Para os dados de DLBCLB o BRKGA+LS facilmente resolveu o problema de agrupamento com restrições, obtendo soluções viáveis para todas as restrições. Em 100% das execuções o BRKGA+BL obteve soluções viáveis para as restrições 2 e 350. Além disso, a percentagem de soluções viáveis encontradas foram melhores em comparação com os dados de Wine e Iris. Para as

Tabela 4: Resultados para os dados de Wine.

Restr	GC	BRKGA+BL	%	Tempo(s)
2	2438411	**2416694**	100	27.68
50	5614019	**3890524**	80	394.37
100	-	**4848681**	48	317.36
150	-	**6606605**	8	407.12
200	-	**6559526**	2	96.93
250	-	**6759045**	2	337.30
300	-	**6159239**	2	341.11
350	6061296	6061296	2	9.14
400	5198760*	5192671*	6	45.10
450	5198760*	5198760*	14	9.49
500	5198760*	5198760*	22	37.24

Tabela 5: Resultados para os dados de Soybean.

Restr	GC	BRKGA+BL	%	Tempo(s)
2	205.9637*	205.9637*	100	1.79
50	218.0647*	218.0647*	100	33.41
100	218.0647*	218.0647*	32	84.02
150	218.0647*	218.0647*	80	3.88
200	218.0647*	218.0647*	98	4.07
250	218.0647*	218.0647*	100	1.92
300	218.0647*	218.0647*	98	1.98
350	218.0647*	218.0647*	100	1.98
400	218.0647*	218.0647*	100	2.14
450	218.0647*	218.0647*	100	2.11
500	218.0647*	218.0647*	100	2.22

Tabela 6: Resultados para os dados de BreastA.

Restr	GC	BRKGA+BL	%	Tempo(s)
2	118545	**15691**	100	7.57
50	96493	**69873**	100	57.58
100	-	**112745**	36	74.83
150	-	**118545**	32	160.19
200	118684*	118684*	42	24.14
250	118726*	118726*	42	136.53
300	118937*	118937*	48	8.00
350	118937*	118937*	62	15.84
400	118937*	118937*	90	3.91
450	118937*	118937*	92	4.40
500	118937*	118937*	96	4.20

restrições 200-500 as soluções foram ótimas. Quando é analisado o GC, observa-se que o método não foi capaz de encontrar a solução para a restrição 150.

Tabela 7: Resultados para os dados de DLBCLB.

Restr	GC	BRKGA+BL	%	Tempo(s)
2	68184	**68094**	100	327.41
50	75092	**73641**	98	299.73
100	77450	**76663**	56	326.13
150	-	**78787**	24	127.39
200	79421*	79421*	48	46.18
250	80472*	80472*	90	10.05
300	80757*	80757*	94	14.10
350	81077*	81077*	100	11.03
400	81077*	81077*	98	11.15
450	81429*	81429*	98	10.09
500	81429*	81429*	98	10.16

A Tabela 8 mostra a média do número de restrições violadas quando o BRKGA+BL não encontrou uma solução viável em 50 execuções. Podemos notar que para os dados de Soybean e DLBCLB, há apenas a violação de uma restrição. Para os dados de BreastA as violações foram apenas de duas restrições. Para os dados da Iris no pior caso as violações foram de onze e para os dados de Wine no pior casos as violações totalizaram dezenove.

Tabela 8: Resultados da média de violações das restrições nas 50 execuções.

Restr	Iris	Wine	Soybean	BrestA	DLBCLB
2	0	0	0	0	0
50	0	1	0	0	1
100	1	1	1	1	1
150	2	3	1	2	1
200	5	5	1	1	1
250	10	9	0	1	1
300	11	14	1	1	1
350	5	19	1	1	0
400	5	17	1	1	1
450	2	14	1	1	1
500	0	6	1	1	1

5. Conclusão

Este trabalho apresenta uma contribuição para o problema de agrupamento com restrições usando a heurística híbrida BRKGA+BL. Os resultados computacionais consideraram os dados de Iris, Wine, Soybean, BreastA e DLBCLB. O BRKGA+BL encontrou soluções viáveis para todos os dados e restrições geradas.

O BRKGA+BL foi melhor para o os dados de Soybean, BreastA e DLBCLB. Isto pode ser explicado pelos valores dos atributos nos dados. No caso de dados de Soybean os atributos são números inteiros e tem menos objetos que os demais dados. Os dados de BreastA são números reais, mas estão bem separados. Os dados de DLBCLB são compostos por números reais e existem no máximo duas casas decimais. Os dados da Iris são números reais e Wine apresenta números reais e também há uma coluna com números inteiros.

As soluções encontradas foram ótimas para as restrições 250-500 dos dados de Iris, para as restrições 400-500 dos dados de Wine, para todas as restrições de Soybean, para as restrições 200-500 de BreastA, para as restrições 200-500 de DLBCLB. A otimalidade das soluções foram provadas por GC Babaki et al. [2014] que utiliza um método exato.

O problema de agrupamento com restrições é classificado como *NP-Hard*, então é difícil encontrar soluções viáveis em todos os conjuntos de restrições [Jiang et al., 2013]. Os bons resultados obtidos, mostram que a heurística híbrida BRKGA+BL, pode ser usada como uma alternativa para resolver o problema de agrupamento com restrições. Isto se torna evidente pela estabilidade do método na busca de soluções viáveis em todos os conjunto de dados e com baixo tempo computacional.

Agradecimentos

Os autores agradecem a FAPESP (process 2012/17523-5), CNPq (301836/2014-0, 307330/2015-0) e CAPES.

Referências

Babaki, B., Guns, T., e Nijssen, S. (2014). Constrained clustering using column generation. 8451: 438–454.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

Chang, D.-X., Zhang, X.-D., e Zheng, C.-W. (2009). A genetic algorithm with gene rearrangement for k-means clustering. *Pattern Recogn.*, 42(7):1210–1222.

Chaves, A., Lorena, L., Senne, E., e Resende, M. (2016). Hybrid method with cs and brkga applied to the minimization of tool switches problem. *Computers & Operations Research*, 67:174 – 183.

Dao, T.-B.-H., Duong, K.-C., e Vrain, C. (2015). Constrained clustering by constraint programming. *Artificial Intelligence*.

Davidson, I. e Ravi, S. S. (2005). Clustering with Constraints: Feasibility Issues and the k-Means Algorithm.

Davidson, I. e Ravi, S. S. (2007). Intractability and clustering with constraints. p. 201–208.

Davidson, I., Wagstaff, K. L., e Basu, S. (2006). Measuring constraint-set utility for partitional clustering algorithms. p. 115–126.

Gonçalves, J. F. e Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Jiang, H., Ren, Z., Xuan, J., e Wu, X. (2013). Extracting elite pairwise constraints for clustering. *Neurocomputing*, 99:124–133.

Kleinberg, J. (2002). An impossibility theorem for clustering. p. 446–453.

Lichman, M. (2013). UCI machine learning repository.

Oliveira, R. M., Lorena, L. A. N., Chaves, A. A., e Mauri, G. R. (2014). Hybrid heuristics based on column generation with path-relinking for clustering problems. *Expert Systems with Applications*, 41(11):5277 – 5284.

Wagstaff, K. e Cardie, C. (2000). Clustering with instance-level constraints. p. 1103–1110.