BRKGA PARA AUTO-PARAMETRIZAÇÃO DO GRASP COM PATH-RELINKING NO AGRUPAMENTO DE DADOS

Rafael de Magalhães Dias Frinhani
Universidade Federal de Itajubá – Instituto de Matemática e Computação
Av. BPS, 1303 - Pinheirinho, Itajubá/MG. CEP 37500-903. +55 35 3629-1830
frinhani@unifei.edu.br

Rui Martins Lacerda
Universidade Federal de Itajubá – Instituto de Matemática e Computação
Av. BPS, 1303 - Pinheirinho, Itajubá/MG. CEP 37500-903. +55 35 9812-2582
rui.mlac@gmail.com

Ricardo Martins Abreu Silva
Universidade Federal de Pernambuco – Centro de Informática
Cidade Universitária, Recife/PE. CEP 50740-560. +55 81 2126-8430.
rmas@cin.ufpe.br

Geraldo Robson Mateus
Universidade Federal de Minas Gerais – Departamento de Ciência da Computação
Av. Antônio Carlos, 6627 - ICEx Pampulha, Belo Horizonte/MG. CEP 31270-901.
+55 31 3409-5879
mateus@dcc.ufmg.br

RESUMO

O agrupamento é uma tarefa que visa a classificação de objetos similares em grupos de acordo com uma medida de similaridade. Metaheurísticas tem sido aplicadas com sucesso no agrupamento, mas embora os parâmetros de entrada possuam forte influência no sucesso do algoritmo sua definição geralmente é feita manualmente abrindo questões se realmente são os que trarão os melhores resultados. Este trabalho teve como o objetivo a aplicação do BRKGA na auto-parametrização do GRASP com *Path-Relinking* no agrupamento de dados. Experimentos comprovaram que o BRKGA contribui na obtenção de melhores resultados em comparação a parametrização manual.

PALAVARAS CHAVE. Auto-parametrização, Agrupamento, GRASP.

Área principal (Metaheurísticas, Otimização Combinatória)

ABSTRACT

Clustering is an task that aims to classify similar objects into groups according to a similarity measure. Metaheuristics has been successfully applied in clustering, but although the input parameters have a strong influence on the success of the algorithm its definition is usually done manually opening issues if the parameters chosen will bring better results. This work aims to apply the BRKGA in the auto-tuning of GRASP with Path-Relinking in data clustering. Experiments confirmed that the BRKGA contributes to obtain better results compared with manual parametrization.

KEYWORDS. Automatic Tuning, Clustering, GRASP.

Main area (Metaheuristics, Combinatorial Optimization)

1. Introdução

O agrupamento de dados é uma tarefa cujo objetivo é a partir de uma medida de similaridade reunir objetos com características similares em grupos (coesão interna), sem que se tenha um prévio conhecimento dos grupos que os objetos pertencem, bem como a quantidade de agrupamentos necessários [Anderberg, 1973; Braga, 2005; Berkhin, 2006; Jain et al., 1999].

O grande volume de dados e a quantidade ideal de grupos tornam a tarefa de agrupamento um complicado problema combinatório. Diversos algoritmos estão disponíveis para essa tarefa e cada um tem vantagens e desvantagens frente a diferentes conjuntos de dados. Os algoritmos clássicos de agrupamento são baseados em técnicas matemáticas, estatísticas e de análise numérica e tem como exemplos o *k-means* [Hartigan et al., 1979], *k-medians* [Jain & Dubes, 1978] e *Partitioning Around Medoids* [Kaufman & Rousseeuw, 1990].

Recentemente técnicas de otimização como programação linear/inteira e metaheurísticas têm sido utilizadas de forma complementar aos algoritmos clássicos visando melhorar o desempenho e/ou precisão das tarefas de agrupamento. Exemplos de metaheurísticas aplicadas no agrupamento incluem *Simulated Annealing* [Brown & Huntley, 1992], Algoritmos Genéticos [Hall et al., 1999], Redes Neurais [Herrero et al., 2001], *Greedy Randomized Adaptive Search Procedure* (GRASP) [Nascimento et al., 2010], GRASP com *Path-Relinking* (GRASP+PR) Frinhani et al. [2011]. Em metaheurísticas como o GRASP+PR embora a definição dos parâmetros seja crucial para obtenção de bons resultados, sua escolha geralmente é feita de forma manual, abrindo questões se os parâmetros escolhidos são os que trarão os melhores resultados.

O presente trabalho tem como objetivo aplicar o *Biased Random-Key Genetic Algorithm* (BRKGA) para auto-parametrização do GRASP+PR no agrupamento de dados visando obter melhorias em relação as versões parametrizadas manualmente. Este trabalho está organizado como se segue: A seção 2 contém uma descrição formal do modelo utilizado, a seção 3 descreve o GRASP e GRASP+PR no agrupamento de dados, a seção 4 descreve técnicas de auto-parametrização de metaheurísticas em especial a BRKGA, a seção 5 contém a metodologia utilizada e por fim a seção 6 apresenta a análise dos resultados e discussão.

2. Modelo Matemático

O modelo utilizado neste trabalho segue o adotado por Nascimento et al. [2010] que considera o conjunto de dados como um único grafo totalmente conexo onde a distância entre cada par de objetos, obtida a partir de seus atributos, é representada por uma aresta valorada. Durante as execuções do GRASP as arestas vão sendo eliminadas de modo a formar cliques. O objetivo é minimizar a soma total das distâncias entre os objetos do mesmo grupo considerando que, quanto menor a distância maior a similaridade. O modelo em questão é formalizado como:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} y_{ij}$$

Sujeito a:

$$\sum_{k=1}^{M} x_{ik} = 1, \quad i = 1, \dots, N \qquad (1)$$

$$\sum_{i=1}^{M} x_{ik} \geq 1, \quad k = 1, \dots, M \qquad (2)$$

$$x_{ik} \in \{0, 1\}, \qquad i = 1, \dots, N, \qquad k = 1, \dots, M \qquad (3)$$

$$y_{ij} \geq x_{ik} + x_{jk} - 1, \quad i = 1, \dots, N, \quad j = i + 1, \dots, N, \quad k = 1, \dots, M \qquad (4)$$

$$y_{ij} \geq 0 \quad i = 1, \dots, N, \quad j = i + 1, \dots, N \qquad (5)$$

Onde d_{ij} é a distância entre os objetos i e j; N é a quantidade de objetos do conjunto de dados, M é a quantidade de agrupamentos, x_{ik} é a variável binária que assumirá o valor 1 se o objeto pertencer ao grupo k ou 0 em caso contrário; y_{ij} é uma variável real que assume o valor 1 se os objetos i e j pertencem ao mesmo grupo. A restrição (1) garante que o objeto i pertença a apenas um grupo; (2) garante que o grupo k possua pelo menos um objeto; (3) garante que a variável x_{ik} seja binária; (4) e (5) garante que y_{ij} assume o valor 1 se ambos os valores x_{ik} e x_{jk} são 1.

3. GRASP e GRASP com *Path-Relinking* no agrupamento de dados

GRASP [Feo & Resende, 1995] é um algoritmo de busca, multi-inícios, que numa primeira fase combina as abordagens gulosa e aleatória para construir soluções e numa segunda fase aplica uma busca local em cada solução construída visando obter melhorias. O funcionamento do GRASP tradicional não se baseia em aprendizagem sobre a sua execução já que as soluções encontradas em iterações anteriores não influenciam o algoritmo na solução atual. A utilização de mecanismos de memória permite um direcionamento do GRASP para as melhores soluções previamente encontradas através da construção de um conjunto de soluções elite (*pool*), evitando a exploração de soluções não promissoras.

Path-Relinking [Glover et al. 2000; Glover & Marti 2006] é um método de intensificação de busca que provê um mecanismo de memória ao GRASP. Seu objetivo é encontrar boas soluções no espaço entre duas soluções de alta qualidade através da exploração de trajetórias, iniciando de uma dessas soluções, chamada solução inicial, e gerando um caminho no espaço de vizinhança que conduz a solução guia. *Forward*, *Backward*, *Mixed*, *Greedy Randomized Adaptive* e *Truncated* são variantes *Path-Relinking* que possuem estratégias distintas na determinação da solução inicial/guia bem como a trajetória a ser seguida [Resende & Ribeiro, 2005; Resende et al., 2010].

Algoritmo 1: ***GRASP+PR()***

Data: *I,M,TP,p,DS,Ism,DC,S*;
Result: Solução $x^* \in X$;
1. $P := \emptyset$;
2. **while** critério de parada não alcançado **do**
3. $x' :=$ *FaseConstrutiva*();
4. **if** P possuir ao menos p elementos **then**
5. $x' :=$ *FaseBuscaLocal*(x');
6. Aleatoriamente seleciona solução $y \in P$;
7. $x' :=$ *PathRelinking*(x', y);
8. $x' :=$ *FaseBuscaLocal*(x');
9. **if** P estiver completo **then**
10. **if** $c(x') \leq \max\{c(x)\} | x \in P\}$ and $x' \not\approx P$ **then**
11. Substitui elemento mais similar a x' entre todos com custo pior que x';
12. **end**
13. **else if** $x' \not\approx P$ **then**
14. $P := P \cup \{x'\}$;
15. **end**
16. **else if** $x' \not\approx P$ and $P = \emptyset$ **then**
17. $P := P \cup \{x'\}$;
18. **end**
19. **end**

20.**end**
21.**return** *x**

Frinhani et al. [2011] utilizou o *Path-Relinking* para hibridização do GRASP no agrupamento de dados e obteve melhores resultados que o GRASP na sua versão tradicional. O Algoritmo 1 descreve o funcionamento do GRASP+PR. O algoritmo toma como entrada um conjunto de dados *I*, a quantidade *M* de grupos desejados, o tamanho *TP* do *pool P* de melhores soluções, a quantidade mínima *p* de elementos no *pool* antes do início do *Path-Relinking*, a diferença simétrica *DS* entre as soluções inicial e guia, a quantidade *Ism* de iterações sem melhoria como critério de parada, a medida de similaridade *DC* e a semente S utilizada pelo método aleatório.

Após inicializar o conjunto elite como vazio (linha 1), o GRASP+PR é executado nas linhas 2 à 19 até que *Ism* seja satisfeito. Em cada iteração uma solução x' é gerada pela fase construtiva (linha 3). Na linha 4 é verificada a quantidade de soluções pertencentes a *P*. Caso *P* esteja vazio, a solução x' é inserida no conjunto. Se *P* não possuir no mínimo *p* elementos e se a solução x' for suficientemente diferente de todas as soluções presentes em *P*, então x' é inserido em *P* (linha 16). Se *P* possuir pelo menos *p* elementos, então os passos 5 à 15 são executados. Na linha 5 é realizada a busca local objetivando a melhoria da solução x' construída na Fase Construtiva (linha 3). Uma solução $y \in P$ é escolhida aleatoriamente para aplicação do *Path-Relinking* com a solução x' (linhas 6 e 7).

Após a realização do *Path-Relinking()*, a solução x' passa pela *FaseBuscaLocal()* visando melhorias (linha 8). Caso *P* esteja completo é verificado se a solução é suficientemente diferente das soluções atualmente presentes em *P* ($x' \not\approx P$). Esta estratégia visa manter em *P* soluções de qualidade e diversificadas. Ao se respeitar este critério, a solução x' se torna candidata a entrar no conjunto elite substituído a solução *x* mais similar a x', isto é, a que apresente a menor diferença simétrica $\Delta(x, x')$. Se *P* não estiver completo a solução x' é inserida caso atenda a restrição $x' \not\approx P$. Detalhes da *FaseConstrutiva()*, *FaseBuscaLocal()* e *PathRelinking()* poderão ser obtidos em Frinhani et al. [2011].

Conforme descrito no Algoritmo 1, para execução do GRASP+PR é necessário a definição de parâmetros cujos valores variam de acordo com o conjunto de dados utilizado. Visando auxiliar esta definição, heurísticas podem ser aplicadas na auto-parametrização de algoritmos visando encontrar os melhores parâmetros para sua execução.

4. Auto-Parametrização de Metaheurísticas

A parametrização de metaheurísticas é um típico problema combinatório que envolve a definição de valores numéricos ou lógicos que deverão ser utilizados como parâmetros de entrada para execução do algoritmo. Na maioria dos trabalhos a parametrização é normalmente realizada de forma empírica abrindo questões se os parâmetros escolhidos são realmente os que trarão os melhores resultados. Nesse contexto a auto-parametrização pode trazer benefícios como a redução do tempo computacional e/ou permitir ao algoritmo alcançar melhores soluções.

Encontra-se na literatura diversos trabalhos relacionados a auto-parametrização de metaheurísticas como os métodos estatísticos *F-Race* [Birattari, 2009], *I-Race* [López-Ibáñez et al., 2011] [Birattari et al., 2002] e I/F-Race [Silva, 2015], Projeto de Experimentos e Redes Neurais [Dobslaw, 2010] e *Biased Random-Key Genetic Algorithm* (BRKGA) [Festa et al. [2010].

4.1 Biased Random-Key Genetic Algorithm (BRKGA)

Algoritmos genéticos (AG) utilizam o conceito de seleção natural, onde os indivíduos mais aptos sobrevivem, e da genética, onde as características dos pais são passadas para os filhos. Uma analogia é feita entre uma solução em um espaço de soluções e um indivíduo em uma população. Cada indivíduo é representado por um cromossomo que está associado a uma função *fitness* que indica sua aptidão em um determinado contexto, ou seja, descreve o quão boa é a solução que o indivíduo representa. Os AG's evoluem o conjunto inicial de indivíduos através da produção de novas gerações [Holland, 1975].

Os AG's clássicos normalmente funcionam da seguinte maneira: partindo de uma população inicial a cada iteração são aplicados operadores genéticos (cruzamento e mutação) visando a evolução da população através da criação de novas gerações. O cruzamento sempre ocorre, enquanto que a mutação, necessária para permitir a variabilidade, possui um pequena probabilidade de ocorrer. A população inicial pode ser criada de maneira aleatória, estática ou induzida através de alguma outra técnica computacional. AG's são capazes de encontrar boas soluções em problemas complexos, não necessariamente encontrando a solução ótima. Um dos fatores centrais dos AG's é a representação do cromossomo. Normalmente se utiliza um vetor binário onde cada posição, ou conjunto de posições, representa uma das características da solução/indivíduo. Entretanto, existem outros tipos de representação.

Random Keys Genetic Algorithms (RKGA) foram introduzidos por Bean [1994] e buscam melhorar a representação dos cromossomos nos problemas de sequenciamento, que em alguns casos, ao se aplicar os operadores genéticos gera filhos que não são soluções factíveis para o problema. Para isso, o RKGA representa os valores dos alelos dos cromossomos através de chaves aleatórias com valores reais no intervalo [0,1). Para a execução da função objetivo, os alelos, que contém os valores dos genes nos cromossomos, são decodificados para seus valores efetivos no espaço de soluções. No RKGA ao se gerar uma nova população é mantido o grupo dos indivíduos mais aptos, chamados elite, e o restante da população é substituída por novos indivíduos gerados através de cruzamentos e mutações.

No RKGA está presente o decodificador (*decoder*) que tem a função de traduzir os valores dos cromossomos e associar esses valores com algum valor dentro do alfabeto definido para o alelo do gene. As estratégias de decodificação podem variar de acordo com o problema e a abordagem utilizada, não sendo definida na literatura uma regra especifica para realizar a decodificação dos cromossomos. Com os valores decodificados dos indivíduos, é possível então aplicar a função objetivo e avaliar a aptidão de cada um.

O RKGA é considerado um algoritmo elitista, pois mantém parte dos indivíduos mais aptos da geração anterior. Entretanto para executar os operadores genéticos o RKGA utiliza indivíduos selecionados aleatoriamente da população o que pode comprometer a qualidade da evolução. O *Biased Random-Key Genetic Algorithm* (BRKGA) [Gonçalves & Resende, 2010] é uma variação do RKGA que difere na maneira como os indivíduos são selecionados para reprodução. No BRKGA um dos pais selecionados necessariamente pertence ao grupo elite. A Figura 2 [Gonçalves & Resende, 2009] representa um fluxograma do BRKGA. Grande parte do algoritmo é independente do problema o que facilita sua adaptação em diferentes tipos de problemas.

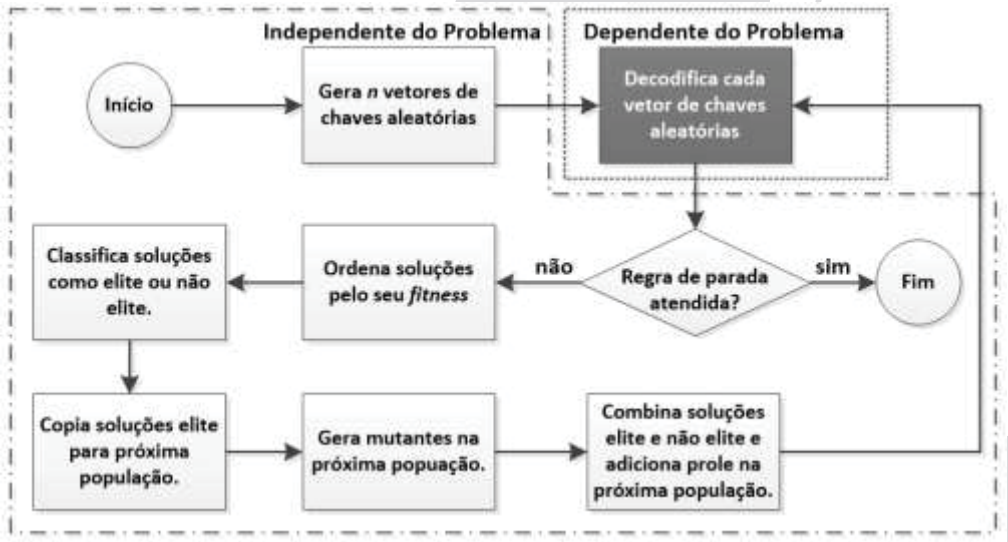

Figura 2. Fluxograma do BRKGA, adaptado de Gonçalves & Resende [2011].

O processo de evolução da população no BRKGA está ilustrado na Figura 3.

Figura 3. Evolução da população no BRKGA, adaptado de Gonçalves & Resende [2011].

É importante destacar que a população de indivíduos elite é mantida ao longo das gerações. Ao se realizar a operação de cruzamento, um dos pais sempre é selecionado do conjunto elite induzindo a busca de soluções próximas das melhores encontradas previamente sem que se perca a variabilidade, que é garantida pelos cruzamentos anteriores e pela geração de indivíduos mutantes.

O procedimento do BRKGA (Algoritmo 2) é semelhante ao de um AG clássico. A diferença é que para o cálculo da função objetivo primeiro é necessário a decodificação dos valores dos cromossomos e a escolha de um dos pais necessariamente seja feita a partir do conjunto elite.

Algoritmo 2. BRKGA

Data: i_{max}, *P*, *PFitness*, *C* ;
Result: Solução $y \in P$;
1.*geraPopulacaoInicial*(*P*, *PFitness*);
2.**for** (i = 1, ... , i_{max}) ***do***
3. *evoluiPopulacao*(*P*, *PFitness*);
4. *y*←*melhorSolucao*(*P*, *PFitness*);
5. **if** *y* antinge *C* **then**
6. **break**;
7. **end**
8.**end**

No algoritmo, i_{max} representa o número de gerações que o algoritmo vai gerar, *P* a população de indivíduos, *PFitness* um vetor contendo os respectivos valores de *fitness* da população *P* e *C* um critério de parada, por exemplo, se uma determinado valor da função objetivo foi atingido. A variável *y* conterá a melhor solução encontrada.

A função *geraPopulacaoInicial()* descrita na linha 1 tem como objetivo gerar de forma aleatória um conjunto inicial de possíveis soluções. Na linha 2 cria-se o número máximo de gerações a serem produzidas. A função *evoluirPopulacao()* aplica na população os operadores genéticos (cruzamento e mutação) com o objetivo de evoluir os indivíduos. Na linha 4 a função *melhorSolucao()* seleciona da população *P* o indivíduo *y* mais apto, ou seja, com o melhor valor de função objetivo. Se o indivíduo *y* atinge o critério de parada *C* (linha 5), a criação de novas gerações é interrompida.

Algoritmo 3. *evoluirPopulacao(P,PFitness)*

Data: *P*, *PFitness*, *P′*, *PFitness′*, *tamPop*, *tamElite*, *chanceMutacao*;
Result: *P*, *PFitness*;
1.**for** (*i*=0; *i*<*tamElite*) **do**
2. *P′*[*i*]←*P*[*i*] ;
3. *PFitness′*[*i*]←*PFitness*[*i*];
4.**end**
5.**for** (*i*=*tamElite*; *i*<*tamPop*) **do**
6. **if** (*valorAleatorio()*≤*chanceMutacao*) **do**
7. *P′*[*i*]←*novoInidividuoAleatorio()*;
8. **else do**

```
9.   x←selecionaindividuoElite(P);
10.  y←selecionaindivíduo(P);
11.  if(x = y) do
12.    while (x = y)
13.      y←selecionaIndividuo(P);
15.    end
16.    P′[i]←cruzaindivíduos(x, y);
17.  end
18.  PFitness′[i]←calculaFnObjetivo(P′[i]);
19.end
20.P ← P′;
21.PFitness ← PFitness′;
```

O Algoritmo 3 contém a função *evoluiPopulacao()*. Nas linhas 1 à 4 os indivíduos do conjunto elite são copiados da geração *k* para a geração *k*+1. A variável *tamElite* define a quantidade de indivíduos da população *P*. Para o restante da população (linhas 5 à 19), são aplicados os operadores genéticos. Na linha 6 é gerada variável aleatória no intervalo de [0,1), a partir da função *valorAleatorio()*, para verificar se o indivíduo será um mutante ou não, conforme a chance de mutação determinada. Como exemplo, se a *chanceMutacao* tem valor 0.5 e a função *valorAleatorio()* retorna 0.3, então o indivíduo será um mutante. Caso ocorra a mutação, serão atribuídos valores aleatórios para as características do indivíduo (linha 7), mas em caso contrário, ocorre a reprodução onde um deles é selecionado do conjunto *elite* (linha 9) e o outro é um indivíduo qualquer da população (linha 10). Nas linhas 11 à 15 é feita uma verificação para garantir que os pais são indivíduos diferentes na população. Por fim, na linha 16, um novo indivíduo é gerado através do cruzamento das características dos pais. A abordagem de cruzamento utilizada considera que para cada característica o indivíduo tem 50% de chance de herdá-la do pai e 50% de chance de herdá-la da mãe. Depois de gerado o novo indivíduo seu *fitness* é obtido (linha 18) através do cálculo da função objetivo (Algoritmo 4). A variável de entrada *x* é o indivíduo, *x′* recebe o indivíduo com os valores decodificados e *xfitness* armazena o valor de *fitness* do indivíduo.

Algoritmo 4. ***calculaFnObjetivo(x)***

```
Data: x, x′, xfitness;
Result: Valor xfitness;
1.x′← decodificaCromossomos(x);
2.xfitness ← calculaFnObjetivo(x′);
3.return xfitness;
```

No Algoritmo 4 é necessário decodificar cada indivíduo da população para realizar o cálculo da função objetivo (linha 1). Como a decodificação será feita depende do problema abordado e da estratégia utilizada. A partir da decodificação do cromossomo é possível calcular a função objetivo de *x′* através da função *calculaFnObjetivo(x′)* (linha 2), que no caso, consiste em executar o GRASP+PR com os parâmetros definidos em cada indivíduo. Por fim a linha 3 retorna o *xfitness* obtido a pelas medidas de avaliação do agrupamento gerado pelo GRASP+PR a partir dos parâmetros definidos.

Como estratégia de decodificação optou-se por utilizar o valor do alelo para indicar um valor no espaço de soluções, através da seguinte formulação:

$$A_{real} = \propto_{min} + floor(A_{cod} \times ((\propto_{max} - \propto_{\min}) + 1))$$

sujeito a:

$$\propto_{max} > \propto_{min} \quad \text{(I)}$$

Onde A_{real} é o valor do alelo no espaço de soluções do problema; A_{cod} é o valor codificado do alelo; $\propto_{min}$ e $\propto_{max}$ são respectivamente os limites inferiores e superiores de valores que A_{real} pode assumir. A restrição (I) garante que $\propto_{min}$ não seja maior que $\propto_{max}$.

5. Metodologia

Neste trabalho parte-se da hipótese que a aplicação do BRKGA na auto-parametrização do GRASP com *Path-Relinking* no agrupamento de dados pode contribuir para melhoria do

desempenho do algoritmo, diminuição do valor da função objetivo e/ou obtenção de agrupamentos mais coesos quando comparado a parametrização manual.

Para fins de análise foram utilizados o tempo de execução, *Time to Target Plot* (*tttplot*) [Aiex et al. 2007], função objetivo (FO) e *Corrected Rand Index* (CRand) [Hubert & Arabie, 1985] que compara os agrupamentos obtidos com os de referência fornecidos com o conjunto de dados. A validação dos experimentos foi feita através da comparação dos resultados obtidos utilizando BRKGA na auto-parametrização do algoritmo GRASP+PR nas variantes *forward* (GRASP+PRf), *backward* (GRASP+PRb), *mixed* (GRASP+PRm), *greedy randomized* (GRASP+PRgr) e *truncated* (GRASP+PRt) parametrizados manualmente.

O BRKGA foi implementado em ambiente *Microsoft Windows 8* utilizando a IDE *NetBeans 7.4*, linguagem *Java SE Ver. 7u45*, e utilização da biblioteca *Colt* ver.1.0.3 [Matsumoto & Nishimura, 1998] para geração de números aleatórios. Os experimentos foram realizados em um *notebook HP pavillion* g4-2220br, com um processador *Core i3-3110M CPU @ 2.40 GHz e 6* GB de memória RAM, O código do GRASP+PR segue o algoritmo utilizado por Frinhani et al. [2011].

A Tabela 1descreve os parâmetros que foram auto-parametrizados. A Tabela 2 descreve os conjuntos de dados utilizados. Com exceção do *Glass*, os resultados com parâmetros manuais foram obtidos de Frinhani et al. [2011]. Os experimentos foram baseados nos procedimentos adotados por Festa et al. [2010] que considera a construção de 30 gerações, cada uma com uma população de 15 indivíduos, sendo 30% da população definida para o conjunto *elite*, 20% para mutantes e 50% indivíduos fruto de cruzamento.

Tabela 1. Descrição dos parâmetros e respectivos limites.

Parâmetro	Descrição	Min	Máx
TP	Tamanho do *pool*. Número de soluções que o *Path-Relinking* irá armazenar no conjunto elite.	2	30
IPR	Número mínimo de soluções presentes no *pool* antes do início do *Path-Relinking*.	2	TP
DS	Diferença simétrica entre uma nova solução e uma solução do *pool* como critérios para que a nova solução possa ser adicionada ao *pool*.	2	50% do conjunto de dados.
ISM	Quantidade máxima de iterações sem melhorias do GRASP+PR como critério de parada.	2	150
BL	Máximo de iterações na fase de busca local.	2	300
NH	Porcentagem do tamanho da lista preliminar de arestas nos conjuntos de dados que contêm mais de 15 objetos.	10	50
QTD_GRUPOS	Quantidade de grupos utilizados.	2	30
VAR_PR	Variante do *Path-Relinking* utilizada (*Forward, Backward, Mixed, Greedy Randomized, Truncated*).	0	4

No cálculo da função objetivo, diferente da estratégia utilizada por Festa et al. [2010], onde para cada indivíduo o GRASP+PR é executado 30 vezes, optou-se por uma estratégia baseada em uma quantidade incremental de iterações. Na primeira geração foi feita 1 iteração do GRASP+PR para cada indivíduo, na segunda 2 iterações e assim sucessivamente até o limite de 30 iterações. O objetivo foi relaxar a construção das gerações iniciais. Experimentos mostraram uma significativa redução do tempo de execução sem comprometimento dos valores.

A distância *City Block* (6) foi utilizada para o cálculo da similaridade entre os objetos, onde, d_{ij} é a similaridade entre os objetos *i* e *j*, *L* é a quantidade de objetos do conjunto de dados, a_{ik} e a_{jk} são respectivamente o *k*-ésimo atributo dos *i*-ésimo e *j*-ésimo objetos a_i e a_j.

$$d_{ij} = \sum_{k=1}^{L} |a_{ik} - a_{jk}| \qquad (6)$$

O critério de melhoria do BRKGA no decorrer das iterações do GRASP+PR foi baseado na seguinte prioridade: (I) maior valor médio do CRand, (II) maior valor médio da função objetivo e (III) menor tempo de execução.

Tabela 2. Conjuntos de dados utilizados e respectivas características

Conjunto de Dados	#Objetos	#Atributos	Referência
Protein	694	128	[Ding & Dubchak, 2001]
BreastB2	49	1213	[Monti et al., 2005]
DLBCLA	141	661	[Monti et al., 2005]
MultiA	103	5565	[Su et al., 2002]
Novartis	103	1000	[Su et al., 2002]
Glass	214	9	[Evett & Spiehler, 1987]

6. Análise dos Resultados e Discussão

A Tabela 3 contém os resultados, os melhores valores estão em negrito. A primeira linha de cada conjunto de dados contém os valores obtidos pelo BRKGA e variante PR que contribuiu para os melhores resultados. A tabela contém a quantidade de grupos encontrados; maior valor, média e desvio padrão do CRand; média e desvio padrão da função objetivo e tempo.

Tabela 3. Resultados obtidos nos experimentos

Algoritmo	Grupos	CRand			Função Objetivo		Tempo	
		Máx.	Média	DP	Média	DP	Média	DP
BreastB2								
BRKGA+GRASP+PRgr	5	**0.531**	**0.442**	0.0541	102710.0	**187.105**	0.3649	0.4008
GRASP+PRb	7	0.380	0.289	**0.0441**	**68673.5**	244.458	0.2301	0.2490
GRASP+PRf	7	0.344	0.250	0.0531	69222.1	367.848	0.1423	0.1532
GRASP+PRgr	7	0.368	0.243	0.0560	69764.6	549.789	0.1155	0.1236
GRASP+PRm	7	0.368	0.245	0.0533	69841.5	474.026	**0.1060**	**0.1114**
GRASP+PRt	7	0.344	0.250	0.0531	69222.1	367.848	0.1618	0.1747
Novartis								
BRKGA+GRASP+PRm	4	**0.950**	**0.950**	**0.0000**	**1033080.0**	**0.000**	**0.2675**	**0.2755**
GRASP+PRb	4	**0.950**	0.883	0.0733	1034690.0	1777.180	3.0977	3.1549
GRASP+PRf	4	**0.950**	0.880	0.0734	1034740.0	1782.020	3.1718	3.2332
GRASP+PRgr	4	**0.950**	0.880	0.0734	1034740.0	1782.020	3.1363	3.1995
GRASP+PRm	4	**0.950**	0.880	0.0734	1034740.0	1782.020	3.0830	3.1440
GRASP+PRt	4	**0.950**	0.880	0.0734	1034740.0	1782.020	3.0891	3.1481
MultiA								
BRKGA+GRASP+PRt	2	0.899	**0.899**	**0.0000**	**2.14748E9**	**0.000**	**2.2922**	**2.7416**
GRASP+PRb	4	**0.924**	0.765	0.1327	1.48953E9	2.36493E7	2.9914	3.0420
GRASP+PRf	4	**0.924**	0.770	0.1324	1.48960E9	2.40648E7	2.9193	2.9685
GRASP+PRgr	4	**0.924**	0.770	0.1324	1.48960E9	2.40648E7	2.9791	3.0335
GRASP+PRm	4	**0.924**	0.770	0.1324	1.48960E9	2.40648E7	2.9932	3.0476
GRASP+PRt	4	**0.924**	0.770	0.1324	1.48960E9	2.40648E7	3.1391	3.2049
DLBCLA								
BRKGA+GRASP+PRt	3	0.838	**0.838**	**0.0000**	**4.60538E8**	**0.000**	**0.9412**	**0.9910**

GRASP+PRb	3	**0.855**	0.717	0.1681	4.61203E8	1271360.0	4.8394	4.9268
GRASP+PRf	3	**0.855**	0.713	0.1695	4.61225E8	1287540.0	4.9072	5.0052
GRASP+PRgr	3	**0.855**	0.713	0.1695	4.61225E8	1287540.0	4.8122	4.9065
GRASP+PRm	3	**0.855**	0.713	0.1695	4.61225E8	1287540.0	4.8239	4.9166
GRASP+PRt	3	**0.855**	0.713	0.1695	4.61225E8	1287540.0	4.8113	4.9026
Glass								
BRKGA+GRASP+PRb	6	**0.265**	**0.260**	**0.0009**	**10173.6**	**1.698**	**12.5009**	**13.6888**
GRASP+PRb	6	0.260	0.237	0.0271	10225.4	101.607	17.3256	19.4522
GRASP+PRf	6	0.260	0.238	0.0275	10225.6	103.402	15.0709	16.7868
GRASP+PRgr	6	0.260	0.238	0.0275	10225.6	103.402	16.1420	18.1358
GRASP+PRm	6	0.260	0.238	0.0275	10225.6	103.402	17.2975	19.4183
GRASP+PRt	6	0.260	0.238	0.0275	10225.6	103.402	15.4967	17.2195
Protein								
BRKGA+GRASP+PRgr	3	**0.370**	**0.300**	**0.0138**	**5.71928E7**	**23352.2**	404.7671	424.6223
GRASP+PRb	5	0.310	0.273	0.0276	3.20945E7	38909.6	356.8755	362.6257
GRASP+PRf	5	0.310	0.273	0.0280	3.20945E7	39063.4	**348.4372**	**354.2628**
GRASP+PRgr	5	0.310	0.273	0.0280	3.20945E7	39063.4	363.0760	369.7144
GRASP+PRm	5	0.310	0.273	0.0280	3.20945E7	39063.4	402.9154	410.9431
GRASP+PRt	5	0.310	0.273	0.0280	3.20945E7	39063.4	348.5078	354.3282

Com relação ao CRand é possível observar melhorias nos resultados das versões auto-parametrizadas nos conjunto de dados Glass, Protein e BreastB2. Um destaque para o conjunto de dados BreastB2 que apresentou um aumento significativo do valor do CRand. Mesmo não alcançando o valor máximo nos conjunto de dados Novartis, MultiA e DLBCLA, as versões auto-parametrizadas obtiveram os melhores valores médios e desvio padrão em comparação as versões parametrizadas manualmente, indicando uma maior robustez do método.

A análise da função objetivo permite verificar se ocorreram melhorias na coesão dos agrupamentos gerados. Nesta métrica as versões auto-parametrizadas contribuíram para melhores resultados em cinco dos seis conjunto de dados analisados. A única exceção foi o BreastB2 que apresentou um maior valor de função objetivo justificado pela menor quantidade de grupos encontrados. Ainda em relação a função objetivo, a redução do desvio padrão foi observada em todos os conjuntos de dados na versão auto-parametrizada.

Com relação ao tempo de execução, as versões auto-parametrizadas obtiveram redução dos valores nos conjunto de dados Novartis, MultiA, DLBCLA e Glass, e respectivas reduções dos desvios padrão. No conjunto de dados Novartis a redução do tempo de execução foi considerável. Melhorias no tempo também podem ser observadas nos *tttplots* dos conjunto de dados Novartis e DLBCLA, Figuras 4 e 5 respectivamente, que apresentaram uma melhora significativa no tempo de convergência do algoritmo para encontrar a função objetivo *target* definida. Como valor *target* para o *tttplot*, considerou-se o maior valor da função objetivo obtido em cada conjunto de dados.

A análise dos dados nos permite confirmar a hipótese que a auto-parametrização do GRASP+PR no agrupamento de dados através BRKGA contribui para melhoria dos resultados quando comparados aos algoritmos parametrizados manualmente. Pesquisas do uso de métodos estatísticos na auto-parametrização do GRASP+PR no agrupamento estão sendo concluídas e possibilitarão uma comparação com os métodos apresentados neste trabalho.

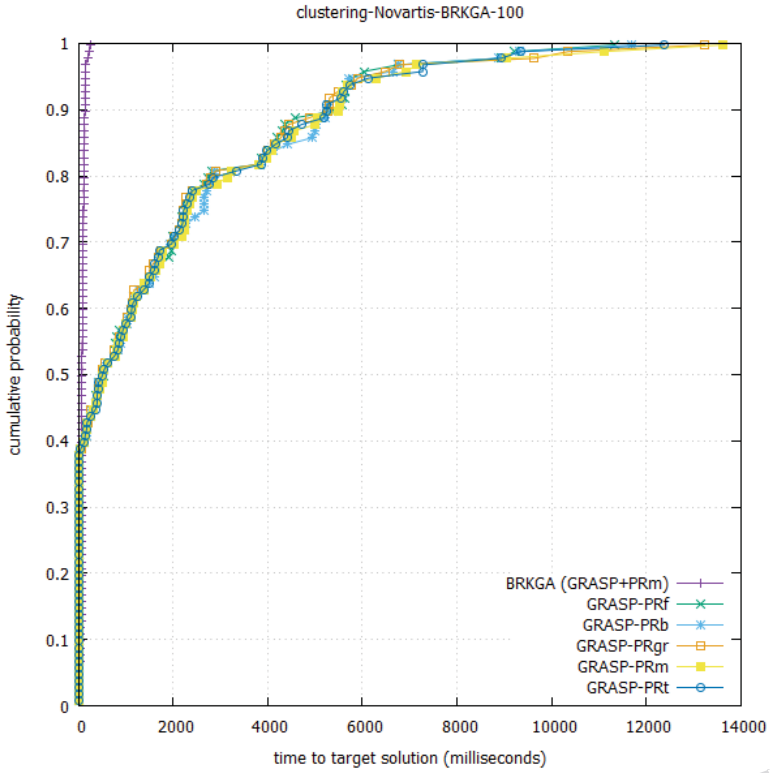

Figura 4. *tttplot* do conjunto de dados Novartis.

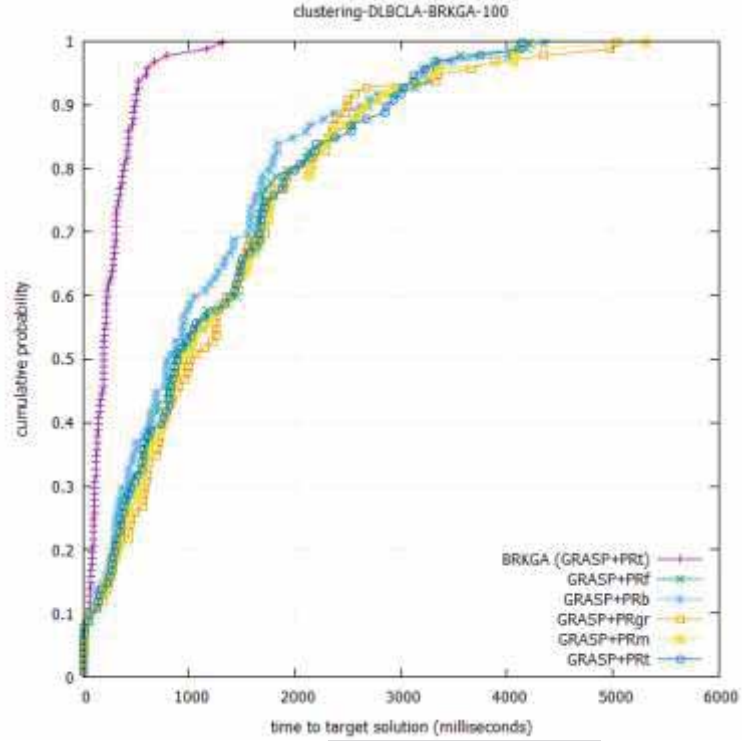

Figura 5. *tttplot* do conjunto de dados DLBCLA

Referências

Aiex, R. M., Resende, M. G., & Ribeiro, C. C. (2007). *"TTT plots: a perl program to create time-to-target plots"*. Optimization Letters 1.4, páginas 355-366.

Anderberg, M. R. (1973*). "Cluster analysis for applications"* (No. OAS-TR-73-9). *Office of the Assistant for Study support Kirtland AFB N Mex.*
Bean, J. C. (1994). *"Genetic algorithms and random keys for sequencing and optimization"*. ORSA journal on computing 6.2, páginas 154-160.
Bennett, K. P., & Mangasarian, O. L. (1992*). "Robust linear programming discrimination of two linearly inseparable sets"*. Optimization methods and software 1.1, páginas 23-34.
Berkhin, P. (2006). *"A survey of clustering data mining techniques"*. Grouping Multidimensional Data, páginas 25-71. Springer Berlin Heidelberg.
Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). *"A Racing Algorithm for Configuring Metaheuristics"*. GECCO Vol. 2, páginas 11-18.
Birattari, M. (2009*). "Tuning metaheuristics: a machine learning perspective"*. Vol. 197. Berlin: Springer.
Braga, L. P. V. (2005). "Introdução à Mineração de Dados - 2ª edição: Edição ampliada e revisada". Editora E-papers.
Braga, A. D. P., Carvalho, A. C. P. L. F., & Ludermir, T. B. (2000). "Redes neurais artificiais: teoria e aplicações". Livros Técnicos e Científicos.
Brown, D. E., & Huntley, C. L. (1992). *"A practical application of simulated annealing to clustering"*. Pattern Recognition 25.4, páginas 401-412.
Ding, C. H., & Dubchak, I. (2001). *"Multi-class protein fold recognition using support vector machines and neural networks"*. Bioinformatics 17.4, páginas 349-358.
Dobslaw, F. (2010). *"A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks"*. Proceeding of the International Conference on Computer Mathematics and Natural Computing 2010. WASET.
Evett, I. W., & Spiehler, E. J. (1987). *"Rule induction in forensic science"*. KBS in Goverment, Online Publications, páginas 107-118.
Feo, T. A., & Resende, M. G. (1995). *"Greedy randomized adaptive search procedures"*. Journal of global optimization 6.2, páginas 109-133.
Festa, P., Gonçalves, J. F., Resende, M. G., & Silva, R. M. (2010). *"Automatic tuning of GRASP with path-relinking heuristics with a biased random-key genetic algorithm"*. Experimental Algorithms, páginas 338-349. Springer Berlin Heidelberg.
Frinhani, R. M. D.; Silva, R. M. A. & Mateus, G. R. (2011). "GRASP com Path-Relinking para agrupamento de dados biológicos". Dissertação de Mestrado. Universidade Federal de Minas Gerais, Departamento de Ciência da Computação.
Glover, F., Laguna, M., & Martí, R. (2000). *"Fundamentals of scatter search and path relinking"*. Control and cybernetics 39.3, páginas 653-684.
Glover, F., & Marti, R. (2006). *"Tabu search"*. Metaheuristic Procedures for Training Neutral Networks, páginas 53-69. Springer US.
Gonçalves, J. F., & Resende, M. G. (2011). *"Biased random-key genetic algorithms for combinatorial optimization"*. Journal of Heuristics, 17.5, páginas 487-525.
Hall, L. O., Ozyurt, I. B., & Bezdek, J. C. (1999). *"Clustering with a genetically optimized approach"*. Evolutionary Computation, IEEE Transactions on 3.2, páginas 103-112.
Hartigan, J. A., & Wong, M. A. (1979*). "Algorithm AS 136: A k-means clustering algorithm."* Applied statistics, páginas 100-108.
Herrero, J., Valencia, A., & Dopazo, J. (2001). *"A hierarchical unsupervised growing neural network for clustering gene expression patterns"*. Bioinformatics 17.2, páginas 126-136.
Holland, J. H. (1975). *"Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence"*. Oxford, England: Michigan Press.
Hoshida, Y., Brunet, J. P., Tamayo, P., Golub, T. R., & Mesirov, J. P. (2007). *"Subclass mapping: identifying common subtypes in independent disease data sets"*. PloS one 2.11, e1195.
Hubert, L. & Arabie, P. (1985). *"Comparing partitions"*. Journal of Classification, 2:193-218.
Jain, A. K., & Dubes, R. C. (1988). *"Algorithms for clustering data."* Prentice-Hall, Inc.

Jain, A., Murty, M. and Flynn, P. (1999). *"Data clustering: A review".* ACM Computing Surveys (CSUR) 31.3, páginas 264-323.
Kaufman, L., & Rousseeuw, P. J. (1990). *"Partitioning around medoids (program PAM)."* Finding groups in data: an introduction to cluster analysis, páginas 68-125.
López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., & Birattari, M. (2011). *"The irace package, iterated race for automatic algorithm configuration"*. IRIDIA, Université Libre de Bruxelles, Tech. Rep. TR/IRIDIA/2011-004.
Matsumoto, M., & Nishimura, T. (1998). *"Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator"*. ACM Transactions on Modeling and Computer Simulation (TOMACS) 8.1, páginas 3-30.
Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003*). "Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data"*. Machine learning, 52.1-2, páginas 91-118.
Monti, S., Savage, K. J., Kutok, J. L., Feuerhake, F., Kurtin, P., Mihm, M., ... & Shipp, M. A. (2005). *"Molecular profiling of diffuse large B-cell lymphoma identifies robust subtypes including one characterized by host inflammatory response"*. Blood 105.5, páginas 1851-1861.
Nascimento, M. C., Toledo, F., & de Carvalho, A. C. (2010). *"Investigation of a new GRASP-based clustering algorithm applied to biological data"*. Computers & Operations Research 37.8, páginas 1381-1388.
Su, A. I., Cooke, M. P., Ching, K. A., Hakak, Y., Walker, J. R., Wiltshire, T., ... & Hogenesch, J. B. (2002). *"Large-scale analysis of the human and mouse transcriptomes"*. Proceedings of the National Academy of Sciences 99.7, páginas 4465-4470.
Silva, J. C. ; Frinhani, R. M. D. ; Silva, Ricardo Martins Abreu ; Mateus, Geraldo Robson . Auto-parametrização do GRASP com Path-Relinking no agrupamento de dados com F-Race e iterated F-Race. Anais do 11º Simpósio Brasileiro de Sistemas de Informação. Goiânia/GO: Instituto de Informática Universidade Federal de Goiás, 2015. v.1. p.47-54.