Auto-parametrização do GRASP com *Path-Relinking* no agrupamento de dados com *F-Race e iterated F-Race*

Alternative Title: Automatic Tuning of GRASP with Path-Relinking in data clustering with F-Race and iterated F-Race

Júlio Cesar da Silva
Universidade Federal de Itajubá
Instituto de Matemática e Computação
Av. BPS, 1303 - Pinheirinho, Itajubá/MG. CEP 37500-903.
+55 35 9812-2582
julio_sxs@hotmail.com

Rafael de M. D. 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

Ricardo Martins A. 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

Em estudos que utilizam metaheurísticas embora os parâmetros de entrada influenciem diretamente o desempenho do algoritmo sua definição na maioria das vezes é feita manualmente levantando questões sobre a qualidade dos resultados. Este artigo se propõe a aplicar o I/F-Race na auto-parametrização do GRASP com *Path-Relinking* no agrupamento de dados visando obter melhores resultados em relação aos parametrizados manualmente. Experimentos realizados com cinco conjuntos de dados demostraram que a utilização do I/F-Race contribui para obtenção de melhores resultados que a parametrização manual.

Palavras-Chave

Auto-parametrização, I/F-Race, GRASP, *Path-Relinking*, Agrupamento de Dados.

ABSTRACT

In studies that use metaheuristics although the input parameters directly influence the performance of the algorithm its definition is mostly done manually raising questions about the quality of the results. This paper aims to apply the F/I-Race in the self parameterization of GRASP with Path-Relinking in the data clustering in order to obtain better results than the manually tuned algorithms. Experiments performed with five data sets showed that the use of I/F-race contributed to achievement best results than manual tuning.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: *Problem Solving, Control Methods, and Search - Heuristic methods*.

General Terms

Performance, Design, Experimentation.

Keywords

Automatic Tunig, I/F-Race, GRASP, *Path-Relinking, Data Clustering*.

1. INTRODUÇÃO

A análise de agrupamentos *(clustering)* é uma tarefa não supervisionada cujo propósito é a separação de objetos com características similares em grupos a partir de uma métrica. A quantidade de objetos e de grupos necessários para uma caracterização coerente do conjunto de dados tornam a tarefa de agrupamento um complicado problema combinatório. O ideal é que as soluções obtidas por um algoritmo de agrupamento apresentem alta homogeneidade entre os objetos de um mesmo grupo e alta heterogeneidade entre os grupos. Na literatura é possível encontrar aplicações do agrupamento de dados na mineração de dados textuais [30], mensuração e detecção de alterações da expansão urbana de regiões [36], séries temporais de dados médicos [42], agrupamento de sistemas orientados a objetos [44], entre outras.

Existe uma grande variedade de algoritmos de agrupamento cada um possuindo vantagens e desvantagens frente a tipos específicos de dados. Em diversos trabalhos as metaheurísticas têm sido utilizadas de forma isolada ou complementar aos algoritmos clássicos de agrupamento (ex. *k-means* [23], *k-medians* [26] e PAM (*Partitioning Around Medoids)* [27]), com objetivo de melhorar o desempenho e/ou precisão dos agrupamentos. Exemplos de metaheurísticas aplicadas no agrupamento de dados inclui *Simulated Annealing* [7], Algoritmos Genéticos (AG) [22], Redes Neurais Artificiais (RNAs) [12], *Fuzzy Clustering Neural Network Architecture* [37], GRASP em bases biológicas [35].

Em um dos trabalhos mais recentes a abordagem híbrida GRASP com *Path-Relinking* (GRASP+PR) proposta por [15] apresentou melhores resultados que o GRASP na sua versão tradicional. Entretanto, como em outras metaheurísticas, a definição dos parâmetros de execução do GRASP+PR é uma tarefa complexa e na maioria das vezes sua escolha é feita empiricamente. Essa abordagem geralmente resulta em uma configuração não otimizada, com possibilidades de não se explorar todas as potencialidades do algoritmo e abre margens para se questionar se os parâmetros escolhidos são os que realmente trarão melhores resultados. Nesse contexto a auto-parametrização *(automatic tuning)* de metaheurísticas, como o GRASP+PR, pode trazer vantagens como redução do tempo computacional e/ou permitir ao algoritmo alcançar melhores soluções que as versões parametrizadas manualmente.

O presente trabalho tem como objetivo aplicar a abordagem *racing* através dos métodos *F-Race* e *Iterated F-Race (Irace)* na auto-parametrização do GRASP+PR no problema de agrupamento

de dados. Partimos da hipótese que sua aplicação pode contribuir para melhoria do desempenho do algoritmo, redução do valor da função objetivo e/ou obtenção de agrupamentos mais coesos quando comparado a parametrização manual.

Este trabalho está organizado como se segue: A seção 2 apresenta a modelagem utilizada. A seção 3 descreve as metaheurísticas GRASP e GRASP+PR e sua aplicação no agrupamento de dados. A seção 4 descreve técnicas de auto-parametrização de metaheuristicas com maior destaque nos métodos *racing (F-Race e Irace)* que são foco de estudo desse trabalho. A seção 5 contém a metodologia utilizada e por fim a seção 6 contém os resultados dos experimentos e discussões.

2. MODELAGEM

O modelo utilizado neste trabalho segue o proposto por [35] que considera o conjunto de dados como um único grafo totalmente conexo. A distância entre cada par de objetos, obtida a partir dos atributos de cada objeto, é representada por uma aresta valorada.

Durante a execução do procedimento as arestas vão sendo gradualmente eliminadas de modo a formar cliques que representam os agrupamentos. O objetivo desta estratégia é minimizar a soma total das distâncias entre os objetos do mesmo grupo (intra-grupo) considerando que quanto menor for a distância entre dois objetos, maior a similaridade entre eles. 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

GRASP [13] é uma metaheurística que combina um método construtivo com busca local em um processo iterativo e independente utilizando o conceito multi-inícios. Em sua primeira fase, chamada fase construtiva, o GRASP utiliza as abordagens gulosa e aleatória para gerar soluções iniciais a partir de um processo semiguloso, e em uma segunda fase, chamada fase exploratória ou busca local, aplica em cada solução previamente construída uma busca local com o objetivo de melhorar os resultados. GRASP foi aplicado com sucesso na resolução de diversos problemas de otimização combinatória [29, 41, 2]. [35] utilizaram o GRASP no agrupamento de dados e encontraram melhores resultados em comparação a métodos clássicos (*k-means*, *k-medians* e PAM).

No GRASP as soluções encontradas em iterações anteriores não influenciam o algoritmo na solução atual, ou seja, o funcionamento do GRASP tradicional não se baseia em aprendizagem sobre a sua execução. Porém é possível melhorar o GRASP através da utilização de mecanismos de memória que permitam um direcionamento do algoritmo para as soluções mais promissoras encontradas previamente.

Path-Relinking [17, 18, 19] é um procedimento de intensificação de busca que tem sido aplicado com sucesso na incorporação de memória ao GRASP. O método trabalha explorando as trajetórias que conectam soluções de alta qualidade (soluções elite), obtidas durante as execuções do GRASP. O procedimento parte de uma solução inicial e gera um caminho de soluções no espaço de vizinhança que conduz a uma solução guia. O objetivo é encontrar soluções diversificadas e de qualidade entre duas soluções elite.

Existem variações do *Path-Relinking* que possuem estratégias diferentes na determinação da solução inicial/guia, bem como a trajetória a ser seguida [39, 40, 2]. Na variante *Forward* a pior solução em termos de função objetivo é tomada como solução inicial e a melhor como guia. Na *Backward* considera-se a pior solução como guia e a melhor como solução inicial. Na *Mixed* o início da trajetória é alternada entre a melhor solução e a pior até que seja encontrada uma solução intermediária. Com a *Greedy Randomized Adaptive* uma solução guia é selecionada aleatoriamente de uma lista de candidatos mais promissores. Por fim, a *Truncated* explora uma parte da trajetória entre as soluções.

Algoritmo 1: GRASP+PR()

```
Data: I,M,TP,p,DS,Ism,DC,S;
Result: Solução x*∈X;
1.P:=∅;
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∈P;
7.     x':= PathRelinking(x',y);
8.     x':= FaseBuscaLocal(x');
9.     if P estiver completo then
10.      if c(x')≤max{c(x)}|x∈P} and x'≉P then
11.        Substitui elemento mais similar a x'
           entre todos com custo pior que x';
12.      end
13.    else if x'≉P then
14.      P:=P∪{x'};
15.    end
16.  else if x'≉P and P=∅ then
17.    P:=P ∪{x'};
18.  end
19. end
20.end
21.return x*
```

O Algoritmo 1 [15] descreve o GRASP+PR no agrupamento de dados, que toma como entrada o conjunto de dados I e os parâmetros: M que é a quantidade agrupamentos, TP é o tamanho do conjunto de soluções elite P, p é a quantidade mínima de elementos em P antes do início do PR, DS é a diferença simétrica entre a solução inicial e a guia, Ism é a quantidade de iterações sem melhoria como critério de parada do algoritmo, DC é a

medida de similaridade entre os objetos do conjunto de dados e *S* a semente utilizada pelo método aleatório.

Após inicializar o conjunto elite *P* como vazio (linha 1), o GRASP+PR é executado nas linhas 2 a 19 até que um critério de parada (*Ism*) seja satisfeito. Em cada iteração uma solução x' é gerada pela fase construtiva na 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. A linha 5 realiza 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 PR com a solução x' (linhas 6 e 7).

Após a realização do *Path-Relinking*, a solução x' passa novamente pela Fase Busca Local visando melhorias (linha 8). Caso o conjunto elite esteja completo é verificado se a solução é suficientemente diferente das soluções atualmente presentes em *P* ($x' \napprox P$). 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 desse conjunto 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 $x' \napprox P$. Ao final, o algoritmo retorna x^* que armazena a melhor solução encontrada. Detalhes da *FaseConstrutiva*(), *FaseBuscaLocal*() e *PathRelinking*() podem ser obtidos em [15].

4. AUTO-PARAMETRIZAÇÃO DE METAHEURÍSTICAS

A parametrização de metaheurísticas, como GRASP+PR, envolve a definição de valores que serão utilizados como parâmetros de entrada para execução do algoritmo. Geralmente existe uma grande variação desses valores caracterizando esta tarefa como um típico problema combinatório. Na maioria dos trabalhos envolvendo metaheurísticas a definição dos parâmetros é realizada de forma manual e empírica levantando questões sobre o desempenho do algoritmo a partir dos parâmetros escolhidos, ou seja, se realmente são os que trarão melhores resultados. A auto-parametrização de metaheurísticas (*Automatic Tuning)* tem recebido atenção por possibilitar a obtenção de melhores soluções em comparação a parametrização manual.

[11] apresentou um *framework* para auto-parametrização de metaheurísticas combinando projetos de experimentos e Redes Neurais Artificiais. [14] utilizaram a variante do algoritmo genético, *Biased Random Key Genetic Algorithm* (BRKGA) [21], para auto-parametrização do GRASP com *Path-Relinking* no *Generalized Quadratic Assignment Problem* (GQAP). [6] realizou a auto-parametrização das metaheuristicas Colônia de formigas e Busca Local iterativa utilizando *F-Race*. [31] implementaram o *iterated F-Race (Irace)* visando corrigir falhas do F-Race. *I/F-Race* utilizam uma abordagem estatística para selecionar a melhor configuração num conjunto de configurações candidatas [4].

4.1 A abordagem *racing*

De acordo com [5] a abordagem *racing* teve inicio quando [32] tiveram problemas com a seleção de modelos para aprendizado de máquina. [4] adaptaram essa abordagem para empregá-la em trabalhos envolvendo a configuração de algoritmos. Essa abordagem consiste na execução de várias etapas para avaliação de um conjunto finito de configurações candidatas, onde após testes estatísticos em cada uma destas etapas, as configurações menos eficientes são descartadas permitindo desta maneira concentrar os cálculos nas configurações candidatas mais promissoras.

A abordagem *racing* considera uma sequência de instâncias i_k, com k=1, 2, 3, ..., onde cada instância representa um conjunto de parâmetros gerados de forma aleatória. Tem-se como c_k^{θ} o custo de um conjunto de parâmetros θ em uma instância i_k. A avaliação dos conjuntos de parâmetros candidatos é realizada de forma incremental de modo que no passo *k* obtém-se um vetor de custos (6) que serão utilizados para avaliar θ.

$$c_k(\theta)=(c_1^{\theta}, c_2^{\theta}, \ldots, c_k^{\theta}) \qquad (6)$$

Uma matriz de custos é obtida unindo-se c_k^{θ} aos vetores de custos finais de uma etapa anterior, dados por $c^{k-1}(\theta)$. Em seguida é gerada a próxima sequência de conjunto de parâmetros candidatos, onde θ_k é o conjunto dos conjuntos de parâmetros sobreviventes antes do passo *k* (7).

$$\theta_0 \supseteq \theta_1 \supseteq \theta_2 \supseteq \ldots, \qquad (7)$$

4.1.1 F-RACE

O *F-Race* [4] utiliza uma abordagem *racing* baseada no teste de Friedman [8] que analisa e classifica os custos levando em consideração a variância encontrada. De acordo com [5] o funcionamento do *F-Race* supõe que estão no processo $m=|\theta_{k-1}|$ conjuntos de parâmetros candidatos. O teste de Friedman assume que os *k* custos observados, são agrupados em b_i blocos independentes (8), com i = 1, 2, ….

$$\begin{aligned} b_1&=\left(c_1^{\theta_{v1}}, c_1^{\theta_{v2}}, \ldots, c_1^{\theta_m}\right)\\ b_2&=\left(c_2^{\theta_{v1}}, c_2^{\theta_{v2}}, \ldots, c_2^{\theta_m}\right)\\ &\vdots\\ b_k&=\left(c_k^{\theta_{v1}}, c_k^{\theta_{v2}}, \ldots, c_k^{\theta_m}\right) \end{aligned} \qquad (8)$$

Nos blocos, os custos c_l^{θ} são organizados em ordem crescente e para cada conjunto de parâmetros $\theta_{v_j} \epsilon\, \theta_{k-1}$, R_{lj} é a classificação de θ_{v_j} no bloco b_l, e $R_j = \sum_{l=1}^{k} R_{lj}$ é a soma da classificação de θ_{v_j}, sobre todas as instâncias i_l, com $1 \le l \le k$. O teste de Friedman utilizado está descrito em (9) [8]:

$$T=\frac{(m-1)\sum_{j=1}^{m}\left(R_j-\frac{k(m+1)}{2}\right)^2}{\sum_{l=1}^{k}\sum_{j=1}^{m}R_{lj}^2-\frac{km(m+1)^2}{4}} \qquad (9)$$

Sob a hipótese nula de que todos os candidatos são equivalentes, *T* é aproximadamente X^2 de uma distribuição com *m-1* graus de liberdade [38]. Se o valor observado de *T* é maior do que $1-\propto$ desta distribuição, onde $\propto$ é o nível de confiança escolhido, a hipótese nula é rejeitada o que indica que ao menos um dos conjuntos de parâmetros candidatos apresenta melhor desempenho do que as demais. Após o teste de hipótese, o teste *post hoc* de Friedman (10) é utilizado visando encontrar o conjunto de parâmetros mais promissor e quais deverão ser eliminados [8].

Se o *F-Race* não rejeita a hipótese nula da comparação, então todas as configurações candidatas de θ_{k-1} passam para θ_k. Os demais conjuntos de parâmetros que não possuem um bom nível de significância são descartados e não mais aparecem em θ_k.

$$\frac{|R_j - R_{jh}|}{\sqrt{\frac{2k(1-\frac{T}{k(m-1)}(\sum_{l=1}^{k}\sum_{j=1}^{m}R_{lj}^2 - \frac{km(m+1)^2}{4}))}{(k-1)(m-1)}}} > t_{1-\alpha/2}\,, \quad (10)$$

O Algoritmo 2 descreve o método *F-Race* [3] que toma como entrada o conjunto de configurações C que serão executadas no conjunto de dados *I e* a quantidade mínima de custos a serem analisados. Tem como resultado o melhor conjunto de parâmetros. Na linha (1) são inicializados os atributos que fornecem a melhor configuração e o número mínimo de custos. Na estrutura de repetição da linha (2) são executados todos os conjunto de parâmetros no conjunto de dados. Já na linha (6) é utilizado o teste de Friedman com base na classificação dos custos para um número de conjuntos de parâmetros independentes ocorridos até o momento e um número de conjuntos de parâmetros ainda no processo. Em seguida na linha (10) são realizadas comparações em pares com os melhores conjuntos de parâmetros até o momento. E por fim na linha (12) são eliminados os conjuntos de parâmetros não promissores. Este procedimento deve ser executado até que exista apenas um conjunto de parâmetros ou quando se atinja uma quantidade máxima de iterações definidas pelo usuário.

Algoritmo 2: *F-Race()*

```
Data: Conjunto de Configurações C;
      Conjunto de Dados I;
      Número de Conjuntos Min ni_min;
Result: Melhor configuração C*;
1. C*:=C; ni:=0;
2. repeat
3.  rodar as configurações de C* em i;
4.  ni:=ni+1;
5.  if ni ≥ ni_min then
6.   Executar o Teste de Friedman
7.  if for indicado uma diferença de
       Desempenho significante then
8.     c*:= retornar melhor Configuração em C*
9.   for c ∈ C*\{c*} do
10.     Executar o  teste pairwise Friedman
        post hoc em c e c*;
11.     if for encontrado um diferença de
         desempenho significante then
12.      eliminar c de C*;
13.     end if;
14.   end for;
15.  end if;
16. end if;
17.until a condição seja satisfeita;
18.return C*
19.end
```

4.1.2 IRACE

[24] mostra que uma grande limitação do *F-Race* é que nos passos iniciais todas as configurações precisam ser avaliadas. Esta característica do *F-Race* limita consideravelmente o tamanho do conjunto de parâmetros para que o procedimento possa ser aplicado de forma eficaz, especialmente quando se trata de um conjunto de parâmetros que contêm diversas combinações de valores.

Iterated F-Race (Irace) [5] é uma variante do *F-Race* que tem o objetivo de resolver os problemas da sua limitação. *Irace* utiliza o procedimento do *F-Race* descrito na seção 4.1.1 como uma sub-rotina, onde em cada iteração uma quantidade de conjuntos de parâmetros sobrevive servindo para gerar uma amostragem de novos conjuntos de parâmetros. Espera-se deste modo, gerar amostras dos conjuntos de parâmetros mais promissores. *Irace* segue diretamente uma estrutura de pesquisa baseado em modelo que normalmente é aplicado em três etapas [43]: (1) Geração de amostras de conjuntos de parâmetros de acordo com uma distribuição específica, (2) Seleção dos melhores conjuntos obtidos por meio do algoritmo *Irace*, e (3) Atualização das amostras removendo conjuntos de parâmetros de pior desempenho de modo a enviesá-las. Estes três passos são repetidos até que um critério predefinido seja satisfeito.

O Algoritmo 3 descreve o método *Irace*. Na linha (1) tem-se o conjunto de configurações que receberá os melhores conjuntos de parâmetros candidatos encontradas pelo *F-Race*. Na linha (3) são gerados novos conjuntos baseados nas melhores configurações encontradas pelo processo do *F-Race* até o momento. Já na linha (7) é retornada o melhor conjuntos de parâmetros do conjunto elite. Este procedimento continuará até atingir um número mínimo de conjuntos de parâmetros sobreviventes ou um número máximo de iterações.

Algoritmo 3: *Irace()*

```
Data: Conjunto de Configurações C;
Result: Conjunto Elite C*;
1. C':=0; //C' é inicializado a partir de uma
          //execução do F-Race
2. repeat
3.  Realiza uma amostragem apartir de C* para
    gerar novas configurações;
4.  Executa as configurações no F-Race;
5.  Adiciona as melhores configurações em C*;
6. until a condição seja satisfeita;
7. return c* ∈ C* //É a melhor configuração
8.end
```

5. METODOLOGIA

Este trabalho se propõe a estender os estudos realizados por [15] e através da aplicação do *I/F-Race* realizar a auto-parametrização do GRASP+PR aplicado ao agrupamento de dados. Parte-se da hipótese que a utilização do *I/F-Race* na auto-parametrização do GRASP+PR no problema de agrupamento de dados pode contribuir para melhoria do desempenho, diminuição da função objetivo e/ou obtenção de agrupamentos mais coesos quando comparados com a parametrização manual. A Figura 1 ilustra a sequência de etapas que envolvem a execução do *F-Race* na auto-parametrização do GRASP+PR.

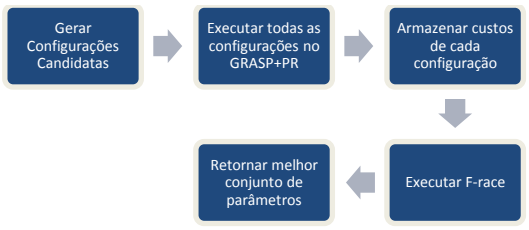

Figura 1 – Etapas do *F-Race* na auto-parametrização do GRASP+PR.

Na primeira etapa são gerados os conjuntos de parâmetros, seguida da execução destes parâmetros no GRASP+PR e do armazenamento dos custos obtidos a partir de cada conjunto de parâmetros. Por fim, após a execução do *F-Race*, o algoritmo retorna o conjunto de parâmetros que forneceu o melhor custo.

A Figura 2 ilustra a sequência de etapas para a execução do *Irace*. A primeira parte é essencialmente o *F-Race*, a diferença é que o melhor conjunto de parâmetros é adicionado a um conjunto elite e

novos conjuntos de parâmetros são gerados a partir desse conjunto. Este ciclo é realizado até que uma condição de parada seja satisfeita e o último conjunto elite é retornado.

As implementações deste trabalho, consideraram a criação de um vetor que contém todos os custos ordenados de forma decrescente, obtidos a partir da execução do GRASP+PR com cada conjunto de parâmetros. O GRASP+PR foi executado três vezes para cada conjunto de parâmetros onde se obteve a média dos valores de CRand, Função objetivo e Tempo de execução. Essa quantidade foi considerada visando reduzir o tempo de execução do GRASP+PR e se mostrou suficiente para obtenção dos melhores resultados. O critério de melhoria foi baseado na seguinte prioridade: (1) maior valor médio do CRand, (2) menor valor médio da função objetivo e (3) menor tempo de execução.

Figura 2 – Aplicação do *Irace* na auto-parametrização do GRASP+PR.

Para executar o teste de Friedman, o vetor de custos precisa ser dividido em uma matriz com no mínimo 2 linhas para que seja montada uma tabela onde será feita a etapa de criação do *ranking* conforme proposto por [8]. A partir dessa definição foi utilizada a seguinte estratégia, a quantidade de custos do vetor obtidos na primeira etapa foi dividida em uma matriz com a quantidade de 15 colunas, denominadas grupos e N/15 linhas denominadas blocos, sendo N a quantidade de conjuntos de parâmetros. Para a realização dos experimentos os métodos *I/F-race* foram executados considerando 60, 90, 150 e 210 conjuntos de parâmetros. Estes valores seguem as definições de populações iniciais nos métodos utilizados em [28] e tem como objetivo verificar se a quantidade de conjunto de parâmetros influencia na obtenção de melhores configurações.

A partir da matriz de custos é construída uma nova matriz contendo o *ranking* de todos valores contidos nos blocos. Nesta etapa é realizado o test *post hoc* de Friedman (10) com nível de confiança de α=0,05 para realizar o cálculo das diferenças observadas. Em seguida, com base na tabela da distribuição T, é verificado quais grupos possuem diferença maior que a diferença crítica, o que fornece subsídios para eliminação dos piores grupos.

Os conjuntos de parâmetros que restaram após as eliminações são reordenados para permitir que os blocos que perderam grupos possam ser completados com os conjuntos de parâmetros sobreviventes de outros blocos. Este balanceamento faz com que o número de blocos diminua gradualmente com as execuções do algoritmo. Os passos anteriores são repetidamente realizados até que não exista mais diferenças entre os grupos e ao final a primeira posição do pool de conjunto parâmetros, que possui o melhor custo, é retornada.

4.1. Métricas Analisadas

Os algoritmos foram analisados considerando o tempo de execução (em segundos.milessegundos), *Time to Target Plot* (*tttplot*) [1], função objetivo (FO) e *Corrected Rand Index* (CRand) [25]. O *tttplot* é uma medida utilizada para comparação de tempo em algoritmos estocásticos. Para construção dos *tttplots* os algoritmos foram executados 200 vezes com os parâmetros manuais e os encontrados pelos métodos *I/F-Race*. A função objetivo do GRASP+PR fornece uma referência para coesão dos agrupamentos gerados e por fim o CRand compara a similaridade entre os agrupamentos obtidos com os agrupamentos de referência fornecidos com o conjunto de dados. A validação dos experimentos se deu pela comparação dos resultados obtidos com *I/F-Race* na auto-parametrização do GRASP+PR nas variantes *forward*, *backward*, *mixed*, *greedy randomized adaptive* e *truncated*, com as mesmas variantes parametrizadas manualmente.

4.2. Ambiente de desenvolvimento e testes

O *I/F-Race* foi implementado em ambiente *Microsoft Windows* utilizando a interface de desenvolvimento *NetBeans 7.1.2* e linguagem *Java SE Ver. 7u21*. Foram utilizadas as seguintes bibliotecas: *Colt* ver. 1.0.3 [33] para geração de números aleatórios, CILIB [10], *Functional* Java [34], *Google-collect-1.0* [20] para realização dos cálculos referentes ao teste de Friedman, e Renjin [16] para executar funções da linguagem R no Java como a função "tnorm" [9] que gera uma amostragem de acordo com distribuição normal truncada. 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, executando o sistema operacional *Microsoft Windows 8* e a máquina virtual *Java SE Ver. 7u45*.

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 presente no *pool* como critérios para que a nova solução possa ser adicionada ao *pool* de soluções.	2	50% do conjunto de dados.
ISM	Quantidade máxima de iterações sem melhorias do GRASP+PR para que este encerre seu funcionamento.	1	150
BL	Máximo de iterações na fase de busca local do GRASP.	1	300
NH	Porcentagem do tamanho da lista preliminar de arestas nos conjuntos de dados que contêem mais de 15 objetos.	10	50
QTDGRUPOS	Quantidade de grupos utilizados na tarefa de agrupamento.	2	30
VAR_PR	Variante *Path-Relinking* usada (0=*Forward*, 1=*Backward*, 2=*Mixed*, 3=*Greedy Randomized*, 4=*Truncated*).	0	4

O código do GRASP+PR utilizado segue o Algoritmo 1 [15]. Os parâmetros que serão auto-parametrizados, foram escolhidos com base na sua influência no comportamento do algoritmo. Os parâmetros considerados nos experimentos estão descritos na Tabela 1.

A distância *City Block* (11) 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.

$$d_{ij} = \sum_{k=1}^{L} |a_{ik} - a_{jk}| \qquad (11)$$

A Tabela 2 contém detalhes dos conjuntos de dados utilizados. Os resultados com parâmetros manuais foram obtidos nos estudos realizados por [15].

Tabela 2 – Conjuntos de dados utilizados nos experimentos.

Conjunto de Dados	#Objetos	#Atributos	Referência
BreastB2	49	1213	[45]
DLBCLA	141	661	[45]
MultiA	103	5565	[46]
Novartis	103	1000	[46]
Glass	214	9	[47]

6. RESULTADOS E DISCUSSÃO

Os resultados obtidos com os experimentos estão descritos na Tabela 3. Os parênteses nas linhas *I/F-Race* contém a variante do PR e a quantidade de conjuntos de parâmetros que apresentaram os melhores resultados, os quais estão destacados em negrito.

É possível observar a ocorrência de maiores CRands em três dos cinco conjunto de dados analisados, tendo o *F-Race* contribuído para melhorias nos conjunto de dados BreastB2 e DLBCLA, e o *Irace* para o MultiA. A redução do desvio padrão no BreastB2, DLBCLA e Glass, mostram uma melhoria na robustez do método.

Com relação a função objetivo, cuja redução indica uma maior coesão dos agrupamentos, embora tenha-se observado melhoria apenas no conjunto de dados BreastB2, é possível constatar reduções significativas no desvio padrão em quatro dos cinco conjunto de dados BreastB2, MultiA, DLBCLA e Glass.

Melhorias no tempo de execução são observadas nos conjunto de dados MultiA, Novartis e DLBCLA. No Novartis, o *Irace* apresentou o melhor tempo sem comprometer o valor do CRand. Como valor *target* para o *tttplot*, considerou-se o maior valor da função objetivo obtido em cada conjunto de dados.

Os gráficos *tttplot* das figuras 4 e 5 ilustram melhorias de tempo no Novartis e DLBCLA respectivamente, onde é possível constatar que nos algoritmos auto-parametrizados o tempo de convergência para alcançar o valor da função objetivo é significativamente menor que os parametrizados manualmente.

A análise dos dados confirma a hipótese que a auto-parametrização do GRASP+PR no agrupamento de dados através do *I/F-Race* contribui para melhoria dos resultados quando comparados aos algoritmos parametrizados manualmente.

Tabela 3 – Resumo dos resultados obtidos nos experimentos.

ALGORITMO	CRAND		FO		TEMPO (s.ms)	
	Média	DP	Média	DP	Média	DP
Breast B2						
***F-Race* (PRb/210)**	**0.322**	**0.0372**	**68261.9**	**58.927**	4.342	4.630
***Irace* (PRb/90)**	0.311	0.0389	68403.1	154.063	1.099	1.223
GRASP-PRb	0.285	0.0478	68798.2	334.293	0.424	0.458
GRASP-PRf	0.266	0.0552	69232.6	431.619	0.379	0.408
GRASP-PRgr	0.236	0.0612	69917.8	507.777	**0.197**	**0.220**
GRASP-PRm	0.243	0.0617	69831.4	476.933	0.276	0.299
GRASP-PRt	0.266	0.0552	69232.6	431.619	0.345	0.381
MultiA						
***F-Race* (PRgr/150)**	0.869	0.0227	1.48E+09	**222774.000**	6.316	6.568
***Irace* (PRm/60)**	**0.880**	0.0415	1.48E+09	847452.000	**0.084**	**0.096**
GRASP-PRb	0.861	0.0178	1.48E+09	290116.000	1.527	1.594
GRASP-PRf	0.864	0.0168	1.48E+09	339546.000	1.386	1.477
GRASP-PRgr	0.878	**0.0157**	1.48E+09	297480.000	1.564	1.653
GRASP-PRm	0.879	0.0169	1.48E+09	322229.000	1.369	1.429
GRASP-PRt	0.864	0.0168	1.48E+09	339546.000	1.399	1.468
Novartis						
***F-Race* (PRm/210)**	0.944	0.026	1033170	344.674	**0.114**	**0.125**
***Irace* (PRf/210)**	**0.950**	**0.0000**	**1033080**	**0.000**	0.489	0.517
GRASP-PRb	**0.950**	**0.0000**	**1033080**	**0.000**	1.889	2.043
GRASP-PRf	**0.950**	**0.0000**	**1033080**	**0.000**	2.195	2.364
GRASP-PRgr	**0.950**	**0.0000**	**1033080**	**0.000**	2.095	2.282
GRASP-PRm	**0.950**	**0.0000**	**1033080**	**0.000**	2.431	2.607
GRASP-PRt	**0.950**	**0.0000**	**1033080**	**0.000**	2.231	2.400
DLBCLA						
***F-Race* (PRm/60)**	**0.834**	**0.0111**	**4.61E+08**	**5413.550**	2.537	2.630
***Irace* (PRm/210)**	0.826	0.0192	**4.61E+08**	11669.400	**0.968**	**1.069**
GRASP-PRb	0.829	0.0174	**4.61E+08**	10493.400	2.136	2.339
GRASP-PRf	0.828	0.0183	**4.61E+08**	11398.800	2.318	2.557
GRASP-PRgr	0.822	0.0212	**4.61E+08**	13845.000	1.989	2.226
GRASP-PRm	0.822	0.0220	**4.61E+08**	14841.300	1.874	2.074
GRASP-PRt	0.828	0.0183	**4.61E+08**	11398.800	2.478	2.711
Glass						
***F-Race* (PRb/60)**	**0.260**	**0.0000**	**10173.3**	**3.28E-11**	19.671	20.724
***Irace* (PRb/90)**	**0.260**	**0.0000**	**10173.3**	**3.28E-11**	17.028	17.892
GRASP-PRb	**0.260**	0.0014	**10173.3**	2.81E-02	8.829	9.463
GRASP-PRf	**0.260**	0.0010	**10173.3**	2.00E-02	11.743	12.423
GRASP-PRgr	0.259	0.0033	**10173.3**	6.53E-02	**7.781**	**8.204**
GRASP-PRm	0.259	0.0029	**10173.3**	5.89E-02	8.206	8.658
GRASP-PRt	**0.260**	0.0010	**10173.3**	2.00E-02	9.186	9.636

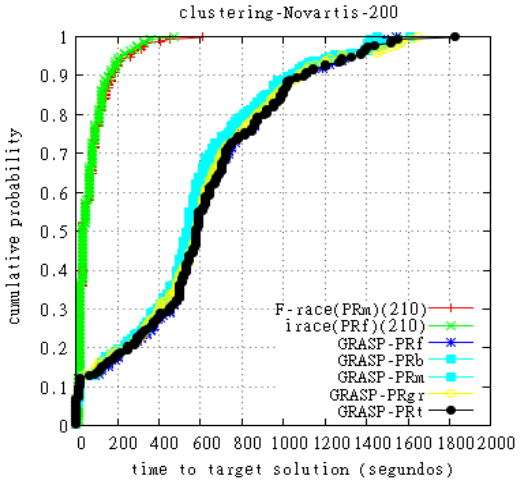

Figura 4 – *tttplot* do conjunto de dados Novartis

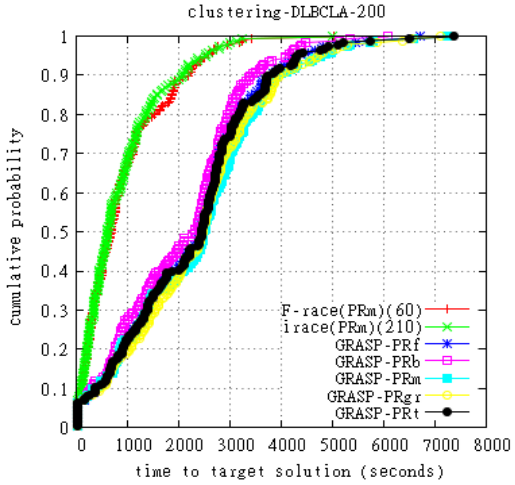

Figura 5 – *tttplot* do conjunto de dados DLBCLA

Pesquisas envolvendo a utilização de métodos evolutivos na auto-parametrização do GRASP+PR no agrupamento estão sendo concluídas e possibilitarão uma comparação com os métodos apresentados neste trabalho.

7. REFERÊNCIAS

[1] 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*, pp. 355-366.

[2] Binato, S.; de Oliveira, G. & de Araujo, J. (2002). *A greedy randomized adaptive search procedure for transmission expansion planning. IEEE Transactions on Power Systems*, 16(2) pp. 247-253.

[3] Birattari, M. & Dorigo, M. (2004). *"The problem of tuning metaheuristics as seen from a machine learning perspective." PhD thesis, Univerité Libre de Bruxelles, Brussels, Belgium.*

[4] Birattari, M.; Stützle, T; Paquete, L. & Varrentrapp, K. (2002). "*A Racing Algorithm for Configuring Metaheuristics*." GECCO. Vol. 2.

[5] Birattari, M.; Yuan, Z; Balaprakash, P. & Stützle, T. (2010). "*F-Race and iterated F-Race: An overview." Experimental methods for the analysis of optimization algorithms*, pp. 311-336, *S*pringer Berlin Heidelberg.

[6] Birattari, M.; Stützle, *T*; *Paquete, L.* & Varrentrapp, *K. (*2009) "*Tuning metaheuristics: a machine learning perspective*". Springer, Vol. 197.

[7] Brown, D. E. & Huntley, C. L. (1992). *A practical application of simulated annealing to clustering.* In Pattern Recognition, pp. 401–412.

[8] Conover, W. J. (1999). *Practical Nonparametric Statistics*, 3rd *Edition*, New York: John Wiley & Sons.

[9] Christopher, J., *Function to sample according to a truncated normal distribution. 2011.* https://github.com/cran/Irace/blob/master/R/tnorm.R. Acessado em 22/10/2014.

[10] CIRG - *Computational Intelligence Research Group* (CIRG@UP) *Department of Computer Science University of Pretoria South Africa.* Biblioteca CILIB. 2010. *http://www.cilib.net/*. Acessado em 22/10/2014.

[11] Dobslaw, F. (2010). *A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In Proceeding of the International Conference on Computer Mathematics and Natural Computing* 2010. WASET.

[12] Du , K.-L. (2010). *Clustering: A neural network approach, Neural Networks, Volume* 23, *Issue* 1, pp. 89- 107.

[13] Feo, T. A. & Resende, M. G. (1995). *Greedy randomized adaptive search procedures. Journal of global optimization*, *6*(2), pp 109-133.

[14] 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". In Experimental Algorithms*, pp. 338-349. Springer Berlin Heidelberg.

[15] 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.

[16] FSF - *Free Software Foundation*. Biblioteca Renjin, 2007. http://www.renjin.org/. Acessado em 22/10/2014.

[17] Glover, F. (1997). "*Tabu search and adaptive memory programming: advances, applications and challenges*. In *Interfaces in computer science and operations research"* pp. 1-75. Springer US.

[18] Glover, F.; Laguna, M. & Martı, R. (2000). *Fundamentals of scatter search and Path-Relinking. In Control and Cybernetics*, pp. 653–684.

[19] Glover, F. & Marti, R. (2006). "Tabu search". In *Metaheuristic procedures for training neutral networks*, pp. 53-69.

[20] Google Inc. Biblioteca Google-Collections .2007. http:// *code.google.com/p/google-collections/.* Acessado em 22/10/2014.

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

[22] Hall, L.; Ozyurt, B. & Bezdek, J. (1999). *Clustering with a genetically optmized aproach. In Evolutionary Computation*, pp 103–112.

[23] Hartigan, J. A. & Wong, M. A. (1979). "*Algorithm AS* 136: *A k-means clustering algorithm*." *Applied statistics*, pp 100-108.

[24] Hamadi, Y.; Monfroy, E. & Saubion, F. (2012) *"Autonomous Search"* XVI, 308 p., ISBN: 978-3-642-21433-2, Springer.

[25] Hubert, L. & Arabie, P. (1985). *Comparing partitions. Journal of Classification*, 2(1):193-218.

[26] Jain, A. K. & Dubes, R. C. (1988). "*Algorithms for clustering data*.". *Englewood Cliffs. Prentice-Hall*.

[27] Kaufman, L. & Rousseeuw, P. J. (1990). "Partitioning *around medoids (program pam).*" *Finding groups in data: an introduction to cluster analysis*, pp 68-125.

[28] Morán-Mirabal, L. F.; González-Velarde, J. L. & Resende, M. G. (2013). *Automatic tuning of GRASP with evolutionary path-relinking. In Hybrid Metaheuristics* (pp. 62-77). Springer Berlin Heidelberg.

[29] Lee, D.; Chen, J. & Cao, J. (2010). *"The continuous berth allocation problem: A greedy randomized adaptive search solution". Transportation Research Part E: Logistics and Transportation Review*, 46(6):1017-1029.

[30] Lopes, M. C. S. (2004). Mineração de dados textuais utilizando técnicas de clustering para o idioma português. *Doctoral dissertation*, Departamento de Engenharia Civil. Universidade Federal do Rio de Janeiro.

[31] López-Ibánez, M.; Dubois-Lacoste, J.; Stützle, T.; & Birattari, M. (2011). *The irace package: Iterated racing for automatic algorithm configuration*. Tech. Rep. TR/IRIDIA/2011-004, Université Libre de Bruxelles, Belgium.

[32] Maron, O. & Moore, A.W. (1994). Hoeffding races: *Accelerating model selection search for classification and function approximation*. In: Cowan J. D., et al. (eds), *Advances in Neural Information Processing Systems* Morgan Kaufmann , San Francisco, CA, USA, vol 6, pp. 59–66.

[33] Matsumoto, M. & Nishimura, T. (1998). *Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation*, 8:3-30.

[34] Morris, T.; Bjarnason, R.; Adams, T; Clow, B; Clarkson, R; Partridge, N. & Zaugg, T. (2010) Biblioteca functionaljava. Regents of the University of California. *Site:* http://www.functionaljava.org/. Acessado em 22/10/2014.

[35] Nascimento, M.; Toledo, F. & de Carvalho, A. (2010). *Investigation of a new grasp based clustering algorithm applied to biological data. In Computers and Operations Research*, pp 1381–1388.

[36] Ohata, A. T.; & Quintanilha, J. A. (2005). O uso de algoritmos de clustering na mensuração da expansão urbana e detecção de alterações na Região Metropolitana de São Paulo. *Anais do XII Simpósio Brasileiro de Sensoriamento Remoto, Goiânia, Brasil, 16–21 de Abril de 2005*, 647-655.

[37] Ozbay, Y.; Ceylan, R. & Karlik, B. (2006). *A fuzzy clustering neural network architecture for classification of ecg arrhythmias. In Computers in Biology and Medicine*, pp 376–388.

[38] Papoulis, A. (1991) & *Random Variable Probability. "Stochastic Processes. " McGraw-Hill*.

[39] Resende, M. G. & Ribeiro, C. C.(2005). *GRASP with path-relinking: Recent advances and applications*. In *Metaheuristics: progress as real problem solvers*, pp. 29-63. Springer US.

[40] Resende, M. G. C.; Ribeiro, C. C.; Glover, F. & Martí, R. (2010b*). Scatter search and path-relinking: Fundamentals, advances and applications*. *Handbook of Metaheuristics*, pp. 87-107.

[41] Sirdey, R.; Carlier, J. & Nace, D. (2010). *A GRASP for a resource-constrained scheduling problem. International Journal of Innovative Computing and Applications*, 2(3):143-149.

[42] Spolaôr, N.; Lee, H. D.; Ferrero, C. A.; Coy, C. S. R., Fagundes; J. J., Wu; F. C.; ... & de Coloproctologia, S. (2008). Um Estudo da Aplicação de Clustering de Séries Temporais em Dados Médicos.

[43] Zlochin, M.; Birattari, M.; Meuleau, N. & Dorigo M. (2004) "*Model-based search for combinatorial optimization: A critical survey*." *Annals of Operations Research, vol* 131. pp 373-395.

[44] Botelho, A. L. V.; Semaan, G. S. & Ochi, L. S. (2011) "Agrupamento de Sistemas Orientados a Objetos com Metaheurísticas Evolutivas" Anais do VII Simpósio Brasileiro de Sistemas de Informação. págs. 153 – 165.

[45] Monti, S.; Savage, K.; Kutok, J.; Feuerhake, F.; Kurtin, P.; Mihm, M.; Wu, B.; Pasqualucci, L.; & Others. (2005) "Molecular profiling of difuse large B-cell lymphoma identifies robust subtypes including one characterized by host inflammatory response. Blood, 105(5):1851-1861.

[46] Su, A.; Cooke, M.; Ching, K.; Hakak, Y.; Walker, J.; Wiltshire, T.; Orth, A.; Vega, R.; Sapinoso, L.; Moqrich, A. & Others (2002). Large-scale analysis of the human and mouse transcriptomes. Proceedings of the National Academy of Sciences, 99(7):4465-4470.

[47] Evett, I. W., & Spiehler, E. J. (1987). "Rule induction in forensic science". KBS in Goverment, Online Publications, páginas 107-118.