GRASP COM *PATH-RELINKING* PARA AGRUPAMENTO DE DADOS BIOLÓGICOS

RAFAEL DE MAGALHÃES DIAS FRINHANI

GRASP COM *PATH-RELINKING* PARA AGRUPAMENTO DE DADOS BIOLÓGICOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Ricardo Martins Abreu Silva
Coorientador: Geraldo Robson Mateus

Belo Horizonte
Março de 2011

Frinhani, Rafael de Magalhães Dias
F914g GRASP com *Path-Relinking* para agrupamento de dados biológicos / Rafael de Magalhães Dias Frinhani. — Belo Horizonte, 2011
xvi, 85 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais— Departamento de Ciência da Computação.
Orientador: Ricardo Martins Abreu Silva
Coorientador: Geraldo Robson Mateus

1. Agrupamento - Teses. 2. Biologia Computacional - Teses. 3. Meta-heurística - Teses. I.Orientador. II.Coorientador. III.Título.

CDU 519.6*93(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

GRASP com path relinking para agrupamento de dados biológicos

RAFAEL DE MAGALHÃES DIAS FRINHANI

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. RICARDO MARTINS DE ABREU SILVA - Orientador
Departamento de Ciência da Computação - UFLA

PROF. GERALDO ROBSON MATEUS - Co-orientador
Departamento de Ciência da Computação - UFMG

PROF. MAURÍCIO GUILHERME DE CARVALHO RESENDE
AT&T Labs Research

PROF. THIAGO FERREIRA DE NORONHA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 02 de março de 2011.

Agradecimentos

A Deus, por iluminar meus pensamentos e me ajudar nas escolhas corretas.
A meus pais, pelo exemplo de força e determinação.
A minha esposa Vanessa, pelo carinho, dedicação e por encher meu coração de orgulho e felicidade.
Ao meus orientadores Ricardo e Robson, por mostrarem caminhos nunca antes imaginados.

Resumo

O agrupamento é um método não supervisionado de classificação de dados em grupos (*clusters*). Em problemas de agrupamento, não se sabe previamente quantas e quais as classes necessárias para descrever coerentemente um conjunto de dados.

Na biologia computacional, o agrupamento mostrou-se ferramenta útil em problemas de descoberta de padrões em dados como classificação de proteínas, predição da localização de proteínas em unidades celulares e diagnóstico de câncer. Recentemente, técnicas de otimização, como as metaheurísticas, têm sido utilizadas na literatura como método alternativo ou auxiliar para aumentar a eficiência e eficácia das ferramentas clássicas de agrupamento, geralmente baseadas em métodos estatísticos e matemáticos.

O objetivo deste trabalho é propor algoritmos híbridos fundamentados nas metaheurísticas *Greedy Randomized Adaptative Search Procedure* (GRASP) e *Path-Relinking* para o problema de agrupamento de dados biológicos, com intuito de obter melhores soluções quando considerado unicamente o GRASP na sua forma padrão. Portanto, neste trabalho, considera-se a hipótese que o *Path-Relinking* como estratégia de intensificação do GRASP, melhora o desempenho e qualidade das soluções.

Considera-se a hibridização do GRASP proposto por Nascimento et al. [2010b], com quatro variações do *Path-Relinking*: *Forward*, *Backward*, *Mixed*, e *Greedy Randomized Adaptative*. A validação das soluções obtidas da-se por meio da comparação com os algoritmos clássicos k-means, k-medians, PAM bem como o GRASP proposto por Nascimento et al. [2010b].

Os experimentos foram realizados com dados reais de 10 instâncias biológicas, e os resultados mostraram que o modelo híbrido melhorou a tarefa de agrupamento. Obteve-se maior exploração do espaço de busca, maior coesão de agrupamentos, aumento da robustez e redução do tempo computacional. As variantes *Greedy Randomized Adaptative* e *Mixed* apresentaram os melhores resultados.

Palavras-chave: Agrupamento, Biologia Computacional, Metaheurísticas, GRASP, *Path-Relinking*.

Abstract

Clustering is an unsupervised method of classifying data into clusters. In clustering problems, it is not previously known how many and which classes are needed to describe coherently a set of data.

In computational biology, data clustering proved to be useful in problems of patterns discovery in data such as protein classification, prediction of protein localization in cell units and cancer diagnosis.

Recently, optimization techniques like metaheuristics, have been used frequently in the literature as an alternative or adjunct to increase the efficiency and effectiveness of the classic tools of clustering, usually based on statistical and mathematical methods.

In this study we proposed hybrid algorithms based on metaheuristics GRASP and Path-Relinking for the clustering problem of biological data, aiming to obtain better solutions when compared to using only the GRASP in its standard form. Therefore, in this work, we considered the hypothesis that using the Path-Relinking as a strategy for intensifying GRASP, improves performance and quality of solutions.

We consider hybridization of the GRASP proposed by Nascimento et al. [2010b], with four variants of Path-Relinking: forward, backward, mixed, and greedy randomized adaptative. The validation of solutions is given by comparing with the classic algorithms k-means, k-medians, PAM (Partitioning Around Medoids) as well as GRASP proposed by Nascimento et al. (2010).

The experiments were performed with real data from 10 biological instances, and results showed that the hybrid model improved the clustering task. We obtained further exploration of the search space, more cohesive clusters, increase of robustness and reduction of computational time. Greedy Randomized Adaptive and Mixed showed the best results.

Keywords: Clustering, Computational Biology, Metaheuristics, GRASP, Path-Relinking.

Lista de Figuras

Sumário

Capítulo 1

Introdução

O crescimento da quantidade de dados em formato digital e das tecnologias de armazenamento contribuíram para a formação de grandes bases de dados. Segundo Bucene et al. [2002], cada vez mais, o volume de informações excede a capacidade de análise pelos métodos tradicionais, incapazes de analisar os dados sob o enfoque do conhecimento. Como resultado desse aumento efetivo, o processamento dessas informações tornou-se complexo e difícil, culminando em dados armazenados, mas sem que sejam utilizados de uma forma realmente eficiente [Halmenschlager, 2000].

Knowledge Discovery in Database (KDD) [Frawley et al., 1992; Piatetsky-Shapiro & Frawley, 1991], *Data Mining* [Witten & Frank, 2005; Fayyad et al., 1996; Han & Kamber, 2006], *Pattern Recognition* [Webb, 1999; Ripley, 2008], *Pattern Discovery* [Wang et al., 1999; Rigoutsos et al., 2000] e *Machine Learning* [Bishop, 2006; Michalski et al., 1994] são segmentos da computação que tratam o desenvolvimento e utilização de métodos para identificar em dados, padrões válidos, previamente desconhecidos, potencialmente úteis e compreensíveis, que visam melhorar o entendimento de um problema ou processo de tomada de decisão.

Diferentes técnicas analíticas podem ser empregadas na descoberta de padrões, variando entre abordagens descritivas e gráficas mais básicas, técnicas multivariadas (ex. análise de agrupamentos ou regressão múltipla ou logística) e modelos de aprendizado (como redes neurais e algoritmos genéticos) [Hair Jr et al., 2006]. Essas técnicas podem ser organizadas como Análise de Regras de Associação Zhang & Zhang [2002], Classificação e Predição [Phyu, 2009], Análise de Padrões Sequênciais [Zhao & Bhowmick, 2003], Análise de *outliers* [Hodge & Austin, 2004] e Análise de Agrupamentos [Berkhin, 2006].

A análise de agrupamentos *(clustering)* consiste em identificar classes de itens em uma base de dados de acordo com alguma medida de similaridade. Cada grupo

consiste de objetos que são similares entre si e diferentes dos objetos dos outros grupos. A diferença entre os objetos e entre grupos é dada por um valor, calculado por medidas de distância ou similaridade. Os algoritmos clássicos de análise de agrupamentos são baseados em técnicas matemáticas, estatísticas e de análise numérica e, diferente de outras tarefas como a classificação e a predição, trabalham sobre dados nos quais as classes não estão previamente definidas. Apesar do grande número de algoritmos para análises de agrupamento, cada algoritmo tem suas vantagens e desvantagens, sendo mais adequado para tipos específicos de dados.

A biologia computacional trata da aplicação de métodos computacionais e analíticos para problemas biológicos, e é uma área em rápida evolução científica. Bases de dados e literatura biológicas estão crescendo de forma acelerada [Gibas & Jambeck, 2001]. Segundo Kitano [2002], o entendimento de sistemas biológicos complexos requer a integração de pesquisa experimental e computacional e visa alcançar inovações práticas na medicina. Diversos experimentos utilizando tarefas de agrupamento de dados para bases biológicas têm sido publicados [Hand & Heard, 2005; An & Chen, 2009; Alon et al., 1999; Monti et al., 2003; Ma et al., 2006].

A grande quantidade de dados e a determinação da quantidade de grupos desejada, tornam a tarefa de agrupamento um complicado problema combinatório. Algoritmos que realizam esta tarefa, em geral, apresentam alta complexidade computacional e/ou baixa precisão. Neste contexto, técnicas de otimização como programação linear, programação inteira e metaheurísticas, têm sido utilizadas complementarmente aos algoritmos clássicos, de forma a melhorar o desempenho e/ou precisão das tarefas de agrupamento.

Recentemente, o interesse no uso de metaheurísticas para agrupamento de dados aumentou significativamente [Ma et al., 2006; Bandyopadhyay & Maulik, 2002; Pacheco, 2005; Al-Sultan, 1995]. Nascimento et al. [2010b] foram pioneiros na utilização da metaheurística GRASP - *Greedy Randomized Adaptative Search Procedure* (Procedimento adaptado de busca gulosa aleatória) para o agrupamento de dados biológicos, obtendo sucesso em comparação a algoritmos clássicos como k-means, k-medians e PAM (*Partitioning Around Medois*).

O GRASP [Feo & Resende, 1995] é um algoritmo de busca, multi-inícios, que constrói soluções iniciais por meio de um processo semi-guloso, e aplica uma busca local em volta de cada solução previamente construída. GRASP foi aplicado com sucesso em vários problemas combinatórios e.g. [Festa & Resende, 2009, 2002; Areibi & Vannelli, 1997; Laguna & Marti, 1999; Cano et al., 2002; Pardalos & Resende, 1994].

As iterações do GRASP são independentes, isto é, as soluções encontradas em iterações anteriores não influenciam o algoritmo na solução atual. Portanto, o funcio-

namento do GRASP tradicional não se baseia em aprendizagem sobre a sua execução. O uso de soluções previamente encontradas para influenciar o procedimento na iteração corrente pode ser obtido através de mecanismos de memória. Tais mecanismos constroem um conjunto de soluções elite, evitando a exploração de soluções muito ruins e/ou redundantes.

Uma das formas de incorporar memória no GRASP é com a estratégia *Path-Relinking* [Glover, 1996, 2000; Glover & Marti, 2006; Glover et al., 2000]. Com *Path-Relinking*, após uma solução ter sido gerada pelo GRASP, ela é combinada com uma solução aleatoriamente selecionada em um *pool* de soluções elite. O objetivo do GRASP com *Path-Relinking* [Laguna & Marti, 1999] é encontrar soluções intermediárias de melhor qualidade, que estejam entre duas boas soluções. GRASP com *Path-Relinking* foi aplicado com sucesso em [Festa et al., 2007; Mateus et al., 2010; Bastos et al., 2005].

O objetivo desse trabalho foi propor algoritmos híbridos fundamentados nas heurísticas GRASP e *Path-Relinking* para o problema de agrupamento de dados biológicos, com intuito de obter melhores soluções quando considerado unicamente o GRASP na sua forma padrão. Portanto, neste trabalho, considerou-se a hipótese que o *Path-Relinking* como estratégia de intensificação do GRASP melhora o desempenho e qualidade das soluções. Para testar essa hipótese, foi realizada a hibridização do GRASP proposto por Nascimento et al. [2010b], com quatro variações do *Path-Relinking*: *forward, backward, mixed, e greedy randomized adaptative.* A validação das soluções obtidas deu-se por meio da comparação com os algoritmos clássicos k-means, k-medians e PAM e com o algoritmo proposto por Nascimento et al. [2010b], que utiliza o GRASP. Para realização dos experimentos, foram utilizadas 10 instâncias biológicas e os algoritmos foram analisados quanto a sua robustez, eficácia e coesão dos agrupamentos em comparação com as classificações reais fornecidas com as instâncias supracitadas.

Esta dissertação está organizada como se segue. No capítulo 2 é apresentado o estado da arte e contém a definição do problema de agrupamento de dados, classificação taxonômica das técnicas e algoritmos clássicos de agrupamento, descreve as etapas necessárias para análise de agrupamentos bem como as medidas de similaridade utilizadas neste trabalho. Este capítulo também contém os conceitos de metaheurísticas e detalha o GRASP, suas respectivas fases e aplicações em diversos problemas de agrupamento, e o *Path-Relinking*. O capítulo 3 descreve a metodologia utilizada no desenvolvimento deste trabalho. O capítulo 4 apresenta os resultados experimentais e a discussão. Por fim, no capítulo 5 são apresentadas as conclusões e trabalhos futuros.

Capítulo 2

Estado da Arte

2.1 Agrupamento de Dados

A classificação é uma função de aprendizado que mapeia dados de entrada em um número finito de categorias. Tal função utiliza um conjunto pré-definido de classes, que servem de exemplo para classificação dos objetos pertencentes a um conjunto de dados. O objetivo de um algoritmo de classificação é encontrar algum relacionamento entre os atributos de um objeto e uma classe, de modo que o processo de classificação possa usar esse relacionamento para predizer a classe de um exemplo novo e desconhecido. A classificação é chamada de método preditivo [Rezende, 2003].

O agrupamento é um método não supervisionado de classificação de padrões em grupos, ou seja, nesse problema não se sabe quais são as classes, nem mesmo quantas são. Basicamente, a tarefa de agrupamento consiste em dividir os dados em grupos de objetos similares. Sendo assim, o agrupamento de dados é uma tarefa descritiva, que procura identificar um conjunto finito de grupos, a partir dos dados. Isso é feito, geralmente, de maneira que objetos com valores de atributos similares sejam reunidos em um mesmo grupo. Medidas de distância são geralmente utilizadas como base para agrupar os objetos. O que se espera é que os grupos obtidos sejam os mais homogêneos (coesão interna) e bem separados possíveis (isolamento externo) [Anderberg, 1973; Braga, 2005; Berkhin, 2006; Jain et al., 1999].

Formalmente, pode-se definir o problema de agrupamento da seguinte forma [Berkhin, 2006; Grabmeier & Rudolph, 2002; Rodrigues, 2009]: Seja $X = \{x_1, x_2, ..., x_n\}$, contendo um conjunto de n objetos para análise. Cada objeto x_i é um vetor com d dimensões, cujos componentes são escalares chamados de atributos. O objetivo do agrupamento de dados é encontrar grupos $C = \{C_1, C_2, ..., C_K\}$, onde K é o número de grupos. Diz-se que C é o conjunto de grupo obtido para X se três características

podem ser observadas nesse conjunto:

1. $C_1 \cup C_2 \ ... \ \cup C_k = X$;
2. $C_i \neq \emptyset; \forall \ i$;
3. $C_i \cap C_j = \emptyset; \forall \ i \neq j$;

O problema de agrupamento, em teoria, pode ser resolvido enumerando todas as possíveis formas de agrupar os objetos em grupos e avaliar cada solução de forma exaustiva ou por métodos exatos de forma implícita [Bertsimas & Shioda, 2007]. A abordagem exaustiva é computacionalmente inviável para problemas grandes, como resultado, muitas técnicas de agrupamento são baseadas na tentativa de maximizar ou minimizar uma função objetivo que representa as similaridades entre os objetos através de métodos heurísticos.

Ward Jr. [1963] explica que, em geral, uma função objetivo pode ser qualquer relação funcional que o investigador seleciona para refletir a necessidade relativa de agrupamento dos objetos. Para Chaves [2009], os métodos de otimização da função objetivo procuram acrescentar um pouco de inteligência ao método exato, reduzindo o número de soluções para serem analisadas no espaço de busca e possibilitando a resolução de problemas de dimensões mais elevadas. Detalhes sobre métodos de agrupamento incluindo o estudo de complexidades de tempo, vantagens e desvantagens podem ser observados em [Jain et al., 1999].

2.1.1 Métodos de Agrupamento

Os métodos de agrupamento podem ser divididos em Hierárquicos e Particionais (Figura 2.1). Métodos hierárquicos constroem uma árvore de agrupamentos, conhecida como dendograma. A maioria dos algoritmos de agrupamento hierárquicos são variações dos algoritmos *single-link* [Sneath, 2005] e *complete-link* [King, 1967].

Algoritmos de agrupamento particionais procuram encontrar a melhor partição dos n objetos de entrada em k grupos de saída, onde k é um parâmetro de entrada do algoritmo. Para grandes bases de dados, os algoritmos particionais apresentam melhor desempenho, quando comparado aos hierárquicos, pois, nesse caso, a construção do dendograma eleva, em demasia, o custo computacional da tarefa de agrupamento.

Em resumo, os algoritmos particionais constroem uma partição dos dados. Os algoritmos hierárquicos constroem uma hierarquia de partições [Jain et al., 1999].

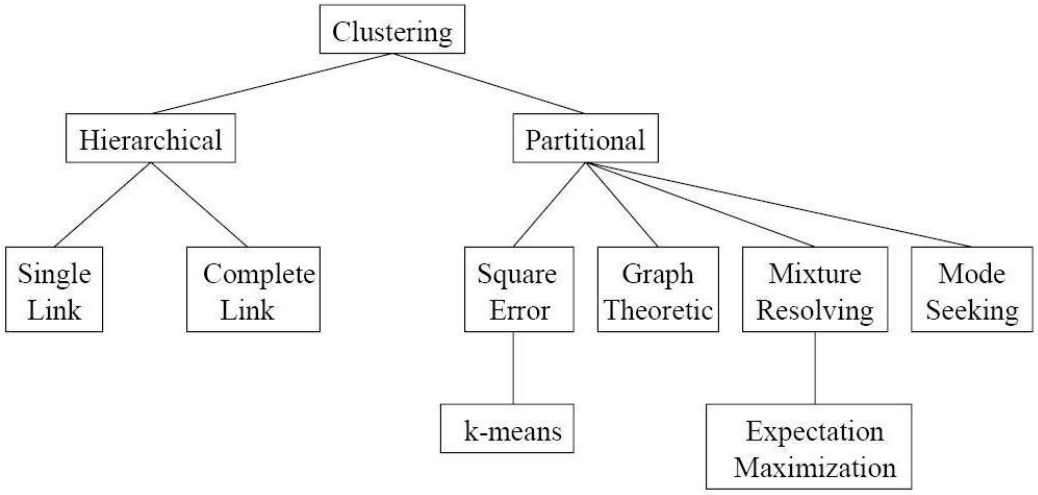

Figura 2.1. Taxonomia de técnicas de agrupamento de dados.

2.1.1.1 Métodos Particionais

As técnicas particionais normalmente produzem grupos por meio da otimização de funções objetivo. Por exemplo, o algoritmo mais popular do método de agrupamento particional, o k-means [MacQueen, 1967], tenta minimizar a soma do erro quadrático entre os objetos e o centro do grupo. Já o k-medians, uma variação do k-means, tenta minimizar a soma das distâncias dos objetos à sua mediana mais próxima. Essa também é a intenção do método PAM, baseado no k-medians. Os detalhes desses três algoritmos são apresentados abaixo.

a) k-means

A idéia central do k-means é encontrar k diferentes centróides entre seus objetos e agrupar os demais objetos com base no quadrado da distância entre o objeto e o centróide mais próximo a ele, onde k é o número de grupos, passado como parâmetro pelo usuário.

Listado por Wu & Kumar [2009] entre os dez mais influentes algoritmos da mineração de dados, o k-means deve sua grande popularidade por sua eficiência em realizar agrupamentos em bases de dados numerosas, facilidade de implementação e complexidade $O(k)$, onde k é o número de grupos. Outra característica do k-means que contribui para sua popularidade é o fato dele ser usado para inicializar algoritmos de agrupamentos mais complexos e caros computacionalmente.

O Algoritmo 1 apresenta o pseudocódigo para k-means. w_{ij} é o j-ésimo objeto do i-ésimo grupo, $\overline{w_i}$ representa o centróide do grupo i, k_i é a quantidade de objetos do grupo i [Pena et al., 1999].

Algoritmo 1: k-means()

1 *Faça aleatoriamente uma partição inicial nos dados em k grupos* $\{C_1, ..., C_k\}$;
2 *Calcule os centróides* $\overline{w_i} = \frac{1}{k_i} \sum_{j=1}^{k_i} w_{ij}$; *i=1, ..., k;*
3 **repeat**
4 | *Atribua os objetos* w_{ij} *ao grupo representado pelo centróide mais próximo;*
5 | *Recalcule os centróides dos grupos, a partir dos objetos realocados;*
6 **until** *não ocorra alteração significativa nos centróides;*

b) *k*-medians

O algoritmo k-medians é uma evolução do k-means, onde os centróides são substituídos por medóides, localizados próximos ao centro do grupo (agora uma mediana) ao invés de um centro médio (média). O objetivo é encontrar um conjunto de grupos de tal forma que cada grupo tenha um ponto central mais representativo em relação a alguma medida, como por exemplo, a distância. Este método é mais robusto para *outliers*[1] por usar medianas e por minimizar uma soma de distâncias (dissimilaridades) ao invés do desvio quadrático médio. Por outro lado, utiliza um tempo maior de processamento [Rodrigues, 2009; Kaufman & Rousseeuw, 1987; Spaeth, 1980].

O pseudocódigo do k-medians pode ser visto no Algoritmo 2:

Algoritmo 2: k-medians()

1 **repeat**
2 | Selecionar, dentre os n objetos disponíveis de X, os k objetos que definem um conjunto M de medianas;
3 | Associar os (n-k) objetos restantes à sua mediana mais próxima;
4 | Substituir as k medianas de forma a minimizar a soma das distâncias dos (n-k) objetos à sua mediana mais próxima.
5 **until** *não ocorra alteração significativa nos medóids*;

c) *Partitioning Around Medoids* (PAM)

Outro algoritmo clássico é o particionamento em torno de medianas (PAM, do inglês *Partitioning Around Medoids*) [Kaufman & Rousseeuw, 1987], baseado no k-medians. A seleção de k medoides é executada em duas fases. Na primeira fase, um agrupamento inicial é obtido pela sucessiva seleção de objetos até que k objetos, que melhor representem os centros de k grupos, tenham sido encontrados. O primeiro objeto é aquele para o qual a soma das dissimilaridades em relação a todos os outros

[1] É um elemento que apresenta características diferentes do restante dos elementos de um conjunto.

objetos é a menor possível. Subsequentemente, a cada passo, PAM seleciona o objeto que diminui a função objetivo (soma de dissimilaridades) tanto quanto possível.

Na segunda fase, todos os objetos são analisados e é escolhido como novo centro de cada grupo aquele objeto que minimiza a função objetivo tanto quanto possível. Essa função é soma de todas as dissimilaridades entre o objeto O_i e os demais objetos O_j do grupo a que ele pertence.

O algoritmo PAM funciona efetivamente com pequenas bases de dados [Rodrigues, 2009; Kaufman & Rousseeuw, 2005].

2.1.2 Análise de Agrupamento

Romesburg [2004] apresenta os passos para a análise de agrupamento, que podem ser resumidos como:

- Obter a matriz de dados;
- Computar a matriz de similaridade;
- Executar o método de agrupamento;
- Computar o coeficiente de correlação;

A matriz de dados é formada por objetos e seus atributos. A matriz de dissimilaridade (também chamada matriz de distâncias) contém o grau de similaridade entre cada par de objetos. Sendo assim, para cada par de objetos, o coeficiente de similaridade é calculado com o uso de uma medida de similaridade, a partir dos valores de seus atributos armazenados na matriz de dados. Na matriz de similaridade MS, linhas e colunas representam objetos e o valor de $MS[ij]$ é a dissimilaridade entre os objetos i e j. Existem diversas formas para se calcular o quão similar é um par de objetos. Na Seção 2.1.2.1 são apresentadas as métricas de similaridade utilizadas neste trabalho.

Ao executar o método de agrupamento, cada objeto da matriz de dados é associado a um único grupo. Posteriormente, na fase de computar o coeficiente de correlação, usa-se a função objetivo para avaliar o agrupamento gerado.

2.1.2.1 Medidas de Similaridade

Como já mencionado, diversas métricas podem ser usadas para calcular a similaridade/dissimilaridade entre os objetos [Lee, 1999]. Uma prática bastante comum é utilizar a métrica de distância *Minkowski*, que é uma generalização da distância normal entre os pontos no espaço euclidiano, dado pela Fórmula 2.1:

$$\mathbf{d}_{ij} = \left(\sum_{k=1}^{L} |a_{ik} - a_{jk}|^r \right)^{\frac{1}{r}} \tag{2.1}$$

Onde,
d_{ij} é a similaridade entre os objetos i e j;
L é a dimensão dos dados;
a_{ik} e a_{jk} são, respectivamente, o k-ésimo atributo dos i-ésimo e j-ésimo objetos a_i e a_j;
r é um fator que determina a métrica do espaço;

As distâncias euclidiana e *City Block* (também chamada *Manhattan*) são casos especiais da formulação acima mensionada. Quando r = 1, tem-se a distância *City Block* (equação 2.2). Quando r = 2, tem-se a distância euclidiana (equação 2.3), a mais comum medida de distância entre dois pontos.

$$\mathbf{d}_{ij} = \sum_{k=1}^{L} |a_{ik} - a_{jk}| \tag{2.2}$$

$$\mathbf{d}_{ij} = \sqrt{\sum_{k=1}^{L} |a_{ik} - a_{jk}|^2} \tag{2.3}$$

Outra métrica comumente utilizada é o uso dos coeficientes de correlação como medida de similaridade. Os coeficientes de correlação medem a força do relacionamento entre duas variáveis. O Cosine é um coeficiente de correlação que calcula a distância vetorial euclidiana entre dois objetos, utilizando valores contidos no intervalo [-1, 1].

A similaridade entre dois objetos utilizando a função Cosine é dada pela Fórmula 2.4 [Anderberg, 1973].

$$\mathbf{D}_{ij} = \frac{\sum_{k=1}^{L} a_{ik} a_{jk}}{\sqrt{\sum_{k=1}^{L} a_{ik}^2 \sum_{k=1}^{L} a_{jk}^2}} \tag{2.4}$$

Onde,
D_{ij} é a similaridade entre os objetos i e j;
L é a dimensão dos dados;
a_{ik} e a_{jk} são, respectivamente, o k-ésimo atributo dos i-ésimo e j-ésimo objetos a_i e a_j;

Para encontrar a dissimilaridade entre os objetos i e j deve-se considerar $d_{ij} = 1 - |D_{ij}|$. Quanto maior o valor de D_{ij}, menor é o ângulo entre os objetos e maior a similaridade entre eles. D_{ij}=1 significa que o ângulo entre os vetores que representam

o objeto é de 0º, enquanto D_{ij}=-1 significa que os vetores formam um ângulo de 90º. Cosine é uma métrica bastante utilizada na recuperação de informação textual.

Outro método usualmente conhecido para medir a correlação entre duas variáveis é o coeficiente de correlação linear de Pearson [Lira & Neto, 2008], muito utilizado em análise de agrupamento. Segundo Anderberg [1973], esse coeficiente mostra a quantidade de distorção causada pelo método de agrupamento na representação da similaridade entre pares de objetos. O coeficiente de Pearson é calculado utilizando Fórmula 2.5 descrita a seguir:

$$D_{ij} = \frac{L\left(\sum_{k=1}^{L} a_{ik}a_{jk}\right) - \left(\sum_{k=1}^{L} a_{ik} \sum_{k=1}^{L} a_{jk}\right)}{\sqrt{L\left(\sum_{k=1}^{L} a_{ik}^2\right) - \left(\sum_{k=1}^{L} a_{ik}\right)^2}\sqrt{L\left(\sum_{k=1}^{L} a_{jk}^2\right) - \left(\sum_{k=1}^{L} a_{jk}\right)^2}} \tag{2.5}$$

Onde,
D_{ij} é a similaridade entre os objetos i e j;
L é a dimensão dos dados;
a_{ik} e a_{jk} são, respectivamente, o k-ésimo atributo dos i-ésimo e j-ésimo objetos a_i e a_j;

A correlação de Pearson mede a relação entre dois objetos e produz um valor no intervalo [-1, 1]. Nessa medida, o valor 1 significa uma associação perfeita entre dois objetos, enquanto o valor -1 significa que existe uma relação linear negativa perfeita entre dois objetos. Assim como na correlação Cosine, calcula-se a dissimilaridade entre os objetos i e j por $d_{ij} = 1 - |D_{ij}|$.

2.2 Metaheurísticas

A palavra heurística tem sua origem na Grécia antiga e deriva da palavra heuriskein, que significa "a arte de descobrir novas estratégias para resolver problemas". O termo "metaheurística" foi introduzido por Glover [1986]. O sufixo meta, também é uma palavra grega, que significa "metodologia de nível superior". As metaheurísticas lidam com problemas de otimização combinatória, onde heurísticas clássicas não são eficientes e eficazes[2][Talbi, 2009; Osman & Kelly, 1996].

As metaheurísticas ganharam muita popularidade nas últimas duas décadas e representam uma família de técnicas de otimização que obtém boas soluções. Nesses

[2]Um algoritmo é eficaz quando resolve o problema para o qual foi proposto. Um algoritmo é eficiente quando alcança seus objetivos utilizando a menor quantidade possível de recursos computacionais como processamento, armazenameto, etc.

métodos a garantia de encontrar soluções ótimas é sacrificada em prol da obtenção de boas soluções em menor tempo [Blum & Roli, 2003; Talbi, 2009].

Na definição de Osman & Kelly [1996], metaheurística pode ser vista como um processo iterativo que guia uma heurística subordinada, combinando inteligentemente diferentes conceitos de exploração e aproveitamento dos espaços de busca, a fim de encontrar eficientemente soluções próximas da ideal.

Para Glover & Kochenberger [2003], metaheurísticas são métodos que orquestram uma interação entre melhoria de procedimentos de busca local e estratégias de alto nível para criar processos capazes de escapar dos ótimos locais e desempenhar uma busca mais robusta no espaço de soluções.

Segundo Blum & Roli [2003], as propriedades fundamentais que caracterizam metaheurísticas são:

- Metaheurísticas são estratégias que direcionam o processo de busca;
- O objetivo é explorar de forma eficiente o espaço de busca para encontrar ou se aproximar das melhores soluções;
- As técnicas que constituem os algoritmos de metaheurística variam de simples procedimentos de busca local até complexos processos de aprendizagem;
- Os algoritmos de metaheurística buscam boas soluções e, geralmente, são não-determinísticos[3];
- Os algoritmos de metaheurística podem incorporar mecanismos que evitem que os mesmos fiquem presos em áreas restritas do espaço de busca;
- Metaheurísticas podem fazer uso de conhecimentos específicos, na forma de heurísticas que são controladas pela estratégia de nível superior.

Na prática, as metaheurísticas fornecem modelos genéricos que permitem criar algoritmos híbridos através da combinação de diferentes conceitos derivados das heurísticas clássicas como a inteligência artificial, evolução biológica, entre outros.

Algumas metaheurísticas amplamente difundidas são: *Simulated Annealing* [Van Laarhoven & Aarts, 1987], Algoritmos Genéticos [Davis & Mitchell, 1991], GRASP

[3]Algoritmos determinísticos resolvem o problema com uma decisão exata e sempre retornam uma mesma saída para uma dada entrada. Algoritmos não-determinísticos resolvem o problema ao deduzir os melhores passos através de estimativas sob forma de heurísticas. Para uma dada entrada, cada execução do algoritmo pode retornar uma saída diferente.

(*Greedy Randomized Adaptive Search Procedure*) [Feo & Resende, 1995], Redes Neurais [Hopfield, 1982], Busca Tabu [Glover & Taillard, 1993] e seus híbridos[4] [Osman & Kelly, 1996].

Talbi [2009] explica que na concepção de metaheurística, dois critérios contraditórios devem ser levados em consideração: a exploração do espaço de busca (diversificação) e a exploração das melhores soluções encontradas (intensificação). Na diversificação, algumas regiões promissoras do espaço de busca são definidas encontrando-se boas soluções. Essas regiões promissoras são exploradas na fase de intensificação na esperança de encontrar soluções melhores do que as anteriormente encontradas. Na diversificação, regiões não exploradas são visitadas para assegurar que todas as regiões do espaço de busca sejam uniformemente exploradas e que a busca não se limite a um número reduzido de regiões.

Assim, segundo Ibaraki et al. [2005], os algoritmos de metaheurísticas podem ser simplesmente vistos como uma repetição de dois passos simples: (1) geração de soluções e (2) aperfeiçoamento das soluções pela busca local. No entanto, diversas generalizações e sofisticações desses dois passos têm sido propostas e testadas.

No contexto desse trabalho, duas metaheurísticas são de especial importância e serão aprofundadas a seguir: o GRASP e o *Path-Relinking*.

2.2.1 GRASP

O GRASP (*Greedy Randomized Adaptative Search Procedure*) foi inicialmente proposto por Feo & Resende [1989] e, desde então, tem sido usado com sucesso na resolução de problemas de otimização combinatória nas mais diversas áreas [Festa & Resende, 2009].

O GRASP é uma heurística sequencial iterativa. Cada iteração consiste em dois estágios: fase construtiva e fase de busca local. Depois de cada iteração, um "ótimo local" é encontrado e a melhor solução de todas as iterações é retornada como a solução final. GRASP combina as abordagens gulosa e aleatória. A fase construtiva constrói uma solução viável através do uso de uma função de aleatoriedade gulosa. A parte "gulosa" da função visa gerar uma solução factível de melhor custo (baixo ou alto custo, dependendo da aplicação). O componente aleatório é incluído para explorar regiões diversas do espaço de soluções e é uma das chaves da efetividade do GRASP. Sendo assim, resumidamente, o GRASP é um procedimentos de busca guloso, aleatório e adaptativo que repetidamente aplica a busca local a partir de soluções construídas com um algoritmo guloso aleatório. O melhor ótimo local dentre todas as buscas locais

[4] A combinação de uma meta-heurística com outras técnicas de otimização, pode proporcionar um comportamento mais eficiente e uma maior flexibilidade para tratar problemas em larga escala.

é retornado como solução da metaheurística [Butenko et al., 2004; Zapfel et al., 2010; Ibaraki et al., 2005; Resende & Ribeiro, 2005].

O pseudocódigo genérico do GRASP pode ser visto no Algoritmo 3. Na linha 1, inicializa-se a variável que armazenará a solução final do GRASP. Como é um método de pesquisa construtivo, começa com uma solução vazia e acrescenta elementos à uma solução parcial, até que seja construída a solução final.

Nas linhas 2 a 6, as fases construtiva e de busca local são executadas iterativamente até que um critério de parada seja atingido. Segundo Ribeiro [2005] este critério, usualmente, consiste em um número pré-definido de iterações, um limite máximo de tempo da CPU ou um número máximo de iterações sem melhoria. Na linha 3, a solução inicial é fornecida pela fase construtiva.

Na linha 4, uma busca local é feita sobre a solução obtida na linha 3. Caso a solução encontrada pela busca local seja melhor que a melhor solução encontrada pelo GRASP até então, esta passa a ser a melhor solução encontrada. Satisfeito o critério de parada, o GRASP retorna a melhor solução encontrada em todas as iterações realizadas [Feo & Resende, 1995].

Algoritmo 3: GRASP()

```
1 MelhorSolução ← φ;
2 repeat
3 |   SoluçãoInicial ← AlgoritmoConstrução;
4 |   SoluçãoMelhorada ← AlgoritmoBuscaLocal(SoluçãoInicial);
5 |   MelhorSolução ← melhor(MelhorSolução, SoluçãoMelhorada);
6 until critério de parada satisfeito;
7 return MelhorSolução
```

Para Ibaraki et al. [2005], o GRASP pode ser pensado como um método que repetidamente aplica uma busca local a partir de diferentes soluções iniciais em X. A cada passo da busca local, a vizinhança $N(x)$ da solução corrente x é pesquisada em busca de uma solução $y \in N(x)$ de forma que $f(y) < f(x)$, onde X é o conjunto de soluções possíveis e $f(\cdot)$ é uma função objetivo a ser minimizada ou maximizada. Se uma solução melhor é encontrada, ela é adotada como solução corrente e a busca local é realizada. Se nenhuma solução melhor é encontrada, o procedimento pára tendo x como solução (ótimo local).

Butenko et al. [2004] afirmam que essa metaheurística produz soluções de boa qualidade com tempo computacional relativamente pequeno para muitos problemas de otimização combinatória. Para Glover & Kochenberger [2003], uma característica especialmente atraente do GRASP é a facilidade com que pode ser implementado. Poucos parâmetros precisam ser definidos e ajustados. Portanto, o desenvolvimento pode ser

focado na eficiência das estruturas de dados para assegurar rápidas implementações. GRASP tem dois parâmetros principais: um relacionado ao critério de parada e o outro à qualidade dos elementos na lista de candidatos restritos (RCL). Segundo Resende & Ribeiro [2003], o primeiro controla o número de iterações dos métodos construtivo e de busca local a serem aplicados. O segundo controla a mistura dos enfoques aleatório e guloso do método construtivo.

Em diversos estudos o GRASP foi aplicado com sucesso na resolução de problemas de otimização combinatória [Tuyttens & Vandaele, 2010; Dharan & Nair, 2009; Lee et al., 2010; Sirdey et al., 2010; Binato et al., 2002].

A seguir, as fases construtiva e de busca local são apresentadas em maiores detalhes.

2.2.1.1 Fase Construtiva

A fase construtiva (Algoritmo 4) constrói uma solução factível através do uso de uma função aleatória gulosa. Zapfel et al. [2010] afirmam que métodos gulosos não executam a busca, mas constroem uma única solução de forma iterativa, avaliando todos os elementos da solução remanescente e, de acordo com seu desempenho, os acrescenta à solução parcial. Elementos são adicionados enquanto a solução parcial é melhorada. Se não há mais melhorias, a construção pára e a solução final é retornada.

Algoritmo 4: *ConstruirSoluçãoGulosaAleatória(Solução)*

```
1 Solução ← ∅ ;
2 repeat
3 |   ConstruirRCL(RCL);
4 |   s ← SelecionaElementoAleatoriamente(RCL);
5 |   Solução = Solução ∪ {s};
6 |   AdaptarFunçãoGulosa(s);
7 until Construção da solução não tiver terminado;
```

A construção de uma solução factível começa com os elementos de melhor custo reunidos em uma lista de candidatos restrita (RCL). A partir dessa lista, um elemento é escolhido de forma aleatória e adicionado à solução parcial corrente. Desta forma boas soluções são encontradas, mas ao mesmo tempo, diferentes soluções podem ser construídas. Assim, a abordagem gulosa é usada para construir a lista de candidatos restritos e a abordagem randômica é usada para selecionar um elemento da lista ou mesmo para construir a lista [Butenko et al., 2004; Ibaraki et al., 2005; Nascimento, 2010].

Os elementos candidatos $e \in C$ são ordenados de acordo com os valores de suas funções gulosas $g(e)$. Segundo Resende & Ribeiro [2005], a RCL consiste de elementos do conjunto $\{e \in C : g_* \leq g(e) \leq g^* + \alpha(g^* - g_*)\}$, onde $g_* = min\{g(e) : e \in C\}$; $g^* = max\{g(e) : e \in C\}$ e α é um parâmetro que satisfaz $0 \leq \alpha \leq 1$.

O α é um parâmetro de crucial importância para o desempenho do GRASP. Uma seleção criteriosa de seu valor proporciona um equilíbrio entre a diversificação e a qualidade da solução. Este parâmetro determina quais elementos serão colocados na RCL em cada iteração da fase construtiva. Num problema de maximização, o $\alpha = 1$ corresponde a um algoritmo puramente guloso, enquanto $\alpha = 0$ equivale a uma construção puramente aleatória.

Visto que o melhor valor de α é difícil de determinar, ele é normalmente atribuído como um valor aleatório para cada iteração do GRASP.

Prais & Ribeiro [2000b] investigaram o comportamento do procedimento GRASP em função de estratégias para variação do parâmetro α e concluíram que a estratégia aleatória garante bons resultados. O uso de diferentes valores para este parâmetro proporciona um aumento de variância nas soluções encontradas. Prais & Ribeiro [2000a] desenvolveram o chamado GRASP Reativo, em que o parâmetro α é auto-ajustado de acordo com a qualidade das soluções obtidas anteriormente. Segundo esses autores, essa abordagem é robusta, atingindo resultados melhores que o GRASP tradicional, e seu ajuste é simples, visto que o valor de α é auto-ajustável e não exige nenhum esforço preliminar de calibração desse parâmetro.

A RCL é atualizada por meio da remoção de elementos cujos custos não são mais viáveis, e da inclusão dos recém-economicamente viáveis. Este procedimento é repetido até que não haja mais possibilidades de seleção de nenhum item, isto é, até que a lista não contenha nenhum item e uma solução viável é obtida [Butenko et al., 2004; Ribeiro, 2005].

Delorme et al. [2004] investigaram três fases construtivas e melhorias do GRASP para o problema do empacotamento de conjuntos. Resende & Ribeiro [2010] apresentam seis diferentes implementações da fase construtiva.

2.2.1.2 Fase Busca Local

A fase de busca local (Algoritmo 5) tenta melhorar a solução encontrada na fase construtiva ao pesquisar a vizinhança da solução atual e verificar se uma melhor solução pode ser encontrada. Se uma solução melhor é encontrada, essa solução é então armazenada como uma solução corrente e uma nova vizinhança é pesquisada. Quando nenhuma solução melhor puder ser encontrada, a busca é terminada e o ótimo local é

retornado [Butenko et al., 2004].

Algoritmo 5: *BuscaLocal*$(P, N(P), s)$

```
1 repeat
2 |   Encontre uma melhor solução t ∈ N(s);
3 |   s ← t;
4 |   AdaptarFunçãoGulosa(s);
5 until s não é localmente ótimo;
6 return s como solução ótima em P
```

Nascimento [2010] alerta que os algoritmos de busca local desconhecem a distância do ótimo global ao ótimo local, ficando muito dependentes da solução inicial para gerar soluções finais de boa qualidade. Neste caso, o GRASP atua com eficiência, visto que a cada iteração, uma nova solução inicial é fornecida como ponto de partida para a busca local. Range et al. [2000] desenvolveram o GRASP Restrito, onde soluções iniciais supostamente ruins são descartadas, não passando para a fase de busca local, diminuindo assim o tempo de execução do algoritmo.

Como exemplo, Carreto & Baker [2002] utilizaram o GRASP para construção de rotas de veículos e como busca local, a heurística 3-opt. Mavridou et al. [1998] utilizaram o GRASP para o problema de alocação biquadrática e a busca local foi feita com o algoritmo 2-exchange.

2.2.1.3 GRASP aplicado ao problema de agrupamento

O problema de agrupamento pode ser visto como um problema de otimização. É necessário maximizar ou minimizar os valores de uma ou mais funções objetivo a fim de medir a similaridade e/ou dissimilaridade entre os objetos analisados. Assim como outras metaheurísticas, o GRASP tem sido utilizado na resolução de problemas de agrupamento.

Areibi & Vannelli [1997] utilizaram o GRASP para particionar circuitos elétricos e gerar grupos de tamanhos moderados. O número de grupos é determinado em função do número de partições necessárias. Inicialmente, a heurística lê a descrição do circuito e redimensiona os blocos a serem utilizados pelo GRASP, que utiliza apenas a fase de construção para gerar o número de grupos necessários. O heurística *hill climbing* é usada como busca local para melhorar a solução inicial gerada.

Carreto & Baker [2002] utilizaram o GRASP na construção de rotas de veículos. O objetivo é reduzir a distância total percorrida por cada veículo. Na fase construtiva é implementa as rotas agrupando os clientes de acordo com os veículos definidos pelas sementes, selecionadas pela aplicação da heurística 3-opt. A função gulosa leva em

consideração as rotas com menor custo de inserção e clientes com a maior diferença entre o menor e o segundo menor custo de inserção e o menor número de rotas que podem atravessar. Como fase de busca local, o 3-opt é usado.

Cano et al. [2000, 2002] propuseram um novo método de inicialização do algoritmo k-means, utilizando o GRASP. Segundo esses autores, o k-means é eficiente e tem baixo custo computacional, mas converge para mínimos locais. A fase construtiva do GRASP corrige esse problema, melhorando a abrangência da busca. Na fase construtiva, os autores utilizaram o algoritmo de inicialização Kaufman. A busca local é feita utilizando o k-means.

Nascimento et al. [2010b] utilizaram o GRASP para o problema de agrupamento de dados biológicos. Os dados foram representados em forma de grafo, e o objetivo foi a minimização da soma total das distâncias intra-grupo. Nesse estudo, os autores linearizaram o modelo original proposto por Rao [1971]. A versão linearizada do modelo, está descrita na equação 2.6 que se segue:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} y_{ij} \tag{2.6}$$

sujeito a

$$\sum_{k=1}^{M} x_{ik} = 1, \quad i = 1, ..., N \tag{2.7}$$

$$\sum_{i=1}^{N} x_{ik} \geq 1, \quad k = 1, ..., M \tag{2.8}$$

$$x_{ik} \in \{0, 1\}, \quad i = 1, ..., N, \quad k = 1, ..., M \tag{2.9}$$

$$y_{ij} \geq x_{ik} + x_{jk} - 1, \quad i = 1, ..., N, \quad j = i + 1, ..., N, \quad k = 1, ..., M \tag{2.10}$$

$$y_{ij} \geq 0 \quad i = 1, ..., N, \quad j = i + 1, ..., N \tag{2.11}$$

A função objetivo visa minimizar a distância entre os objetos localizados no mesmo grupo onde d_{ij} representa a distancia (ou dissimilaridade) entre os objetos i e j; N é o número de objetos; M é o número de grupos; x_{ik} é uma variável binária que assume o valor 1 se o objeto i pertence ao grupo k e 0, em caso contrário. A restrição 2.7 garante que o objeto i pertença a apenas um grupo. A restrição 2.8 garante que o grupo k contenha pelo menos um objeto no seu interior. A restrição 2.9 garante que as variáveis x_{ik} sejam binárias. As restrições 2.10 e 2.11 asseguram que y_{ij} assume o valor 1 se ambas as variáveis x_{ik} e x_{jk} são 1. O modelo proposto tem $N^2/2$ mais variáveis e $N(N-1)(M+1)/2$ mais restrições que o modelo não-linear.

No algoritmo proposto por Nascimento et al. [2010b], inicialmente, todos os elementos analisados são modelados como nós de um grafo G. A princípio todos os nós de G pertencem a único grupo. As arestas do grafo são gradualmente eliminadas e o procedimento pára quando se atinge os k grupos desejados, ou seja, quando a solução torna-se factível. Na fase de busca local, movimentos[5] de nós entre grupos são analisados a fim de verificar se o movimento minimiza a soma dos pesos das arestas dos grupos analisados. Por fim, os resultados obtidos são avaliados com o índice CRand. O CRand [Hubert & Arabie, 1985], é uma medida externa de avaliação, ou seja, usa informações obtidas fora do processo de agrupamento, como o conhecimento da estrutura verdadeira do grupo. O índice compara duas partições por meio da concordância e discordância entre pares de objetos e tem um valor máximo de 1 se as partições são iguais. Por outro lado, valores de CRand próximos a 0 indicam grande dissimilaridade entre os grupos gerados e os da estrutura real. Em suma, o CRand avalia a diferença entre duas partições [Milligan, 1996; Milligan et al., 2009; Nascimento, 2010]. O CRand é obtido pela Fórmula 2.12:

$$CRand = \frac{\sum_{i=1}^{K^a}\sum_{j=1}^{K^b}\binom{|C_i^a \cap C_j^b|}{2} - \left[\sum_{i=1}^{K^a}\binom{|C_i^a|}{2}\sum_{j=1}^{K^b}\binom{|C_j^b|}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i=1}^{K^a}\binom{|C_i^a|}{2} + \sum_{j=1}^{K^b}\binom{|C_j^b|}{2}\right] - \left[\sum_{i=1}^{K^a}\binom{|C_i^a|}{2}\sum_{j=1}^{K^b}\binom{|C_j^b|}{2}\right]/\binom{n}{2}} \quad (2.12)$$

onde a e b são as duas partições a serem comparadas, K^a e K^b, o número de grupos das partições a e b, respectivamente, e C_i^a e C_j^b , o grupo i de a e o grupo j de b, respectivamente. O valor de CRand varia no intervalo [-1,1]. Quanto mais próximo de 1 esse valor estiver, mais concordantes são as duas partições.

Nesta dissertação aproveitam-se as idéias de Nascimento et al. [2010b] para agrupamento de dados biológicos utilizando o GRASP para desenvolver um método híbrido, no qual o *Path-Relinking* é usado para intensificar e diversificar a busca dentro do espaço de soluções. De fato, Nascimento [2010] afirma ter realizado testes utilizando o GRASP com *Path-Relinking*, mas não obteve resultados significativos. Segundo a autora, uma das causas pode ter sido o *Path-Relinking* proposto.

2.2.2 *Path-Relinking*

O *Path-Relinking* é uma generalização do *Scatter Search*. Como métodos evolutivos, eles operam com uma população de soluções ao invés de com uma única solução, e

[5]No contexto do problema de agrupamento, um movimento é visto como a transferência de um objeto de um grupo para outro, o qual é possibilitado ao se constatar a melhoria da função objetivo definida para o problema.

empregam procedimentos que combinam estas soluções para criar novas. Essa metaheurística gera novas soluções explorando trajetórias que conectam soluções de alta qualidade - iniciando de uma dessas soluções, chamada solução inicial, e gerando um caminho no espaço de vizinhança que conduz a outras soluções, chamadas soluções guias. Para criar os caminhos, movimentos entre as soluções inicial e guia são estrategicamente selecionados.

Glover & Kochenberger [2003] explicam que a abordagem é chamada *path-relinking* em virtude da geração de novos caminhos entre soluções previamente ligadas por uma série de movimentos executados durante a busca ou pela geração de caminhos entre soluções previavente ligadas a outras soluções, mas não uma a outra.

Considere a criação de caminhos que unem duas soluções selecionadas x' e x'', restringindo atenção para a parte do caminho que se situa entre estas soluções, produzindo uma solução $x' = x(1), x(2), ..., x(r) = x''$. Para reduzir o número de opções a serem consideradas, a solução $x(i+1)$ deve ser criada a partir de $x(i)$ e a cada passo, escolhendo um movimento que reduz o número de movimentos faltantes para chegar a x'' [Glover et al., 2000].

A Figura 2.2 ilustra os passos realizados pelo *Path-Relinking* para o problema de agrupamento. O algoritmo inicia calculando os movimentos necessários para transformar a solução inicial, na solução final. O algoritmo escolhe o movimento que forneça a melhor função objetivo (linhas em destaque). O procedimento é repetido até que a solução inicial se pareça com a solução final, ou quando é imposto outro critério de parada ao algoritmo. A intenção do *Path-Relinking* é encontrar soluções de qualidade no percurso.

O procedimento do *Path-Relinking* inicia computando a diferença simétrica entre duas soluções. A diferença simétrica é a cardinalidade do conjunto de movimentos necessários para sair da solução inicial e chegar na solução guia. Um caminho de soluções é gerado ligando a solução inicial a solução guia, a melhor solução encontrada no percurso é retornada pelo algoritmo. A cada passo, o procedimento examina todos os movimentos a partir da solução corrente e seleciona o movimento que resulta no menor custo quando este é incorporado à solução corrente. O melhor movimento é realizado e o conjunto de possíveis movimentos é atualizado. Se necessário, a melhor solução é atualizada. O procedimento termina quando a solução final é encontrada, ou seja, quando a diferença simétrica entre a melhor solução e a solução final for 0 [Resende & Ribeiro, 2005].

Para que o *Path-Relinking* seja efetivamente aproveitado, é interessante considerar uma diferença simétrica mínima de 4 componentes entre os pares de soluções participantes para que seja possível obter um mínimo local entre as soluções. A prova

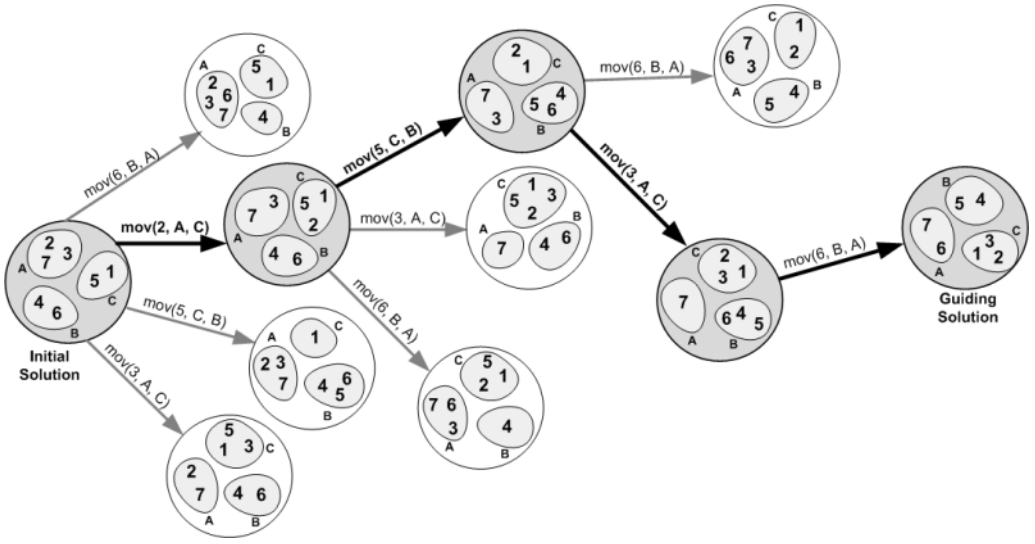

Figura 2.2. Exemplo de *Path-Relinking* aplicado ao problema de agrupamento de dados.

do valor mínimo para diferença simétrica, também chamada distância *Hamming*, está descrita com detalhes em [Ribeiro & Resende, 2010].

O pseudocódigo do *Path-Relinking* é apresentado na Figura 6 [Resende & Ribeiro, 2005].

Algoritmo 6: *PathRelinking(SoluçãoInicial, SoluçãoFinal)*

```
1 Δ ← CalcularDiferençaSimétrica(SoluçãoInicial, SoluçãoFinal);
2 MelhorFunção ← Menor{f(SoluçãoInicial), f(SoluçãoFinal)};
3 MelhorSolução ← MenorArgumento{f(SoluçãoInicial), f(SoluçãoFinal)};
4 while Δ ≠ ∅ do
5 |   MelhorMovimento ← SelecionaMelhorMovimento Δ(SoluçãoInicial, SoluçãoFinal);
6 |   RetiraMovimentoListaMovimentos(MelhorMovimento);
7 |   MelhorSolução ← MelhorSolução + MelhorMovimento;
8 |   Se necessário AtualizaMelhorFunção() e AtualizaMelhorSolução();
9 end
```

Resende & Ribeiro [2010] afirmam que diversas alternativas têm sido consideradas e combinadas em recentes implementações do *Path-Relinking*, incluindo variações como *Forward*, *Backward*, *Back and Forward*, *Mixed*, *Truncated* e *Greedy Randomized Adaptive*. Tais variações são explicadas abaixo [Resende & Ribeiro, 2005]:

- ***Forward Relinking***: o *Path-Relinking* é aplicado usando a pior solução entre x_s e x_t como solução inicial e a melhor solução como guia;
- ***Backward Relinking:*** o *Path-Relinking* é aplicado usando a melhor solução entre x_s e x_t como solução inicial e a pior como a solução guia;

- ***Back and forward relinking:*** duas trajetórias são exploradas, primeiro usando o x_s como solução inicial e depois usando o x_t como solução inicial. A desvantagem é o aumento do tempo computacional.

- ***Mixed relinking:*** dois caminhos são explorados simultaneamente: o primeiro proveniente de x_s e o segundo de x_t, até que eles se encontram em uma solução intermediária equidistante de x_s e de x_t;

- ***Greedy randomized adaptive relinking:*** ao invés de selecionar o melhor movimento ainda não selecionado na trajetória, seleciona-se aleatoriamente um elemento de uma lista dos candidatos mais promissores no caminho a ser investigado;

- ***Truncated relinking:*** apenas uma parte da trajetória entre x_s e x_t é investigada.

2.2.3 GRASP com *Path-Relinking*

O uso do *Path-Relinking* como estratégia de intensificação do GRASP foi proposto por Laguna & Marti [1999]. Segundo Resende & Ribeiro [2003], o *Path-Relinking* consiste em um grande avanço no procedimento GRASP padrão, levando a melhorias significantes de desempenho e qualidade das soluções. Resende & Ribeiro [2005] apresentam algumas aplicações do uso do GRASP com *Path-Relinking* (GRASP+PR). Outras aplicações podem ser vistas em Alvarez-Valdes et al. [2008], Nascimento et al. [2010a], Resende et al. [2010a] e Arroyo et al. [2010].

O pseudocódigo do Algoritmo 7 apresenta o GRASP+PR. Nas linhas 4 e 5, o GRASP básico é executado. A partir da linha 6, é realizada a fase de intensificação do GRASP+PR. O algoritmo PR é aplicado sobre a solução obtida pela busca local e uma solução aleatoriamente selecionada a partir do conjunto elite. A melhor solução encontrada ao longo desta trajetória é também considerada como uma candidata a inserção no conjunto elite. Ao final das iterações GRASP, a fase de pós-otimização combina as soluções do conjunto elite na busca por melhores soluções.

Como se vê, GRASP+PR mantém um conjunto de soluções elite P. É um requisito para P que ele contenha soluções diversificadas e de qualidade. Nesse sentido, para inclusão de uma solução S em P, S deve ser melhor do que a pior solução presente em P, mas para preservar a diversidade de P, a distância $d(P, S)$ entre S e todas as soluções presentes em P deve ser maior que um determinado limiar Δ. No entanto, esta última condição é substituída quando a melhor solução do P é atualizada. Inicialmente, P é preenchido com as $|P|$ soluções geradas pelo GRASP e são armazenadas de forma ordenada de acordo com a função objetivo [Villegas et al., 2011].

Algoritmo 7: *GRASP+PR(SoluçãoInicial, SoluçãoFinal)*

```
   Data: Número de iterações i_max
   Result: Solução x* ∈ X
 1 P ← ∅;
 2 f* ← ∞;
 3 for i=1, ..., i_max do
 4 |   x ← ConstruçãoGulosaAleatória();
 5 |   x ← BuscaLocal(x);
 6 |   if i ≥ 2 then
 7 |   |   Escolha aleatoriamente no pool soluções γ ⊆ P a ser reconectada a x;
 8 |   |   for y ∈ γ do
 9 |   |   |   Determinar qual (x ou y) será solução inicial x_s e qual será guia x_t;
10 |   |   |   x_p ← PathRelinking(x_s, x_t);
11 |   |   |   Atualiza o conjunto elite P com x_p;
12 |   |   |   if f(x_p) < f* then
13 |   |   |   |   f* ← f(x_p);
14 |   |   |   |   x* ← x_p;
15 |   |   |   end
16 |   |   end
17 |   end
18 end
19 P=PosOtimizaçãoP;
20 x*=menorArgumentof(x), x ∈ P;
```

Resende et al. [2010b] explicam que inicialmente, o conjunto de soluções elite P está vazio (linha 1). Cada solução factível gerada pelo GRASP e também as resultantes da aplicação do PR são consideradas soluções candidatas para P. A atualização do conjunto de soluções elite (linha 11) se dá da seguinte forma: Se P ainda não está cheio, a solução candidata é adicionada a P, caso ela seja diferente de todas as soluções já presentes em P. Se P está cheio, e a solução candidata é melhor que a incubente, então tal solução substitui uma do conjunto. Se a solução candidata é melhor do que o pior elemento de P, então ela substitui algum elemento de P se atender o limiar Δ entre todas as outras soluções atuais de P. O PR é então realizado entre a solução gerada pelo GRASP e uma solução aleatoriamente selecionada em P. Detalhes sobre as estratégias utilizadas para permitir a diversificação e qualidade das soluções presentes em P são descritos na Seção 3.4.

Capítulo 3

Metodologia

Neste capítulo será apresentada a metodologia utilizada. As atividades realizadas podem ser acompanhadas pelo fluxograma da Figura 3.1.

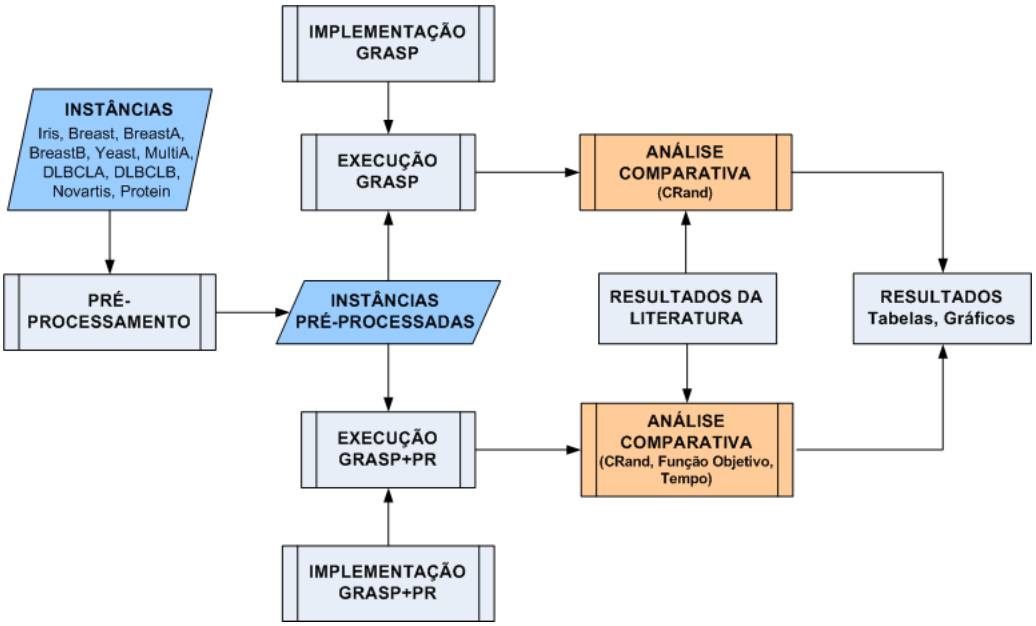

Figura 3.1. Fluxograma das atividades realizadas durante o desenvolvimento do projeto.

Inicialmente as instâncias foram coletadas e pré-processadas resultando nas instâncias preparadas para serem utilizadas nas execuções do GRASP e variantes do *Path-Relinking*. Na atividade Implementação GRASP, o GRASP foi implementado e em seguida executado com as instâncias pré-processadas. Os resultados obtidos após as execuções do GRASP, foram comparados com os resultados da literatura e permitiu a geração das tabelas e gráficos descritos no Capítulo 4. Na atividade Implementação

GRASP+PR, foram realizadas as implementações das variantes propostas e os resultados de cada variante foram comparados entre si e com os resutados da literatura. A descrição detalhada de cada atividade e seus respectivos produtos serão apresentados nas seções a seguir.

3.1 Instâncias

Esta atividade teve como objetivo pesquisar e coletar as instâncias utilizadas nos experimentos realizados por Nascimento et al. [2010b]. Nos repositórios onde as instâncias são encontradas, além dos arquivos de dados (objetos e respectivos atributos), são disponibilizados também um arquivo de teste e as informações relacionadas ao conteúdo de tais instâncias. As instâncias foram organizadas em 4 grupos principais que representam o tipo de problema onde foram utilizados:

1. *Classificação de Dobras (Fold Classification)*;
2. *Predição da Localização de Proteínas (Prediction of Protein Localization Sites)*;
3. *Diagnóstico de Câncer*;
4. *Problemas Taxonômicos*;

A Tabela 3.1 mostra as principais características de cada conjunto de dados. A primeira coluna descreve o nome da instância, a segunda e terceira colunas descrevem respectivamente a quantidade de objetos e atributos, e a quarta coluna descreve se a instância possui mais de uma estrutura de agrupamento, e entre parênteses a quantidade de grupos que cada estrutura possui.

Tabela 3.1. Principais características das bases de dados utilizadas nos experimentos.

Instância	**#Objetos**	**#Atributos**	**#Est(#Grupos)**
Protein	694	125	2 (4,27)
Yeast	1484	8	1 (10)
Breast	699	9	2 (2,8)
Novartis	103	1000	1 (4)
BreastA	98	1213	1 (3)
BreastB	49	1213	2 (2,4)
DLBCLA	141	661	1 (3)
DLBCLB	180	661	1 (3)
MultiA	103	5565	1 (4)
Iris	150	4	1 (3)

A seguir cada instância será descrita com detalhes.

3.1.1 Classificação de Dobras *(Fold Classification)*

Problema inicialmente formulado e estudado por Turcotte et al. [2001]. A tarefa consiste em classificar proteínas de estrutura conhecida (ex. proteínas experimentalmente separadas e armazenadas em um banco de proteínas), dado o seu alto-nível de descrições sobre sua estrutura secundária e sequência de aminoácidos. SCOP (veja http://scop.mrc-lmb.cam.ac.uk/scop/) é um banco de proteínas hierarquicamente organizada de acordo com suas propriedades estruturais.

A instância **Protein** [Ding & Dubchak, 2001] foi a utilizada para experimentos do tipo *Fold Classification*. É composta por 694 objetos que correspondem a dobras[1] de proteínas com 125 atributos. São duas estruturas de agrupamento: A primeira (Protein1) é dividida em 4 tipos de dobras resultando em grupos com 115, 226, 258 e 95 objetos, enquanto a segunda (Protein2) estrutura representa a divisão de 27 subtipos de dobras com 19, 16, 32, 15, 18, 15, 74, 21, 29, 13, 16, 32, 12, 13, 16, 77, 23, 24, 40, 22, 17, 22, 18, 15, 15, 40, 40 objetos. Protein pode ser obtida em http://ranger.uta.edu/∼chqding/protein/.

3.1.2 Predição da Localização de Proteínas *(Prediction of Protein Localization Sites)*

Para funcionar adequadamente, as proteínas devem ser transportadas para vários locais (unidades celulares) dentro de uma célula. A localização de uma proteína na célula afeta suas funcionalidades, e a sua eficiência nos tratamentos através de drogas. Segundo Horton & Nakai [1997], as informações necessárias para a localização correta da proteína é geralmente encontrada na sua própria sequência de aminoácidos.

Combinar uma grande variedade de sequências é uma tarefa custosa, o que vem justificar a utilização de ferramentas de classificação e aprendizado de máquina para reduzir o trabalho de especialistas e tentar aumentar a acurácia das classificações. Nakai & Kanehisa [1991, 1992] realizaram experimentos *Prediction of Protein Localization Sites* (PPLS) utilizando o kNN (*k Nearest Neighbor*), árvore de decisão binária e o classificador Bayes Naïve.

Yeast [Nakai & Kanehisa, 1991] é uma base de dados com objetos correspondentes a 1484 proteínas de levedura com 8 atributos para cada objeto relativos às características calculadas a partir das seqüências de aminoácidos. Yeast classifica suas proteínas em 10 grupos diferentes: citoplasmática ou citoesqueleto (463 objetos), nuclear (429 objetos), mitocondrial (244 objetos), proteína de membrana sem

[1] Ao dobrar e enrolar-se para tomar uma forma tridimensional específica, as proteínas são capazes de realizar a sua função biológica.

sinal N-terminal (163 objetos), proteína de membrana sem mórula com sinal de divisão (51 objetos), proteína de membrana com mórula com sinal de divisão (44 objetos), extracelular (37 objetos), vacuolar (30 objetos), peroxissomal (20 objetos), e localizadas no lúmen do retículo endoplasmático (5 objetos). Yeast pode ser obtida em http://archive.ics.uci.edu/ml/datasets/Yeast.

3.1.3 Diagnóstico de Câncer

Pesquisas tem demonstrado o poder das técnicas de agrupamento como ferramenta que facilita o processo de diagnóstico de doenças. A correta identificação de tumores cancerígenos é uma tarefa clinicamente desafiadora. Identificar tumores distintos e que possuem aparência morfologicamente semelhantes é importante, já que cada tipo de tumor exige formas diferentes de tratamento.

Segundo DeRisi et al. [1996] o advento dos *microarrays*[2] de genes forneceu aos pesquisadores e clínicos a habilidade de analisar milhares de genes simultaneamente. Os níveis de expressão desses *microarrays* possibilitam a diferenciação e classificação de tumores em subclasses. Evidências sugerem que terapias específicas para o tipo de tumor são essenciais para um tratamento eficiente e com um mínimo de toxidade para o paciente. Neste trabalho serão consideradas 7 bases de dados com conteúdo voltado para diagnóstico de câncer.

Breast [Bennett & Mangasarian, 1992] é uma base de dados de células cancerosas que possui 699 objetos compostos por 9 atributos. A classificação dessas células é dada segundo a sua natureza, como maligna ou benigna, confirmada pela técnica *fine needle aspirate* (FNA). FNA permite a biópsia por tecidos recuperados por meio de análise microscópica. Os atributos desse conjunto de dados são: a espessura do grupo, a uniformidade do tamanho da célula, a uniformidade do formato da célula, a adesão marginal, o tamanho da célula epitelial única, o núcleo vazio, a cromatina suave, o nucléolo normal e a mitose. Essa instância possui duas estruturas de agrupamento, a primeira (Breast1) é composta por 2 grupos sendo um com 241 objetos que representam células de características benigna e outro com 458 objetos de característica malígna. A segunda (Breast2) estrutura de agrupamento é composta por objetos inseridos em ordem cronológica, possui 8 grupos que contém 367, 70, 31, 17, 48, 19, 31 e 86 objetos. Breast por ser obtida em http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29.

Novartis [Su et al., 2002; Monti et al., 2003] é uma base de dados de expressão gênica com 103 objetos de tecidos cancerosos. Tais objetos são agrupados em 4

[2]É uma técnica experimental da Biologia Molecular que busca medir de forma simultânea uma quantidade em larga escala de transcritos.

diferentes tipos de câncer: 26 de mama, 26 de próstata, 28 de pulmão e 23 de cólon. Cada objeto tem 1000 atributos, que correspondem a níveis de expressão de 1000 genes. A base em questão é uma versão pré-processada da disponibilizada por Monti et al. [2003].

O conjunto de dados **MultiA** [Su et al., 2002] tem os mesmos objetos da Novartis, mas foram pré-processadas por Hoshida et al. [2007], e apresentam 5565 atributos.

BreastA [Van't et al., 2002] apresenta 98 objetos com 1213 atributos. Os dados representam tumores de mama obtidos através de *microarray* de oligonucleotídeos com um e com dois canais. Hoshida et al. [2007] apresentaram uma análise do conjunto de dados BreastA dividindo-o em 3 grupos com 11, 51 e 36 objetos.

BreastB [West et al., 2001] apresenta 48 objetos com 1213 atributos. Os objetos são do mesmo tipo da instância BreastA também foram obtidos através de *microarray* de oligonucleotídeos com um e com dois canais. BreastB possui duas estruturas de agrupamento: a primeira (BreastB1) foi decomposta em 2 grupos levando-se em consideração o receptor de estrogênio (RE) sendo 25 objetos de RE positivo (RE+) para um grupo e 24 objetos de RE negativo (RE-) para outro grupo. A segunda estrutura (BreastB2) foi decomposta em 4 grupos sendo 13 tipos de tumores ER+ lymphnode(LN)+, 12 ER- LN+, 12 ER+ LN- e 12 ER- LN-.

A instância **DLBCLA** [Monti et al., 2005], apresenta exemplos de *Diffuse large B-cell lymphoma* (linfoma difuso de grandes células-B). É composta por 141 objetos com 661 atributos cada que podem ser agrupados de acordo com três mecanismos moleculares relevantes: 49 objetos fosforilação oxidativa (OxPhos), 50 objetos de respostas das células-B (BCR) e 42 objetos de resposta de hospedeiros (HR).

A instância **DLBCLB** [Rosenwald et al., 2002]) é uma versão pré-processada de 180 objetos e 661 atributos [Hoshida et al., 2007]. Assim como a base de dados DLBCLA, os objetos são agrupados de acordo com os mecanismos moleculares BCR com 12 objetos, OxPhos com 51 e HR com 87 objetos.

As instâncias Novartis, MultiA, BreastA, BreastB, DLBCLA e DLBCLB podem ser obtidas em (http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi)

3.1.4 Problemas Taxonômicos

No cenário biológico problemas taxonômicos envolvem a classificação de seres vivos com base nas suas características fenóticas[3] e genóticas[4]. Técnicas de agrupamento

[3]Características observáveis ou caracteres de um organismo como: morfologia, desenvolvimento, propriedades bioquímicas, fisiológicas e comportamento.

[4]Presença de genes que constitue o material hereditário herdado dos progenitores.

vêm sendo utilizadas na tentativa de agilizar e/ou melhorar a tarefa de classificação taxonômica de seres.

Iris é uma base clássica de análise discriminante de tipos de flores proposta por Fisher et al. [1936], que contém 150 exemplos de três espécies diferentes de flores Iris: *Iris setosa*, *Iris virgínica* e *Iris versicolor*. Na Iris, cada espécie de flor tem 50 exemplos, cada um com 4 atributos correspondendo à largura e espessura da pétala e da sépala. A instância Iris pode ser obtida em http://archive.ics.uci.edu/ml/datasets/Iris.

3.2 Pré-Processamento

Nesta atividade, as instâncias foram preparadas para os experimentos por meio da limpeza e padronização das bases de dados. Tanto na base de dados quanto no arquivo de teste, foram retirados dados não relevantes para o processo de agrupamento (ex. cabeçalho, informações sobre a instância, etc). Os objetos foram organizados em linhas e seus respectivos atributos organizados em colunas.

A normalização é a tarefa utilizada para padronizar o valor dos dados, que por ventura se apresentam muito discrepantes, em um *range* de valores pequenos como -1.0 a 1.0, ou 0.0 a 1.0. A instância Novartis foi a única a exigir esta tarefa. Utilizou-se a Medida Normalizada ou **z-score** descrita por Han & Kamber [2006]. O cálculo do *z-score* é realizado de acordo com a Fórmula 3.1:

$$z = \frac{v - \mu}{\sigma} \tag{3.1}$$

onde z é o valor do objeto após a normalização, v é a valor a ser normalizado, μ a média e σ o desvio médio absoluto (DMA). Tanto a média quanto o DMA são obtidos considerando os valores de todos os objetos para cada atributo. Ainda segundo Han & Kamber [2006], o DMA é mais robusto a *outliers* que o desvio padrão e de certa forma contribui para redução de objetos com essas características.

Como resultado da tarefa de Pré-Processamento, tem-se as instâncias preparadas para serem computadas nos algoritmos propostos.

3.3 Implementação GRASP

Após a coleta e preparação das instâncias, iniciou-se a atividade de implementação. O GRASP é composto por três partes. A parte principal é detalhada no Algoritmo 8 e tem como principais funções chamar as fases Construtiva (linha 3), e Busca Local (linha 4), além de verificar se houve melhoria na função objetivo. O algoritmo toma

como entrada a instância I, a quantidade M de grupos desejados, a quantidade máxima de iterações sem melhorias Ism, a distância ou correlação DC e a semente S utilizada pelo método aletório. Após a realização de todos os passos, o algoritmo retorna com a melhor solução encontrada pelo GRASP (linha 10).

A linha 2 contém o critério de parada utilizado pelo GRASP. O critério de parada escolhido foi a quantidade máxima de iterações sem melhorias (Ism), cujo valor é passado como parâmetro de entrada para o programa.

O comando condicional da linha 5 verifica se o valor da função objetivo da solução retornada pela busca local, é melhor que a última solução encontrada. A variável x^* armazena a melhor solução encontrada e f^* armazena o respectivo valor da função objetivo. As linhas 6 e 7 visam atualizar as variáveis supracitadas em caso de melhoria.

Algoritmo 8: GRASP()

```
   Data: I M Ism DC S
   Result: Solução x* ∈ X
 1 f* ← ∞;
 2 while critério de parada não for alcançado do
 3 |   x ← FaseConstrutiva(·);
 4 |   x ← FaseBuscaLocal(x);
 5 |   if f(x) < f* then
 6 |   |   f* ← f(x);
 7 |   |   x* ← x;
 8 |   end
 9 end
10 return x*
```

O GRASP utilizado neste trabalho tem como base o modelo proposto por Nascimento et al. [2010b]. Tal abordagem considera os dados de uma instância como nós de um grafo *G=(V, E)*, onde V é o conjunto de nós N que representam os objetos do conjunto de dados, e E é o conjunto de arestas ponderadas e não direcionadas *(i, j)*, sendo que uma aresta conecta o nó i ao nó j. O peso associado a aresta que liga os nós i e j representa a distância (ou correlação) entre esses objetos. Quanto menor o peso da aresta maior a similaridade entre os objetos. Detalhes sobre medidas de similaridade podem ser obtidos no capítulo 2.

Inicialmente a instância é vista como único grupo e neste caso os dados são representados sob a forma de um grafo completo, o que quer dizer que cada nó possui uma aresta que o conecta a cada outro nó do grafo.

A Figura 3.2 mostra as etapas realizadas durante o GRASP as quais estão detalhadas a seguir.

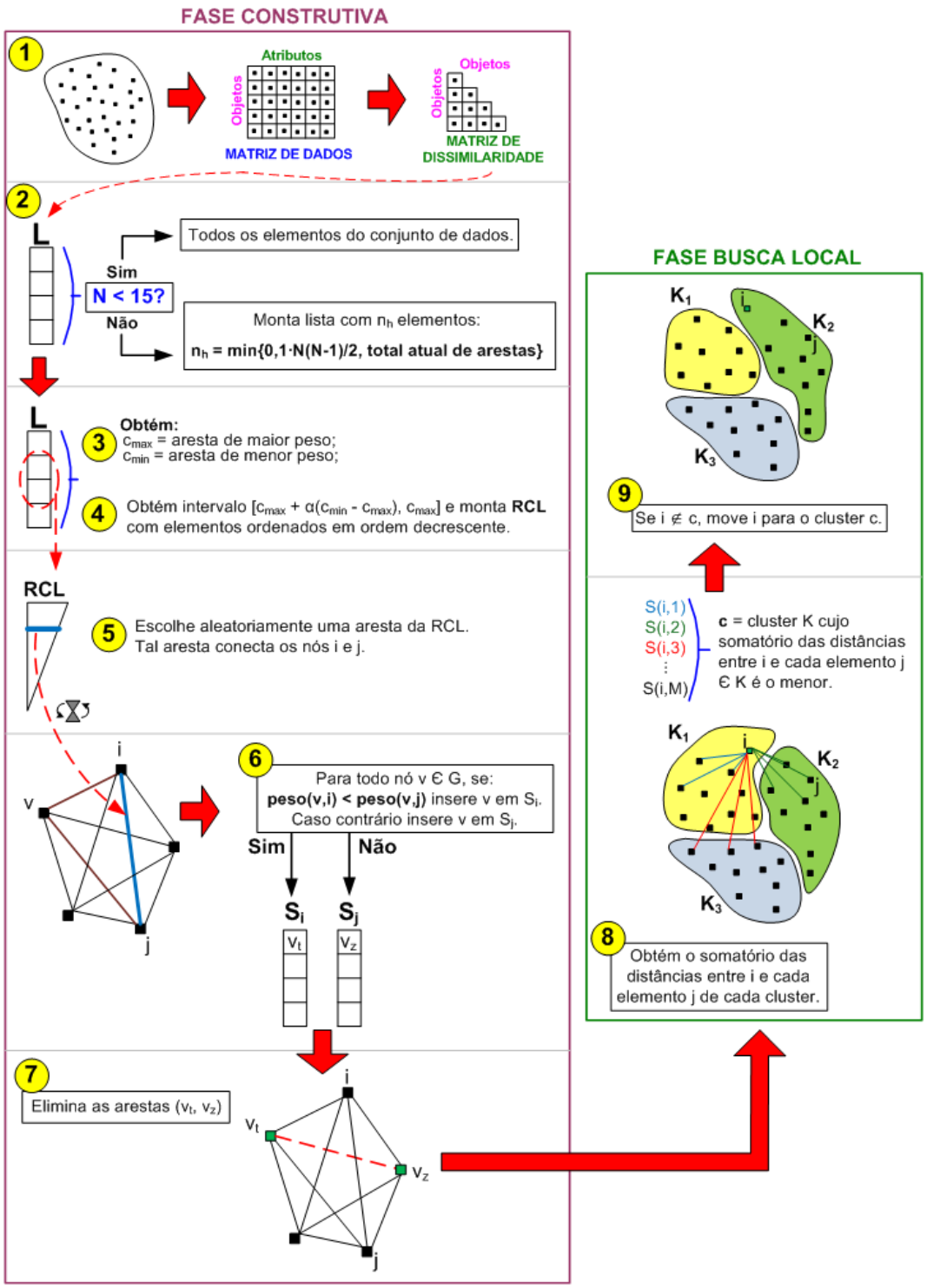

Figura 3.2. Fluxograma das etapas realizadas durante o GRASP.

3.3.1 Fase Construtiva

O GRASP tem inicio na etapa **1** onde os dados de entrada são lidos e armazenados em uma matriz de dados (objeto x atributos). A partir da matriz de dados, é construída a matriz de dissimilaridade (objeto x objeto) que armazena os pesos entre os objetos.

A etapa **2** tem o objetivo de construir a lista L a partir da matriz de dissimilaridade. Esta lista tem a função de armazenar as arestas considerando um dos dois critérios: caso a quantidade de objetos N seja menor que 15, L é preenchido com todas as arestas do Grafo, em caso contrário, L é preenchida com as n_h arestas de maior peso do Grafo, sendo n_h = min{0.1*N(N-1)/2; total atual de arestas}. Nascimento [2010] utilizou essa restrição como forma de melhorar o tempo computacional do algoritmo e garantir uma melhor estabilidade das soluções encontradas. A autora também observou que esse limite gera soluções de boa qualidade em um tempo computacional razoável. Neste trabalho, utilizou-se a busca binária na implementação da lista L, de forma a reduzir o tempo de busca e inserção de arestas mais promissoras na mesma.

A partir da lista L, na etapa **3** as arestas de maior e menor pesos (c_{max} e c_{min}) são obtidas e a na etapa **4** a RCL (*Restrict Candidate List* - Lista Restrita de Candidatos) é construida considerando o intervalo $[c_{max} + \alpha(c_{min} - c_{max}); c_{max}]$, onde α é um número escolhido aleatoriamente no intervalo [0,1]. A RCL é ordenada de forma decrescente e é formada pelos melhores elementos, isto é, aqueles cuja incorporação na solução parcial/corrente resultam em melhores custos incrementais.

Na etapa **5**, uma aresta é escolhida de forma aleatória. Tal aresta conecta os nós i e j, que servirão de referência para construção das listas S_i e S_j descritas na etapa **6**. Na lista S_i são armazenados todos os nós v que estão mais próximos de i do que de j, ou seja, $v \in G : peso(v, i) < peso(v, j)$, e na lista S_j o contrário.

A etapa **7** corresponde ao final da fase construtiva, e é responsável pela eliminação gradual das arestas $(v_t, v_z) \in V$ onde $v_t \in S_i$ e $v_z \in S_j$. Esse processo objetiva a construção de subgrafos totalmente desconexos (cliques) e continua de forma que, a cada passo, uma nova componente conexa que representa um novo grupo da partição, seja criada. Tal procedimento pára quando se obtém a quantidade M de grupos, e a solução torna-se factível.

O Algoritmo 9 contém os procedimentos realizados durante a Fase Construtiva. A linha 1 descreve a estrutura de repetição do GRASP, sendo que o número de repetições é determinado por M que representa a quantidade de grupos desejados (etapa 7 da Figura 3.2). A estrutura condicional descrita nas linhas 2 a 6, limita o tamanho da lista L considerando a quantidade de objetos do grupo (etapa 2 da Figura 3.2).

A linha 7 obtém as arestas de maior e menor peso (c_{max} e c_{min}) a partir da lista

L (etapa 3 da Figura 3.2). A linha 8 cria a RCL considerando os objetos situados no limite $[c_{max} + \alpha(c_{min} - c_{max}); c_{max}]$ (etapa 4 da Figura 3.2). A linha 9 (corresponde a etapa 5 da Figura 3.2), faz a escolha de uma aresta da RCL de forma aleatória.

As linhas 10 a 16, contém os procedimentos necessários para criação das listas S_i e S_j (etapa 6 da Figura 3.2). E por fim, a linha 17 faz a eliminação das arestas (v_t, v_z), procedimento que corresponde a etapa 7 da Figura 3.2.

Algoritmo 9: FaseConstrutiva()

```
   Data: Matriz de Distâncias
   Result: Solução x* ∈ X
 1 for i = 0 to M − 1 do
 2     if N<15 then
 3         Considere L uma lista com cada aresta do grafo;
 4     else
 5         Considere n_h = min{0.1*N(N - 1)/2, número total de arestas atuais} e seja L uma
           lista com as n_h arestas de maior peso;
 6     end
 7     Considere c_max e c_min, respectivamente as arestas de maior e menor pesos de L;
 8     Escolha aleatoriamente um índice α ∈ [0, 1] e construa a RCL em ordem descrescente
       com o peso das arestas pertencentes ao intervalo [cmax + α(cmin - cmax), cmax];
 9     Escolha aleatoriamente uma aresta da RCL. Esta aresta conecta um nó i a outro nó j
       no grafo;
10     Considere S_i = ∅ e S_j = ∅;
11     ∀ v: (v, i) ∈ G and (v, j) ∈ G, i ≠ j;
12     if peso(v, i) < peso(v, j) then
13         S_i ← v;
14     else
15         S_j ← v;
16     end
17     Elimine do grafo G as arestas (v_t, v_z) ∈ V where v_t ∈ S_i e v_z ∈ S_j.
18 end
19 return x*
```

A solução criada ao final da fase Construtiva é transferida para a fase de Busca Local.

3.3.2 Fase de Busca Local

O objetivo da Busca Local é melhorar a solução criada na Fase Construtiva. A estratégia é a mesma utilizada por Nascimento et al. [2010b] e foi inicialmente proposta por Kernighan & Lin [1970]. Para cada nó i é calculada a função objetivo que obtém a distância total de i para cada elemento j pertencente ao grupo k (etapa 8 da Figura 3.2). Na etapa 9, o objeto i é movido para o grupo que fornecer a menor distância total. O algoritmo em questão utiliza a estratégia *best-improvement*, que é uma estratégia onde a solução vizinha escolhida é a melhor dentre todos os vizinhos de uma solução corrente.

O Algoritmo 10, contém os procedimentos realizados durante a Fase Busca Local. A linha 2 representa a condição de parada na Busca Local. Neste trabalho foi considerada a quantidade de iterações. Observações mostraram que 35 repetições da busca local são suficientes para melhorar os resultados da função objetivo. A linha 1 inicia a variável utilizada para contar as iterações da Buscal Local.

A linha 3 contém a estrutura de repetição que verifica para cada objeto o grupo c que fornece a menor distância total entre o objeto i e cada objeto $j \in k$. A linha 5 verifica se o objeto i já está inserido no grupo c, e em caso negativo, o objeto é movido para este grupo.

Algoritmo 10: Fase Busca Local(x)

```
   Data: x ∈ X
   Result: Solução x* ∈ X
 1 it ← 0;
 2 repeat
 3 |   for i = 0 to N − 1 do
 4 |   |   Calcule c: s(i, c)=min_{1≤k≤M} s(i, k) where s(i, k) = Σ_{i=1}^{N} d_ij x_jk;
 5 |   |   if i ∉ G(c) then
 6 |   |   |   G(c) ← i;
 7 |   |   end
 8 |   |   it ← it + 1;
 9 |   end
10 until it > qtdMaxIterações;
11 return x*
```

Após a implementação do GRASP, foram realizados os testes computacionais visando a análise comparativa com os resultados disponibilizados por Nascimento et al. [2010b]. Tais resultados são descritos com detalhes no capítulo 4 e deram suporte para a validação da corretude do algoritmo, o que possibilitou o avanço para a próxima fase - a Implementação do GRASP com *Path-Relinking*.

3.4 Implementação GRASP com *Path-Relinking*

A partir das análises comparativas e validação do GRASP, iniciou-se implementação do GRASP com *Path-Relinking* (GRASP+PR). O GRASP+PR utiliza as mesmas implementações da FaseConstrutiva (Seção 3.3.2) e FaseBuscaLocal (Seção 3.3.1), utilizadas pelo GRASP na sua forma tradicional, e não foram necessárias alterações nessas estruturas. Esta constatação, de certa forma, comprova a modularidade como uma das grandes vantagens das metaheurísticas, fato que reduz o tempo de implementação. Este trabalho considerou a implementação de quatro variações do PR, *Forward*, *Backward*, *Mixed* e *Greedy Randomized Adaptative*.

A incorporação do PR, exigiu modificações na estrutura principal GRASP de forma a possibilitar a chamada do PR durante o processo, bem como incorporar restrições necessárias para o correto funcionamento da estratégia em questão. O Algoritmo 11, ilustra o pseudocódigo da estrutura GRASP+PR utilizada neste trabalho.

Algoritmo 11: GRASP+PR()

```
   Data: I M TP ρ DS Ism DC S
   Result: Solução x* ∈ X
 1 P ← ∅;
 2 while critério de parada não for alcançado do
 3 |   x' ← FaseConstrutiva(·) conforme descrito no Algoritmo 9;
 4 |   if conjuto elite P possuir ao menos ρ elementos then
 5 |   |   x' ← FaseBuscaLocal(x') conforme descrito no Algoritmo 10;
 6 |   |   Aleatoriamente seleciona uma solução y ∈ P;
 7 |   |   x' ← PathRelinking(x', y);
 8 |   |   x' ← FaseBuscaLocal(x') conforme descrito no Algoritmo 10;
 9 |   |   if conjuto elite 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 os elementos com custo
   |   |   |   |   pior que x';
12 |   |   |   end
13 |   |   else if x' ≉ P then
14 |   |   |   P ← P ∪ {x'};
15 |   |   end
16 |   else if x' ≉ P or P = ∅ then
17 |   |   P ← P ∪ {x'};
18 |   end
19 end
20 return x*
```

O algoritmo toma como entrada a instância I, a quantidade M de grupos desejados, o tamanho TP do *pool* de melhores soluções, a quantidade mínima ρ de elementos no pool antes do início do PR, a diferença simétrica DS, a quantidade Ism de iterações sem melhoria, a distância ou correlação DC e a semente S utilizada pelo método aletório. Após inicializar o conjunto elite (também chamado *pool*) como vazio (linha 1), o GRASP+PR é executado nas linhas 2 a 19 até que um critério de parada (Ism) seja satisfeito.

Durante cada iteração, uma solução x' é gerada pela Fase Construtiva na linha 3. Na linha 4 é verificada quantidade de soluções pertencentes a (P). Se P não possuir no mínimo ρ elementos, e se a solução x' for *suficientemente diferente* de todas as soluções presentes em P, então x' é inserido em P (linha 16). Caso P esteja vazio, a solução x' é inserida no conjunto.

O termo *suficientemente diferente*, em uma definição mais precisa, seria o atendimento de uma diferença simétrica mínima δ calculada entre duas soluções. Seja $\Delta(x', x)$, definida como a quantidade mínima de movimentos necessários para transformar a solução x' na solução x, ou vice-versa. Para uma diferença δ, dizemos que x' é suficientemente diferente de todas as soluções elite pertecentes a P, se $\Delta(x', x) > \delta$, para todo $x \in P$, onde indicamos a notação $x' \not\approx P$. Atender a característica suficientemente diferente é essencial para o correto funcionamento do PR que exige diversidade entre as soluções participantes do processo.

Se o conjunto elite P possuir pelo menos ρ elementos, então os passos 5 a 15 são computados. 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 (linhas 6 e 7).

Após a realização do *Path-Relinking*, a solução x' passa novamente pela Fase Busca Local (linha 8) visando melhorias. Caso o conjunto elite esteja completo, é verificado se a solução x' possui qualidade igual ou superior a pior solução do conjunto elite e $x' \not\approx P$. Ao respeitar estes critérios, a solução x' se torna candidata a entrar no conjunto elite sendo substituído desse conjunto a solução x mais similar a x', isto é, a que apresente o menor valor da diferença simétrica $\Delta(x', x)$. Se P não estiver completo a solução x' é inserida caso atenda a restrição $x' \not\approx P$.

Com o GRASP preparada (chamada a partir de agora GRASP+PR), iniciou-se a implementação de cada variante do PR abordada neste trabalho. A descrição das variantes utilizadas será feita a seguir:

3.4.1 *Forward Path-Relinking* e *Backward Path-Relinking*

A implementação do PR para as variantes *Forward* e *Backward*, teve como base o Algoritmo 12 proposto por Resende & Ribeiro [2005], cujos detalhes poderão ser observados no capítulo 2. Esse algoritmo foi adotado para ambas variantes, mas como cada uma considera uma regra diferente para determinar qual solução deverá ser considerada inicial (x_s), na implementação o GRASP recebeu a adição de uma estrutura condicional que verifica o valor das funções objetivo das soluções x (gerada após a busca local) e y (escolhida no *Pool*).

Na variante *Forward* a solução que apresentar o pior valor de função objetivo entre as soluções x e y é considerada como inicio do caminho, ou $x_s = max\{f(x), f(y)\}$, sendo a solução guia a que apresente o melhor valor. Já na variante *Backward*, é considerada como inicio a solução que apresente o melhor valor de função objetivo entre as soluções x e y, ou $x_s = min\{f(x), f(y)\}$, sendo a solução guia a que apresente o pior valor.

O Algoritmo 12 tem como entrada as soluções inicial (x_s) e guia (x_t) obtidas no GRASP+PR respeitando a regra imposta pela variante do PR escolhida.

Algoritmo 12: *Forward* e *Backward PathRelinking*()

Data: Solução inicial x_s e solução guia x_t
Result: Melhor solução x^* no caminho entre x_s e x_t
1 Computar diferença simétrica $\Delta(x_s, x_t)$;
2 $f^* \leftarrow \min\{f(x_s), f(x_t)\}$;
3 $x^* \leftarrow \text{argmin}\{f(x_s), f(x_t)\}$;
4 $x \leftarrow x_s$;
5 **while** $\Delta(x, x_t) \neq \emptyset$ **do**
6 $\quad m^* \leftarrow \text{argmin}\{f(x \oplus m) : m \in \Delta(x, x_t)\}$;
7 $\quad \Delta(x \oplus m^*, x_t) \leftarrow \Delta(x, x_t) \backslash \{m^*\}$;
8 $\quad x \leftarrow x \oplus m^*$;
9 $\quad$ **if** $f(x) < f^*$ **then**
10 $\quad\quad f^* \leftarrow f(x)$;
11 $\quad\quad x^* \leftarrow x$;
12 $\quad$ **end**
13 **end**
14 **return** x^*

Na linha 1, é calculada a diferença simétrica entre as soluções x_s e x_t e neste passo também é construída a lista Γ dos movimentos necessários para que a solução inicial fique igual a solução guia. Γ é classificada na ordem crescente do valor da função objetivo de cada movimento. As linhas 2 e 3 obtém a menor função objetivo e sua respectiva solução, entre as soluções inicial e guia. Esse valor é visto como referência inicial para melhoria do PR.

Enquanto existir movimentos, os passos descritos entre as linhas 5 a 12 são realizados. A cada passo o procedimento examina todos os movimentos $m \in \Delta(x, x_t)$, selecionando aquele que minimiza a função objetivo quando se incorpora m a solução x (linha 6). Na linha 7 o movimento é retirado de Γ e a solução x é atualizada considerando agora o movimento realizado. Caso necessário, a solução x^* é atualizada na linha 11.

3.4.2 *Mixed Path-Relinking*

A implementação do PR para a variante *Mixed*, teve como base o Algoritmo 13 proposto por Resende et al. [2010b]. O PR para essa variante se diferencia das variantes *Forward* e *Backward* com a inclusão das linhas 13 a 15.

Para implementar a estratégia, as regras para as soluções inicial e guia são trocadas a cada passo nas linhas 13 e 14. A linha 15 realiza um novo cálculo da diferença simétrica entre as soluções antes do próximo passo.

Algoritmo 13: *Mixed Path-Relinking*()

Data: Solução inicial x_s e solução guia x_t
Result: Melhor solução x^* no caminho entre x_s e x_t

```
1  Computar diferença simétrica Δ(x_s, x_t);
2  f* ← min{f(x_s), f(x_t)};
3  x* ← argmin{f(x_s), f(x_t)};
4  x ← x_s;
5  while Δ(x, x_t) ≠ ∅ do
6  |   m* ← argmin{f(x ⊕ m) : m ∈ Δ(x, x_t)};
7  |   Δ(x ⊕ m*, x_t) ← Δ(x, x_t)\{m*};
8  |   x ← x ⊕ m*;
9  |   if f(x) < f* then
10 |   |   f* ← f(x);
11 |   |   x* ← x;
12 |   end
13 |   x_s ← x_t;
14 |   x_t ← x;
15 |   Computar diferença simétrica Δ(x_s, x_t);
16 end
17 return x*
```

Grande parte do escopo do algoritmo *Mixed Path-Relinking* é utilizado na variante *Greedy Randomized Adaptative*, a qual será descrita a seguir.

3.4.3 *Greedy Randomized Adaptative*

A implementação do PR para a variante *Greedy Randomized Adaptative*, teve como base o Algoritmo 14, proposto por Binato et al. [2002]. A principal diferença com relação ao Algoritmo 13 são as linhas 5 e 7-10.

Ao invés de selecionar o movimento que resulta na melhor solução, como é o caso do *Path-Relinking* padrão, uma lista restrita de candidatos (RCL - *Restricted Candidate List*) é construída com os movimentos que resultam nos custos situados em um intervalo que depende do valor do melhor movimento (linha 7), pior movimento (linha 8) e um parâmetro aleatório α (linha 5). A partir da RCL, um movimento é selecionado aleatoriamente para produzir o próximo passo no caminho.

Algoritmo 14: *Greedy Randomized Adaptative Path-Relinking*()

```
   Data: Solução inicial x_s e solução guia x_t
   Result: Melhor solução x* no caminho entre x_s e x_t
 1 Computar diferença simétrica Δ(x_s, x_t);
 2 f* ← min{f(x_s), f(x_t)};
 3 x* ← argmin{f(x_s), f(x_t)};
 4 x ← x_s;
 5 Selecionar α ∈ [0,1] ⊂ ℝ aleatoriamente;
 6 while Δ(x, x_t) ≠ ∅ do
 7 |   f⁻ ← min{f(x ⊕ m) : m ∈ Δ(x, x_t)};
 8 |   f⁺ ← max{f(x ⊕ m) : m ∈ Δ(x, x_t)};
 9 |   RCL_pr ← {m ∈ Δ : f(x ⊕ m) ≤ f⁻ + α(f⁺ − f⁻)};
10 |   Selecionar m* ∈ RCL_pr aleatoriamente;
11 |   Δ(x ⊕ m*, x_t) ← Δ(x, x_t)\{m*};
12 |   x ← x ⊕ m*;
13 |   if f(x) < f* then
14 |   |   f* ← f(x);
15 |   |   x* ← x;
16 |   end
17 |   x_s ← x_t;
18 |   x_t ← x;
19 |   Computar diferença simétrica Δ(x_s, x_t);
20 end
21 return x*
```

A seguir são descritos os detalhes do ambiente de desenvolvimento, testes preliminares e parâmetros utilizados nos experimentos.

3.5 Ambiente de Desenvolvimento e Experimentos Computacionais

Nessa seção serão apresentados os aspectos relacionados ao ambiente utilizado para desenvolvimento e experimentos computacionais realizados sobre as metaheurísticas propostas. Primeiro será descrito o ambiente utilizado para o desenvolvimento do trabalho e realização dos testes computacionais incluindo equipamento, sistema operacional, liguagem e bibliotecas utilizadas. Em seguida, serão descritos os parâmetros de

configuração como entrada em cada metaheurística, bem como os critérios de parada das estruturas de repetição mais relevantes.

3.5.1 Ambiente

Todo desenvolvimento e experimentos foram realizados em um notebook Dell com processador Intel Core 2 Duo modelo T8100 de 2.1GHz, 3GB de memória, sistema operacional Windows XP Professional versão 5.1 2002 SP3 executado em x86.

As metaheurísticas GRASP e todas as variantes do *Path-Relinking* foram implementadas em Java HotSpot(TM) Client VM 16.3-b01 e compilada em bytecode com javac versão 1.6.0.20. As implementações foram realizadas no ambiente NetBeans versão 6.9.1 (Build 201011082200).

O gerador de números aleatórios é uma implementação do algoritmo Mersenne Twister [Matsumoto & Nishimura, 1998] da biblioteca COLT[5].

Os algoritmos k-means e k-medians, utilizados por Nascimento et al. [2010b], estão disponíveis em (http://bonsai.hgc.jp/∼mdehoon/software/cluster), e PAM no pacote de agrupamento do R-project (http://www.r-project.org/).

3.5.2 Parâmetros

Na definição dos parâmetros o principal objetivo foi encontrar os melhores índices CRand quando comparados com os da literatura. Em seguida o GRASP e variantes foram ajustados de forma a se obter os menores tempos sem que isso comprometesse os melhores resultados CRand já encontrados.

Cada instância foi analisada separadamente e os parâmetros obtidos são um reflexo das características da própria instância como quantidade de objetos e atributos. Foi observado que instâncias com poucos objetos (menos que 100) e poucos agrupamentos (menos que 4) apresentam uma menor diferença simétrica entre as soluções. Um valor alto para diferença simétrica reflete em uma maior rigidez na inserção das soluções no conjunto elite, já para um valor baixo temos uma condição mais relaxada e a solução é inserida com mais facilidade. Em ambas as situações, temos o comprometimento da diversidade das soluções. Nesse sentido os parâmetros considerados procuraram ser coerentes com relação a inserção das soluções no conjunto elite. A Tabela 3.2 contém os parâmetros utilizados e que refletiram nos resultados descritos no capítulo 4.

A semente (S) de todos os algoritmos foi passada como parâmetro sendo o número 191177 o valor inicial definido para cada instância. A semente foi incrementada em uma unidade a cada repetição do algoritmo.

O critério de parada utilizado pelo GRASP e GRASP+PR foi a quantidade máxima de iterações sem melhorias para função objetivo. O contador é reiniciado sempre que encontrar uma solução melhor que a solução até então encontrada.

Inicialmente o critério de parada considerado na Busca Local foi de 50 iterações, mesmo valor utilizado por Nascimento [2010]. Visando a redução do tempo computaci-

[5]COLT é uma biblioteca de software para aplicações científicas e computacionais de alto desempenho em Java (http://acs.lbl.gov/∼hoschek/colt/)

Tabela 3.2. Os parâmetros utilizados no *Path-Relinking*

	TP	ρ	DS	Ism
Iris	3	1	4	15
Novartis	5	3	70	15
BreastA	4	1	4	15
BreastB1	3	1	30	15
BreastB2	3	1	30	15
DLBCLA	5	2	100	15
DLBCLB	5	2	100	15
MultiA	5	2	70	15
Breast1	3	1	4	15
Breast2	6	3	550	15
Protein1	5	2	450	15
Protein2	5	3	450	15
Yeast	7	3	1200	5

LEGENDA

TP	Tamanho do Pool de Soluções Elite.
ρ	Quantidade mínima de elementos no Pool antes de iniciar o PR.
DS	Diferença Simétrica.
Ism	Iterações sem melhoria.

onal, esse parâmetro foi reduzido para 35 iterações e mostrou-se suficiente para que os melhores CRands fossem mantidos. A busca local com o critério de parada em questão fornece como resultado um ótimo local aproximado.

Para cada algoritmo, realizou-se experimentos computacionais com as 10 instâncias (Seção 3.1) e as 4 distâncias/correlações descritas na Seção 2.1.2.1. Para cada experimento foram coletados dados da função objetivo, CRand e tempo. Enquanto os algoritmos de Nascimento et al. [2010b] GRASP-L, k-means, k-medians e PAM foram repetidos 100 vezes, as cinco versões propostas neste trabalho foram repetidos 30 vezes. A execução dos algoritmos considerou a mesma quantidade de grupos descritas pelos autores para o algoritmo GRASP-L.

No capítulo 4, serão apresentados os resultados computacionais dos modelos híbridos propostos.

Capítulo 4

Resultados e Discussão

Nesse capítulo, serão apresentados os resultados dos experimentos computacionais das cinco variantes propostas nesse trabalho, aqui chamadas GRASP-L, GRASP, GRASP+PRf, GRASP+PRb, GRASP+PRm and GRASP+PRgr, representando, respectivamente, o algoritmo de Nascimento et al. [2010b], o GRASP tradicional e as quatro variações *forward*, *backward*, *mixed* e *greedy randomized* do *Path-Relinking*.

As análises visaram verificar a robustez e eficiência dos algoritmos propostos. Por apresentarem uma componente aleatória, execuções diferentes na heurística podem resultar em soluções diferentes. A robustez analisa se um algoritmo, em suas diversas execuções, apresenta resultados similares. A análise da eficiência foi feita sobre dois critérios, onde no primeiro procurou-se analisar o valor da função objetivo e seu tempo computacional, e o segundo utilizou a medida CRand que avalia a qualidade das soluções geradas pelo algoritmo em comparação com as classificações reais fornecidas com os conjuntos de dados.

Para validar a significância entre os resultados obtidos pelos diferentes algoritmos, foi utilizado *Wilcoxon Paired Signed Rank Test*, que é um teste não paramétrico que testa a diferença média entre pares de dados [Wilcoxon, 1945].

Os experimentos foram divididos em duas partes: A primeira fez uma comparação dos resultados descritos em Nascimento et al. [2010b] com os cinco algoritmos propostos nesse trabalho. Nesse experimento, a análise comparativa considerou apenas o CRand, que foi a medida disponibilizada pelos autores. O segundo experimento, teve o propósito de realizar uma análise comparativa entre o GRASP e todas as variantes do *Path-Relinking* e nesse caso, foi considerando o CRand, a função objetivo e o tempo médio de execução de cada algoritmo. A seguir são detalhados os experimentos e seus respectivos resultados.

4.1 Comparação com os resultados da literatura

Nesse experimento, são apresentados os valores CRand para os algoritmos PAM, k-means, k-medians, GRASP-L, GRASP, GRASP+PRf, GRASP+PRb, GRASP+PRm, GRASP+PRgr, para todas as instâncias descritas no capítulo 3. Foram consideradas as medidas Euclidiana e *City Block*, e as correlações Cosine e Pearson. Nas tabelas a

seguir, a coluna instância indica a base de dados utilizada, e as colunas M e CRand, mostram respectivamente o número de grupos para o melhor CRand encontrado. Os resultados CRand foram analisados com o teste *Wilcoxon paired signed rank* [Wilcoxon, 1945]. Na Tabela 4.1 são descritos os valores CRand para as distâncias Euclidiana e *City Block*. Os melhores valores CRand estão destacados em negrito.

Para a distância **Euclidiana**, a implementação do GRASP e variações do PR obtiveram bons resultados quando comparados com os demais algoritmos. A variação *Backward* (GRASP+PRb) obteve 8 instâncias com melhores CRands. As variações *Greedy Randomized*, *Mixed* e *Forward* apresentaram 7 melhores CRands. As melhorias mais significativas ocorreram sobre a instância DLBCLA, com aumento de 19% no valor do CRand utilizando o GRASP+PRb. A instância DLBCLB obteve 10.4% de melhoria no CRand com o GRASP+PRb e a instância BreastA com 5.6% com o GRASP+PRm.

Para a distância ***City Block***, a variação GRASP+PRm apresentou a maior quantidade de melhores CRand's (total de 7), seguida das variações GRASP+PRg, GRASP+PRb, GRASP+PRf com 6 melhores CRand cada. A instância BreastB (estrutura 2) foi a que obteve melhorias mais significativas, aumentando o CRand em 56,7% (GRASP) com relação ao valor obtido por Nascimento et al. [2010b], e 10% com relação ao melhor CRand até então encontrado pelo k-means. Ainda na BreastB (estrutura 1), apesar do k-means apresentar o maior CRand (0.563), houve uma melhoria de 37.7% do GRASP com relação ao GRASP-L. Mesmo não apresentando um aumento significativo, o algoritmo GRASP+PRm para a instância MultiA, foi o único que melhorou o CRand, cujo valor, é constante para os algoritmos GRASP-L, GRASP e demais variações do PR.

Na Tabela 4.2 são descritos os valores CRand para as correlações Cosine e Pearson. Para a **Cosine**, nota-se uma sensível queda na quantidade de melhores CRands encontradas por cada algoritmo quando comparadas com as métricas de distância. Os algoritmos GRASP+PRgr e GRASP+PRf obtiveram 6 melhores CRand, seguido dos algoritmos GRASP+PRm e GRASP+PRb com 5 melhores, e os algoritmos GRASP e k-medians com 4. As instâncias mais favorecidas foram a BreastB com um aumento de 9,4%, e a MultiA com 3.1% no valor do CRand. Para ambas as instâncias, o valor do CRand foi o mesmo para o GRASP e demais variações do PR.

Para a **Pearson**, os algoritmos GRASP+PRgr, GRASP+PRm e GRASP obtiveram 5 melhores CRand, seguido dos algoritmos GRASP+PRb, GRASP+PRf e k-medians com 4 melhores CRands. A instância BreastB obteve melhoria de 9,4% com os algoritmos GRASP, GRASP+PRf, GRASP+PRm e GRASP+PRgr, e a instância Breast 8.7% de melhoria considerando o GRASP e todas as variações do PR. Para as demais instâncias, os resultados obtidos foram muito similares.

Na Tabela 4.3, estão descritos os melhores resultados CRand para cada algoritmo, independente da medida de dissimilaridade utilizada. Os algoritmos GRASP+PRgr, GRASP+PRm e GRASP+PRf apresentaram 7 maiores CRands; GRASP+PRb e GRASP, 6; k-means e k-medians, 1 e finalmente, GRASP-L e PAM não obtiveram nenhum CRand maior que o dos demais.

Tabela 4.1. Resultados CRand para as distâncias **Euclidiana** e **City Block** como medidas de dissimilaridade.

EUCLIDIANA																				
	GRASP+PRgr		**GRASP+PRm**		**GRASP+PRb**		**GRASP+PRf**		**GRASP**		**GRASP-L**		**KMEANS**		**KMEDIANS**		**PAM**	
Instância	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**
Protein	4	0.297	4	0.297	4	0.294	4	0.294	4	0.294	4	**0.322**	7	**0.322**	7	0.313	6	0.250
	11	**0.169**	11	0.168	11	0.167	11	**0.169**	11	0.168	11	0.168	17	0.139	25	0.134	13	0.098
Breast	2	**0.878**	2	**0.878**	2	**0.878**	2	**0.878**	2	**0.878**	2	0.877	2	0.803	2	0.782	2	0.828
	15	0.016	15	0.016	15	0.014	15	0.016	15	0.017	15	0.015	18	-0.010	17	**0.036**	5	0.012
Yeast	9	0.151	9	0.153	9	0.153	9	0.150	9	0.151	9	0.150	7	0.170	8	**0.173**	8	0.143
Novartis	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	0.921	4	0.946	4	0.946	4	0.897
BreastA	2	0.682	2	**0.723**	2	0.682	2	0.682	2	0.682	2	0.682	2	0.654	2	0.654	2	0.543
BreastB	2	**0.694**	2	**0.694**	2	**0.694**	2	**0.694**	2	**0.694**	2	0.626	3	0.502	4	0.500	2	0.388
	2	**0.322**	2	**0.322**	2	**0.322**	2	**0.322**	2	0.321	2	0.314	3	0.286	3	0.260	2	0.187
DLBCLA	4	0.465	4	0.431	4	**0.504**	4	0.408	4	0.443	4	0.408	4	0.309	5	0.365	4	0.276
DLBCLB	4	0.520	4	0.519	4	**0.537**	4	0.509	4	0.519	4	0.481	2	0.420	3	0.424	3	0.391
MultiA	4	**0.874**	4	**0.874**	4	**0.874**	4	**0.874**	4	**0.874**	4	**0.874**	6	0.765	5	0.682	4	0.765
Iris	3	**0.757**	3	**0.757**	3	**0.757**	3	**0.757**	3	**0.757**	3	0.756	3	0.730	3	0.744	3	0.730
CITY BLOCK																				
	GRASP+PRgr		**GRASP+PRm**		**GRASP+PRb**		**GRASP+PRf**		**GRASP**		**GRASP-L**		**KMEANS**		**KMEDIANS**		**PAM**	
Instância	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**
Protein	5	**0.310**	5	**0.310**	5	0.309	5	0.309	5	0.309	5	0.293	8	0.223	7	0.229	3	0.192
	9	0.179	9	0.176	9	**0.180**	9	**0.180**	9	0.175	9	0.166	17	0.158	28	0.141	19	0.084
Breast	2	**0.877**	2	**0.877**	2	**0.877**	2	**0.877**	2	**0.877**	2	**0.877**	2	0.770	2	0.765	2	0.807
	19	0.016	19	0.015	19	0.015	19	0.015	19	0.016	19	0.013	19	-0.009	10	**0.023**	13	0.010
Yeast	7	0.161	7	0.159	7	0.160	7	0.161	7	0.161	7	0.157	7	**0.181**	6	0.167	7	0.152
Novartis	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	0.921	4	0.946	4	0.921	4	0.947
BreastA	2	**0.723**	2	**0.723**	2	0.682	2	**0.723**	2	0.722	2	0.682	2	0.583	2	0.618	4	0.560
BreastB	4	0.366	4	0.366	4	0.261	4	0.288	4	0.328	4	0.228	3	**0.563**	2	0.561	2	0.388
	7	0.333	7	0.344	7	0.244	7	0.328	7	**0.367**	7	0.159	3	0.328	3	0.284	2	0.187
DLBCLA	3	**0.838**	3	**0.838**	3	**0.838**	3	**0.838**	3	**0.838**	3	0.800	3	0.805	3	0.784	3	0.406
DLBCLB	2	0.701	2	0.701	2	**0.703**	2	0.701	2	0.701	2	0.700	2	0.690	2	0.690	3	0.350
MultiA	4	0.899	4	**0.924**	4	0.899	4	0.899	4	0.899	4	0.899	4	0.851	4	0.875	5	0.750
Iris	3	**0.818**	3	**0.818**	3	**0.818**	3	**0.818**	3	**0.818**	3	**0.818**	3	0.717	3	0.717	3	0.772

Tabela 4.2. Resultados CRand para as correlações **Cosine** e **Pearson** como medidas de dissimilaridade.

COSINE																		
	GRASP+PRgr		**GRASP+PRm**		**GRASP+PRb**		**GRASP+PRf**		**GRASP**		**GRASP-L**		**KMEANS**		**KMEDIANS**		**PAM**	
Instância	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**
Protein	4	**0.350**	4	0.348	4	0.344	4	0.342	4	0.348	4	0.349	7	0.320	6	0.304	6	0.247
	12	0.170	12	0.170	12	0.170	12	**0.173**	12	0.170	12	0.166	20	0.134	21	0.125	15	0.091
Breast	3	0.294	3	0.294	3	0.294	3	0.294	3	0.294	3	0.293	4	0.258	3	0.306	3	**0.332**
	8	0.022	8	0.021	8	0.021	8	0.022	8	0.022	8	0.020	2	0.027	8	**0.052**	3	0.021
Yeast	9	0.137	9	0.137	9	0.137	9	0.137	9	0.136	9	0.135	9	0.138	6	0.132	7	**0.146**
Novartis	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	0.920	4	0.919	4	0.919	4	0.745
BreastA	2	0.687	2	0.687	2	0.687	2	0.687	2	0.687	2	0.686	2	**0.691**	2	**0.691**	2	0.664
BreastB	2	**0.694**	2	**0.694**	2	**0.694**	2	**0.694**	2	**0.694**	2	0.626	2	0.561	3	0.502	4	0.443
	2	**0.322**	2	**0.322**	2	**0.322**	2	**0.322**	2	0.321	2	0.314	2	0.269	3	0.264	4	0.239
DLBCLA	4	0.607	4	0.619	4	0.607	4	0.607	4	0.607	4	0.605	5	0.642	4	**0.678**	3	0.547
DLBCLB	4	0.500	4	0.500	4	0.500	4	0.500	4	0.500	4	0.502	3	0.501	3	**0.623**	5	0.385
MultiA	4	**0.831**	4	**0.831**	4	**0.831**	4	**0.831**	4	**0.831**	4	0.805	4	0.718	7	0.731	6	0.716
Iris	3	**0.942**	3	**0.942**	3	**0.942**	3	**0.942**	3	**0.942**	3	0.941	3	0.904	3	0.941	3	0.904
PEARSON																		
	GRASP+PRgr		**GRASP+PRm**		**GRASP+PRb**		**GRASP+PRf**		**GRASP**		**GRASP-L**		**KMEANS**		**KMEDIANS**		**PAM**	
Instância	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**
Protein	4	**0.345**	4	**0.345**	4	**0.345**	4	**0.345**	4	**0.345**	4	0.344	7	0.313	7	0.306	6	0.245
	12	**0.172**	12	**0.172**	12	**0.172**	12	0.169	12	**0.172**	12	0.167	20	0.129	27	0.136	14	0.096
Breast	3	0.311	3	0.311	3	0.311	3	0.311	3	0.311	3	0.284	2	**0.441**	2	0.368	2	0.289
	11	0.017	11	0.016	11	0.017	11	0.016	11	0.016	11	0.017	9	0.015	19	**0.024**	6	0.015
Yeast	9	0.138	9	0.138	9	0.138	9	0.138	9	0.138	9	0.131	8	0.135	8	0.133	7	**0.145**
Novartis	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	0.920	4	0.919	4	0.919	4	0.746
BreastA	2	0.692	2	0.692	2	0.692	2	0.692	2	0.692	2	0.692	2	**0.705**	2	**0.705**	2	0.635
BreastB	2	**0.766**	2	**0.766**	2	0.626	2	**0.766**	2	**0.766**	2	0.694	3	0.502	3	0.529	3	0.445
	2	0.322	2	0.322	2	0.306	2	0.281	2	0.279	2	**0.355**	4	0.289	3	0.283	3	0.227
DLBCLA	4	0.604	4	0.604	4	0.607	4	0.604	4	0.604	4	0.585	4	0.605	4	**0.684**	4	0.586
DLBCLB	2	0.585	2	0.585	2	0.585	2	0.585	2	0.585	2	0.527	3	**0.665**	3	0.561	3	0.545
MultiA	4	**0.829**	4	**0.829**	4	**0.829**	4	**0.829**	4	**0.829**	4	0.828	4	0.718	9	0.691	4	0.705
Iris	3	0.886	3	0.886	3	0.886	3	0.886	3	0.886	3	0.886	3	0.886	3	**0.941**	3	0.886

Tabela 4.3. Melhores CRand obtidos entre as quatro medidas de dissimilaridade

	GRASP+PRgr		GRASP+PRm		GRASP+PRb		GRASP+PRf		GRASP		GRASP-L		KMEANS		KMEDIANS		PAM	
Instância	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**	**M**	**CRand**
Protein	4	**0.350**	4	0.348	4	0.345	4	0.345	4	0.348	4	0.349	7	0.322	7	0.313	6	0.250
	9	0.179	9	0.176	9	**0.180**	9	**0.180**	11	0.175	11	0.168	17	0.158	28	0.141	13	0.098
Breast	2	**0.878**	2	**0.878**	2	**0.878**	2	**0.878**	2	**0.878**	2	0.877	2	0.803	2	0.782	2	0.828
	8	0.022	8	0.021	8	0.021	8	0.022	8	0.022	8	0.020	2	0.027	8	**0.052**	3	0.021
Yeast	7	0.161	7	0.159	7	0.160	7	0.161	7	0.161	7	0.157	7	**0.181**	6	0.173	7	0.152
Novartis	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	**0.950**	4	0.921	4	0.946	4	0.946	4	0.947
BreastA	2	**0.723**	2	**0.723**	2	0.692	2	**0.723**	2	0.722	2	0.692	2	0.705	2	0.705	2	0.664
BreastB	2	**0.766**	2	**0.766**	2	0.694	2	**0.766**	2	**0.766**	2	0.694	3	0.563	2	0.561	3	0.445
	2	0.333	2	0.344	2	0.322	2	0.328	2	**0.367**	2	0.355	3	0.328	3	0.284	4	0.239
DLBCLA	3	**0.838**	3	**0.838**	3	**0.838**	3	**0.838**	3	**0.838**	3	0.800	3	0.805	3	0.784	4	0.586
DLBCLB	2	0.701	2	0.701	2	**0.703**	2	0.701	2	0.701	2	0.700	2	0.690	2	0.690	3	0.545
MultiA	4	0.899	4	**0.924**	4	0.899	4	0.899	4	0.899	4	0.899	4	0.851	4	0.875	4	0.765
Iris	3	**0.942**	3	**0.942**	3	**0.942**	3	**0.942**	3	**0.942**	3	0.941	3	0.904	3	0.941	3	0.904

As Tabelas 4.4 e 4.5 descrevem os ρ-values do teste Wilcoxon para as distâncias euclidiana e *City Block* e para as correlações Cosine e Pearson. Os valores ρ são classificados como Altamente Significativos quando $\rho < 0.01$, Significativos quando $0.01 \leq \rho \leq 0.05$ e Não Significativos quando $\rho > 0.05$.

Tabela 4.4. ρ-values do teste *Wilcoxon paired signed rank* para as distâncias **Euclidiana** e **City Block**

	EUCLIDIANA				CITY BLOCK			
	GRASP-L	**KMEANS**	**KMEDIANS**	**PAM**	**GRASP-L**	**KMEANS**	**KMEDIANS**	**PAM**
GRASP	0.02734	0.00610	0.01709	0.00024	0.00195	0.04785	0.03979	0.00244
GRASP+PRf	0.07422	0.00610	0.01709	0.00024	0.00195	0.06396	0.03979	0.00464
GRASP+PRb	0.05371	0.00806	0.01709	0.00024	0.00391	0.14648	0.12720	0.01050
GRASP+PRm	0.01855	0.00244	0.01709	0.00024	0.00098	0.05737	0.03979	0.00171
GRASP+PRgr	0.01855	0.00244	0.01709	0.00024	0.00195	0.04785	0.03979	0.00171

Tabela 4.5. ρ-values do teste *Wilcoxon paired signed rank* para as correlações **Cosine** e **Pearson**

	COSINE				PEARSON			
	GRASP-L	**KMEANS**	**KMEDIANS**	**PAM**	**GRASP-L**	**KMEANS**	**KMEDIANS**	**PAM**
GRASP	0.01343	0.04785	0.41431	0.00464	0.08301	0.56934	0.63550	0.00146
GRASP+PRf	0.02661	0.04785	0.41431	0.00464	0.08301	0.56934	0.63550	0.00146
GRASP+PRb	0.03271	0.04785	0.45483	0.00488	0.37500	0.30127	0.49731	0.00146
GRASP+PRm	0.01050	0.03271	0.45483	0.00488	0.05371	0.33936	0.49731	0.00146
GRASP+PRgr	0.00464	0.04785	0.41431	0.00464	0.04883	0.30127	0.49731	0.00146

É possível observar a vantagem das medidas de distância sobre as correlações quanto a contribuição dos melhores resultados. Para as medidas é observado que 63% dos resultados possuem uma diferença altamente significativa em comparação com os resultados da literatura, sendo que nas correlações este índice é de 33%. As correlações detêm 76% dos resultados não significativos, sendo que nas distâncias este índice é de 24%.

Um dos motivos dessa desvantagem pode ser explicado pelo fato das correlações serem medidas que priorizam o intervalo de dados e acabam por automaticamente desconsiderar diferenças entre as variáveis, como por exemplo diferenças de escala [Borgatti, 2011]. Nesse sentido, mesmo que a correlação considere dois objetos similares, o que em um primeiro momento os classifica no mesmo grupo, o fato de não se considerar a diferença de escala e a real distância entre eles pode comprometer a coerência do agrupamento.

O gráfico 4.1 ilustra a quantidade de melhores CRands por métrica onde é possível observar que o algoritmo GRASP+PRb obteve a maior quantidade de melhores resultados com a métrica Euclidiana. É interessante notar que os algoritmos que utilizam abordagem clássica (k-means, k-medians e PAM), obtiveram melhores resultados com as correlações. A justificativa pode estar baseada no fato das correlações possuirem característica de normalização dos dados o que de certa forma contribui para valores menos dispersos da mediana (para o k-medians e PAM) e erro quadrático (k-means), já que nas correlações a distância entre os objetos estão restritos ao intervalo [-1, 1].

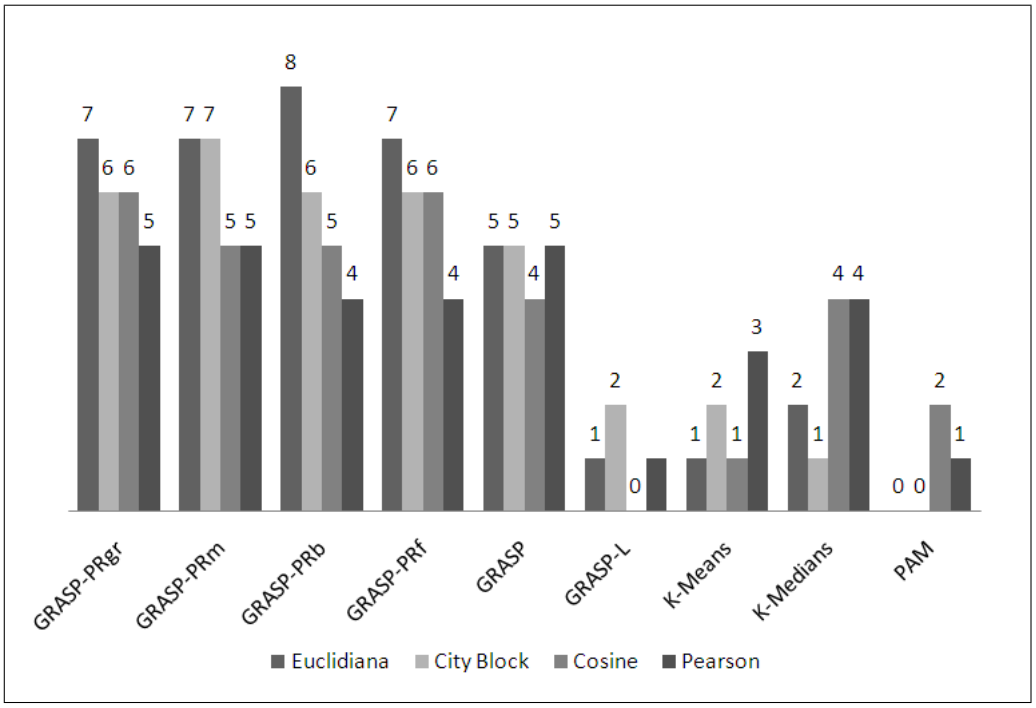

Figura 4.1. Quantidade de melhores CRands para cada métrica e algoritmo.

4.2 Comparação entre os algoritmos propostos

Esse experimento tem o propósito de realizar uma comparação de eficiência e robutez dos algoritmos propostos. Foram analisados os resultados obtidos para o CRand, Função Objetivo (FO) e Tempo (em segundos) para cada algoritmo, métrica e instância. Cada instância foi analisada individualmente e para cada métrica foram descritos os valores mínimo, máximo, média e desvio padrão (DP) obtidos para o CRand e Função Objetivo.

A análise da eficiência foi realizada de duas maneiras, a primeira considerando o menor tempo médio obtido por cada algoritmo, e a segunda com a utlização do gráfico Time-to-target (TTTplot) [Aiex et al., 2007]. O TTTplot ilustra no eixo das ordenadas a probabilidade de um algoritmo encontrar uma solução pelo menos tão boa quanto um determinado valor alvo (ex. função objetivo) em um certo tempo de execução, ilustrado no eixo das abscissas. TTTplot foi utilizado por Feo et al. [1994] e defendido por Hoos & Stützle [1998b,a] como forma de caracterizar os tempos de execução de algoritmos estocásticos para otimização combinatória.

Já para análise da robustez, foi considerado o menor DP obtido para as funções objetivo. Quanto menor o DP, menor a variabilidade das soluções encontradas.

Na Tabela 4.6 estão descritos os valores da instância **Breast1**. Os resultados mostram que todos os algoritmos apresentaram os mesmos valores CRand e função objetivo. O desvio padrão, tanto para os CRand's quanto para a função objetivo, mostra que não existe variabilidade de soluções. Esta informação é justificada pelo fato da instância em questão possuir poucos grupos e tender a um comportamento determinístico.

Para casos onde a quantidade de grupos é pequena (geralmente entre 2 a 4 grupos), a Busca Local se mostra eficiente para convergência do menor valor da função objetivo. Se por um lado a pouca variabilidade indica sinais de robustez, a pouca diversidade de soluções após a busca local elimina a necessidade do *Path-Relinking*, já que tal estratégia exige uma diferença simétrica mínima entre as soluções participantes. Esse comportamento foi observado em várias situações cujas instâncias apresentam as características supracitadas, corroborando [Ribeiro & Resende, 2010].

Com relação a eficiência, as distâncias apresentaram um tempo médio 50% menor quando comparadas as correlações. O algoritmo GRASP+PRgr obteve o menor tempo médio para as métricas euclidiana, *City Block* e Pearson, sendo que o algoritmo GRASP+PRb obteve o menor tempo médio na correlação Cosine.

A Tabela 4.7 contém os resultados da instância **Breast2**. Diferente da Breast1, a Breast2 apresentou uma maior diversidade de resultados. O valor da função objetivo obteve o melhor resultado com a métrica euclidiana e o GRASP+PRm. Para as correlações Cosine e Pearson, o GRASP+PRb mostrou-se mais robusto ao apresentar um menor desvio padrão e uma ligeira queda na função objetivo.

Analisando a eficiência para a instância Brest2, as metricas euclidiana e *City Block* apresentaram os maiores tempos, fato justificado pela maior quantidade de grupos. Os algoritmos que apresentaram os menores tempos médios foram GRASP+PRb (euclidiana), GRASP+PRg (*City Block*), GRASP+PRm (Cosine) e GRASP+PRf (Pearson). Embora o *Path-Relinking* tenha a tendência de aumentar o custo computacional quando comparado ao GRASP básico, este fato não foi observado na instância em questão. O provável motivo é a capacidade do PR de aumentar a velocidade de convergência do GRASP.

Os resultados obtidos para instância **BreastA**, estão descritos na Tabela 4.8. Para esta instância o CRand para as quatro métricas apresentou baixa variabilidade, mas o mesmo não ocorreu para os resultados da função objetivo. Este resultado mostra que valores diferentes de função objetivo podem apresentar o mesmo CRand. Na métrica euclidiana mesmo que os algoritmos com *Path-Relinking* não tenham contribuido para redução da FO, os algoritmos GRASP+PRm e GRASP+PRf possibilitaram uma maior robustez ao apresentarrem uma redução no DP. Na *City Block* o algoritmo GRASP+PRf foi o único que obteve uma redução na função objetivo. Na Cosine, o algoritmo GRASP+PRf obteve uma redução no DP e na correlação Pearson todos os algoritmos obtiveram o mesmo valor da função objetivo.

Com relação aos tempos médios de execução, os algoritmos híbridos apresentaram redução quando comparados aos GRASP ficando os melhores tempos para o algoritmo GRASP+PRb nas métricas euclidiana, *City Block* e Cosine, e o algoritmo GRASP+PRm na Pearson.

A Tabela 4.9 ilustra os resultados para a instância **BreastB1**. Esta instância apresentou comportamento similar as instâncias com pequena quantidade de grupos. Nesse sentido as métricas euclidiana, Cosine e Pearson apresentaram valores semelhantes em se tratando da função objetivo. Para essas métricas a maior contribuição dos algoritmos híbridos foi uma ligeira redução nos tempos médios quando comparados ao GRASP. Devido a quantidade de grupos, a métrica *City Block* apresentou uma maior variabilidade nos valores CRand e Função Objetivo. O algoritmo GRASP+PRf foi o

único a apresentar uma redução no valor da FO.

A Tabela 4.10 apresenta os resultados da instância **BreastB2**. Assim como na BreastB1, devido à baixa quantidade de grupos, os valores da função objetivo são iguais para todos os algoritmos das métricas euclidiana, Cosine e Pearson. Nessas métricas, o GRASP+PRb foi o algoritmo que apresentou os maiores valores de DP, os menores tempos e se manteve entre os melhores CRands nas métricas euclidiana e Cosine.

Já na métrica *City Block* foi observada uma maior diversidade de DPs, fato proporcionado pela maior quantidade de grupos. Nesta métrica, o algoritmo GRASP+PRm foi o único a possibilitar a redução da Função Objetivo. Com relação aos tempos, os algoritmos híbridos obtiveram melhor desempenho quando comparados ao GRASP.

Os resultados da instância **DLBCLA** estão descritos na Tabela 4.11. Na métrica euclidiana o algoritmo GRASP+PRb foi o único que apresentou um valor diferente da função objetivo em comparação aos demais algoritmos. Interessante notar que mesmo tendo um valor da FO e DP maior que dos demais algoritmos, o algoritmo GRASP+PRb obteve o maior CRand, a maior média CRand e o menor tempo médio. Esperava-se que a redução da FO repercutisse de forma direta no aumento do CRand, o que não aconteceu. O mesmo comportamento foi observado na correlação Pearson onde, mesmo apresentando uma FO e um DP maior que a dos demais algoritmos, o GRASP+PRb obteve o maior CRand e um menor tempo médio. Um provável motivo para este comportamento, é que para algumas instâncias a estratégia mono-objetivo não é suficiente para uma modelagem coerente do problema. Handl & Knowles [2005] e Handl & Knowles [2007] propuseram uma abordagem multi-objetivo para o problema de agrupamento de dados e obtiveram resultados superiores com relação a abordagem mono-objetivo.

Ainda para a instância DLBCLA, na correlação Cosine, embora o valor da FO tenha sido o mesmo para todos os algoritmos, a estratégia GRASP+PRm foi o única a apresentar o maior CRand. Essa constatação acaba por corroborar com comportamento supracitado, ou seja, em alguns casos não existe uma relação na diminuição da função objetivo, com o aumento do CRand.

A Tabela 4.12 descreve os resultados da instância **DLBCLB**. Na métrica euclidiana, é possível observar o mesmo comportamento que o constatado na instância DLBCLA. O algoritmo GRASP+PRb apresentou o valor mais alto da FO e DP, mas em contrapartida, apresentou o melhor CRand. Na métrica *City Block*, os valores FO foram os mesmos encontrados para todos os algoritmos mas a estratégia GRASP+PRb obteve uma pequena melhora no CRand e um menor tempo médio. Nas correlações, foi observado que todos os algoritmos alcançaram os mesmos valores da FO. O algoritmo GRASP+PRb obteve os melhores tempos médios em todas as quatro métricas.

A instância **Iris**, cujos resultados estão descritos na Tabela 4.13 não apresentou diferenças significativas para o CRand e FO. O principal motivo é a pequena quantidade de grupos (3), objetos (150) e atributos (4) o que repercute em um comportamento determinístico na tarefa de agrupamento. A correlação Pearson, foi a que apresentou os piores tempos médios.

As instâncias **MultiA** (Tabela 4.14) e **Novartis** (Tabela 4.15), apresentaram comportamentos similares, talvez porque suas bases possuem a mesma origem. A

instância Novartis não apresentou variabilidade nos CRand's e FOs, fato justificado pelo pré-processamento (normalização) realizada nesta base.

Com relação à instância MultiA, a métrica *City Block* foi a única a apresentar um desvio padrão em todos os algoritmos, e também foi a única a apresentar melhoria no CRand (obtido pelo algoritmo GRASP+PRm). Nas métricas euclidiana e Pearson, o algoritmo GRASP+PRm foi o único que apresentou mudanças no desvio padrão, já na Cosine a mudança do desvio padrão foi observada no algoritmo GRASP+PRgr. Para ambas as instâncias os melhores tempos médios foram obtidos pelas métricas *City Block*, euclidiana, Cosine e Pearson respectivamente.

A Tabela 4.16 descreve os valores obtidos pela instância **Yeast**. Com relação a função objetivo, os algoritmos híbridos não apresentaram melhorias significativas. A métrica *City Block* apresentou os mesmos valores da FO para todas as estratégias tendo o algoritmo GRASP+PRm o menor DP. Para as demais métricas, as estratégias híbridas não foram eficientes para redução da FO encontrada pelo GRASP mas é possível observar uma pequena redução do DP e tempo na métrica euclidiana, proporcionado pelo algoritmo GRASP+PRm. Nota-se nesta métrica o mesmo comportamento de não relação FO e CRand observado com mais intensidade nas instâncias DLBCLA e DLBCLB. Mesmo apresentando uma FO maior que o GRASP, o algoritmo GRASP+PRb obteve um CRand maior. Nas métricas *City Block* e Pearson, o GRASP apresentou um menor DP. A correlação Cosine apresentou os mesmos resultados da FO para todos os algoritmos, tendo a estratégia GRASP+PRm obtido uma pequena redução no DP. Para esta instância, as correlações apresentaram melhores tempos médios quando comparadas as medidas de distância.

A instância **Protein1** cujos resultados experimentais estão descritos na Tabela 4.17, apresentou variações similares nos DPs do CRand e FO nas métricas euclidiana, Cosine e Pearson. Para todos os algoritmos, o valor da FO manteve-se igual em todas as métricas, com a exceção do GRASP na métrica *City Block*, que embora tenha apresentado o menor valor da FO, não obteve o melhor CRand. Esse fato corrobora com o comportamento observado para as instâncias DLBCLA, DLBCLB e Yeast. Mesmo possuindo uma maior quantidade de grupos que as demais métricas, *City Block* foi a que apresentou os menores tempos médios, seguido das métricas euclidiana, Cosine e Pearson.

A Tabela 4.18 descreve os resultados da instância **Protein2**. Os melhores valores de FO foram obtidos respectivamente pelo algoritmo GRASP na métrica euclidiana, GRASP+PRb na *City Block* e Pearson, e GRASP+PRgr na correlação Cosine. Os melhores tempos médios foram observados nas métricas *City Block*, euclidiana, Cosine e Pearson respectivamente.

Tabela 4.6. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **Breast1**

EUCLIDIANA - Qtd. Grupos = 2

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.878	0.878	0.878	0.0000	687524.839	687524.839	687524.839	0.0000	16.952
GRASP+PRm	0.878	0.878	0.878	0.0000	687524.839	687524.839	687524.839	0.0000	20.004
GRASP+PRb	0.878	0.878	0.878	0.0000	687524.839	687524.839	687524.839	0.0000	19.413
GRASP+PRf	0.878	0.878	0.878	0.0000	687524.839	687524.839	687524.839	0.0000	19.437
GRASP	0.878	0.878	0.878	0.0000	687524.839	687524.839	687524.839	0.0000	18.594

CITY BLOCK - Qtd. Grupos = 2

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.877	0.877	0.877	0.0000	1414121.000	1414121.000	1414121.000	0.0000	14.796
GRASP+PRm	0.877	0.877	0.877	0.0000	1414121.000	1414121.000	1414121.000	0.0000	17.792
GRASP+PRb	0.877	0.877	0.877	0.0000	1414121.000	1414121.000	1414121.000	0.0000	17.446
GRASP+PRf	0.877	0.877	0.877	0.0000	1414121.000	1414121.000	1414121.000	0.0000	17.306
GRASP	0.877	0.877	0.877	0.0000	1414121.000	1414121.000	1414121.000	0.0000	16.578

COSINE - Qtd. Grupos = 3

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.289	0.294	0.294	0.0009	8340.249	8340.275	8340.250	0.0048	34.832
GRASP+PRm	0.294	0.294	0.294	0.0000	8340.249	8340.249	8340.249	0.0000	37.821
GRASP+PRb	0.289	0.294	0.291	0.0025	8340.249	8340.275	8340.266	0.0129	32.837
GRASP+PRf	0.294	0.294	0.294	0.0000	8340.249	8340.249	8340.249	0.0000	35.958
GRASP	0.294	0.294	0.294	0.0000	8340.249	8340.249	8340.249	0.0000	33.869

PEARSON - Qtd. Grupos = 3

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.311	0.311	0.311	0.0000	38752.798	38752.798	38752.798	0.0000	39.464
GRASP+PRm	0.311	0.311	0.311	0.0000	38752.798	38752.798	38752.798	0.0000	42.095
GRASP+PRb	0.311	0.311	0.311	0.0000	38752.798	38753.860	38753.011	0.4319	42.847
GRASP+PRf	0.311	0.311	0.311	0.0000	38752.798	38752.798	38752.798	0.0000	42.871
GRASP	0.311	0.311	0.311	0.0000	38752.798	38752.798	38752.798	0.0000	40.332

Tabela 4.7. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **Breast2**

EUCLIDIANA - Qtd. Grupos = 15

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.008	0.016	0.014	0.0020	57357.669	57588.206	57440.097	59.1910	197.601
GRASP+PRm	0.007	0.016	0.013	0.0022	57332.552	57698.040	57474.336	66.0992	175.241
GRASP+PRb	-0.003	0.014	0.010	0.0034	57408.031	59385.477	57884.799	443.3203	130.223
GRASP+PRf	0.008	0.016	0.013	0.0024	57337.219	57535.413	57430.817	52.3426	209.026
GRASP	0.008	0.017	0.014	0.0020	57370.382	57752.772	57452.335	69.0768	158.492

CITY BLOCK - Qtd. Grupos = 19

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.006	0.016	0.010	0.0024	82835.000	83745.000	83137.467	223.5213	221.760
GRASP+PRm	0.006	0.015	0.011	0.0031	82657.000	83633.000	83109.167	252.7089	229.036
GRASP+PRb	0.006	0.015	0.011	0.0023	82759.000	83589.000	83047.067	199.9381	246.728
GRASP+PRf	0.006	0.015	0.011	0.0028	82598.000	83434.000	83033.633	180.9670	246.153
GRASP	0.004	0.016	0.011	0.0031	82722.000	83570.000	83075.900	207.8513	235.962

COSINE - Qtd. Grupos = 8

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.018	0.022	0.020	0.0008	2115.512	2119.666	2115.834	0.7707	116.402
GRASP+PRm	0.019	0.021	0.020	0.0006	2115.536	2119.121	2115.861	0.6522	114.592
GRASP+PRb	0.019	0.021	0.020	0.0006	2115.512	2116.077	2115.631	0.1371	117.293
GRASP+PRf	0.019	0.022	0.020	0.0007	2115.512	2116.098	2115.654	0.1628	118.414
GRASP	0.019	0.022	0.020	0.0008	2115.523	2116.098	2115.693	0.1778	122.167

PEARSON - Qtd. Grupos = 11

	CRand				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.012	0.017	0.015	0.0013	5891.492	5902.837	5895.167	3.4438	136.432
GRASP+PRm	0.013	0.016	0.014	0.0006	5891.492	5902.521	5894.178	3.3017	145.571
GRASP+PRb	0.013	0.017	0.014	0.0011	5891.409	5903.095	5894.374	3.4347	149.906
GRASP+PRf	0.013	0.016	0.015	0.0009	5891.492	5917.020	5894.359	4.8623	134.871
GRASP	0.012	0.016	0.014	0.0008	5891.409	5934.725	5894.890	8.0526	139.011

Tabela 4.8. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **BreastA**

EUCLIDIANA - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.682	0.682	0.682	0.0000	31746.598	31764.189	31759.246	6.8786	7.798
GRASP+PRm	0.682	0.723	0.683	0.0075	31746.598	31764.347	31760.365	6.6815	6.663
GRASP+PRb	0.682	0.682	0.682	0.0000	31753.058	31764.189	31763.818	2.0322	5.947
GRASP+PRf	0.682	0.682	0.682	0.0000	31746.598	31764.189	31758.756	7.9607	7.039
GRASP	0.682	0.682	0.682	0.0000	31746.598	31764.189	31758.229	7.7014	7.515

CITY BLOCK - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.682	0.723	0.697	0.0201	802761.809	802874.230	802833.009	55.1008	2.162
GRASP+PRm	0.682	0.723	0.700	0.0207	802761.809	802874.230	802825.514	56.6605	2.134
GRASP+PRb	0.682	0.682	0.682	0.0000	802874.230	802874.230	802874.230	0.0000	1.903
GRASP+PRf	0.682	0.723	0.693	0.0184	802590.211	802874.230	802815.849	91.1340	2.076
GRASP	0.682	0.722	0.695	0.0192	802761.809	802874.230	802836.756	53.9013	2.047

COSINE - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.687	0.687	0.687	0.0000	1567.229	1568.096	1567.669	0.4351	15.375
GRASP+PRm	0.687	0.687	0.687	0.0000	1567.229	1568.096	1567.781	0.4210	13.219
GRASP+PRb	0.687	0.687	0.687	0.0000	1568.096	1568.096	1568.096	0.0000	12.566
GRASP+PRf	0.687	0.687	0.687	0.0000	1567.229	1568.096	1567.778	0.4247	12.898
GRASP	0.687	0.687	0.687	0.0000	1567.229	1568.096	1567.643	0.4319	15.320

PEARSON - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.692	0.692	0.692	0.0000	1711.558	1711.558	1711.558	0.0000	22.511
GRASP+PRm	0.692	0.692	0.692	0.0000	1711.558	1711.720	1711.571	0.0436	21.036
GRASP+PRb	0.652	0.692	0.681	0.0180	1711.558	1711.799	1711.666	0.1068	21.136
GRASP+PRf	0.692	0.692	0.692	0.0000	1711.558	1711.558	1711.558	0.0000	23.305
GRASP	0.692	0.692	0.692	0.0000	1711.558	1711.558	1711.558	0.0000	21.723

Tabela 4.9. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **BreastB1**

EUCLIDIANA - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.694	0.694	0.694	0.0000	11679.080	11679.080	11679.080	0.0000	2.012
GRASP+PRm	0.694	0.694	0.694	0.0000	11679.080	11679.080	11679.080	0.0000	2.068
GRASP+PRb	0.694	0.694	0.694	0.0000	11679.080	11680.317	11679.575	0.6162	2.009
GRASP+PRf	0.694	0.694	0.694	0.0000	11679.080	11679.080	11679.080	0.0000	2.136
GRASP	0.694	0.694	0.694	0.0000	11679.080	11679.080	11679.080	0.0000	2.024

CITY BLOCK - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.074	0.366	0.240	0.1052	133027.081	137032.281	134011.183	746.3320	1.491
GRASP+PRm	0.072	0.366	0.264	0.1032	133027.081	135392.671	133831.560	359.1399	1.540
GRASP+PRb	0.080	0.261	0.205	0.0719	134518.302	141937.954	137534.967	1897.1025	1.013
GRASP+PRf	0.074	0.288	0.140	0.0397	132996.726	134518.302	133649.979	623.2515	1.589
GRASP	0.062	0.328	0.245	0.1048	133027.081	134518.302	133672.996	280.8159	1.608

COSINE - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.694	0.694	0.694	0.0000	16.956	16.956	16.956	0.0000	3.177
GRASP+PRm	0.694	0.694	0.694	0.0000	16.952	16.956	16.956	0.0008	3.098
GRASP+PRb	0.694	0.694	0.694	0.0000	16.952	16.956	16.955	0.0019	3.457
GRASP+PRf	0.694	0.694	0.694	0.0000	16.952	16.956	16.954	0.0022	3.243
GRASP	0.694	0.694	0.694	0.0000	16.952	16.956	16.955	0.0021	3.612

PEARSON - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.500	0.766	0.639	0.0616	177.993	178.339	178.283	0.1259	5.768
GRASP+PRm	0.626	0.766	0.672	0.0618	177.993	178.339	178.211	0.1617	5.906
GRASP+PRb	-0.021	0.626	0.238	0.3224	178.339	187.468	183.817	4.5488	4.791
GRASP+PRf	0.626	0.766	0.682	0.0698	177.993	178.339	178.201	0.1727	6.752
GRASP	0.626	0.766	0.645	0.0484	177.993	178.339	178.293	0.1199	5.594

Tabela 4.10. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **BreastB2**

EUCLIDIANA - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.322	0.322	0.322	0.0000	11679.080	11679.080	11679.080	0.0000	2.016
GRASP+PRm	0.322	0.322	0.322	0.0000	11679.080	11679.080	11679.080	0.0000	2.107
GRASP+PRb	0.322	0.322	0.322	0.0000	11679.080	11680.317	11679.575	0.6162	1.956
GRASP+PRf	0.322	0.322	0.322	0.0000	11679.080	11679.080	11679.080	0.0000	2.141
GRASP	0.321	0.321	0.321	0.0000	11679.080	11679.080	11679.080	0.0000	2.027

CITY BLOCK - Qtd. Grupos = 7

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.147	0.333	0.238	0.0475	69170.544	70906.325	70069.809	405.0838	1.624
GRASP+PRm	0.122	0.344	0.219	0.0504	68808.009	71286.806	70187.262	507.5346	1.667
GRASP+PRb	0.066	0.244	0.160	0.0433	69981.923	74098.362	72130.609	1002.0902	1.002
GRASP+PRf	0.137	0.328	0.221	0.0398	68970.580	69981.923	69754.189	315.6761	1.481
GRASP	0.147	0.367	0.256	0.0568	68903.198	70836.189	69771.520	452.1238	1.704

COSINE - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.322	0.322	0.322	0.0000	16.956	16.956	16.956	0.0000	3.154
GRASP+PRm	0.322	0.322	0.322	0.0000	16.952	16.956	16.956	0.0008	3.091
GRASP+PRb	0.322	0.322	0.322	0.0000	16.956	16.961	16.958	0.0025	3.095
GRASP+PRf	0.322	0.322	0.322	0.0000	16.952	16.956	16.955	0.0019	3.092
GRASP	0.321	0.321	0.321	0.0000	16.952	16.956	16.955	0.0021	3.614

PEARSON - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.240	0.322	0.248	0.0199	177.993	178.339	178.283	0.1259	5.796
GRASP+PRm	0.240	0.322	0.262	0.0299	177.993	178.339	178.211	0.1617	5.940
GRASP+PRb	0.240	0.306	0.280	0.0329	178.339	187.468	183.817	4.5488	4.854
GRASP+PRf	0.240	0.281	0.251	0.0184	177.993	178.339	178.247	0.1559	6.268
GRASP	0.238	0.279	0.243	0.0142	177.993	178.339	178.293	0.1199	5.598

Tabela 4.11. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **DLBCLA**

EUCLIDIANA - Qtd. Grupos = 4

									Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.332	0.465	0.373	0.0349	29435407.826	29457484.636	29445664.508	6338.0785	12.603
GRASP+PRm	0.332	0.431	0.370	0.0287	29435407.826	29466872.135	29445966.347	7792.6139	10.837
GRASP+PRb	0.104	0.504	0.387	0.0906	29439800.444	29982819.076	29526413.592	133376.5795	7.095
GRASP+PRf	0.332	0.408	0.356	0.0195	29435407.826	29447887.991	29440738.301	3779.6904	11.694
GRASP	0.332	0.443	0.368	0.0297	29435407.826	29458319.710	29445002.384	6419.7172	11.399

CITY BLOCK - Qtd. Grupos = 3

	CRAND				**Função Objetivo**				**Tempo**
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.760	0.838	0.815	0.0258	460537685.653	460603105.651	460551382.821	19520.7064	2.623
GRASP+PRm	0.760	0.838	0.817	0.0246	460537685.653	460603105.651	460550540.973	19020.5081	2.687
GRASP+PRb	0.275	0.838	0.627	0.2074	460537685.653	464887772.647	461846611.725	1721285.5928	2.041
GRASP+PRf	0.801	0.838	0.827	0.0172	460537685.653	460555573.574	460543052.029	8337.3972	2.563
GRASP	0.760	0.838	0.823	0.0212	460537685.653	460606933.953	460545956.570	14315.2399	2.518

COSINE - Qtd. Grupos = 4

	CRAND				**Função Objetivo**				**Tempo**
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.500	0.607	0.597	0.0202	209.017	210.229	209.079	0.2208	18.682
GRASP+PRm	0.586	0.619	0.602	0.0088	209.017	209.095	209.040	0.0245	18.827
GRASP+PRb	0.311	0.607	0.463	0.0935	209.017	213.412	210.967	1.4796	12.522
GRASP+PRf	0.573	0.607	0.602	0.0080	209.017	209.203	209.028	0.0351	14.765
GRASP	0.573	0.607	0.600	0.0093	209.017	209.098	209.033	0.0232	15.963

PEARSON - Qtd. Grupos = 4

	CRAND				**Função Objetivo**				**Tempo**
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.586	0.604	0.602	0.0055	245.789	245.828	245.797	0.0137	28.910
GRASP+PRm	0.586	0.604	0.600	0.0077	245.789	245.841	245.804	0.0178	27.541
GRASP+PRb	0.336	0.607	0.486	0.1123	245.813	253.677	249.028	3.0700	20.810
GRASP+PRf	0.604	0.604	0.604	0.0000	245.789	245.813	245.793	0.0094	32.208
GRASP	0.585	0.604	0.601	0.0066	245.789	245.828	245.798	0.0148	26.255

Tabela 4.12. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **DLBCLB**

EUCLIDIANA - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.420	0.520	0.492	0.0234	106431.030	106526.654	106474.935	27.1797	19.054
GRASP+PRm	0.420	0.519	0.484	0.0224	106418.928	106556.492	106468.402	26.3842	19.145
GRASP+PRb	0.230	0.537	0.339	0.0998	106458.677	107602.738	106999.512	396.0039	11.078
GRASP+PRf	0.420	0.509	0.491	0.0213	106418.928	106556.492	106456.111	27.5364	21.644
GRASP	0.448	0.519	0.494	0.0187	106426.763	106532.770	106461.567	26.9601	18.597

CITY BLOCK - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.699	0.701	0.700	0.0010	4308801.344	4309153.247	4308942.105	175.3437	3.575
GRASP+PRm	0.699	0.701	0.700	0.0010	4308801.344	4309153.247	4308965.565	178.5613	3.428
GRASP+PRb	0.137	0.703	0.681	0.1027	4308801.344	4366778.894	4311082.784	10519.7270	2.906
GRASP+PRf	0.699	0.701	0.700	0.0010	4308801.344	4309153.247	4308942.105	175.3437	3.467
GRASP	0.699	0.701	0.700	0.0009	4308801.344	4309153.247	4308906.915	164.0190	3.478

COSINE - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.468	0.500	0.496	0.0076	3164.281	3165.175	3164.450	0.2571	36.326
GRASP+PRm	0.434	0.500	0.493	0.0162	3164.281	3169.330	3164.735	1.0858	28.896
GRASP+PRb	0.255	0.500	0.378	0.0872	3164.281	3218.045	3191.463	21.2296	23.316
GRASP+PRf	0.489	0.500	0.499	0.0035	3164.281	3165.192	3164.341	0.1864	25.353
GRASP	0.467	0.500	0.497	0.0071	3164.281	3166.081	3164.405	0.3536	36.909

PEARSON - Qtd. Grupos = 2

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.585	0.585	0.585	0.0000	7015.971	7015.971	7015.971	0.0000	48.491
GRASP+PRm	0.585	0.585	0.585	0.0000	7015.971	7016.414	7016.030	0.1530	43.118
GRASP+PRb	0.500	0.585	0.565	0.0326	7015.971	7022.243	7017.417	2.3704	37.098
GRASP+PRf	0.585	0.585	0.585	0.0000	7015.971	7015.971	7015.971	0.0000	45.351
GRASP	0.585	0.585	0.585	0.0000	7015.971	7015.971	7015.971	0.0000	41.886

Tabela 4.13. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **Iris**

EUCLIDIANA - Qtd. Grupos = 3

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.757	0.757	0.757	0.0000	3409.739	3409.739	3409.739	0.0000	0.324
GRASP+PRm	0.757	0.757	0.757	0.0000	3409.739	3409.739	3409.739	0.0000	0.323
GRASP+PRb	0.757	0.757	0.757	0.0000	3409.739	3409.739	3409.739	0.0000	0.399
GRASP+PRf	0.757	0.757	0.757	0.0000	3409.739	3409.739	3409.739	0.0000	0.411
GRASP	0.757	0.757	0.757	0.0000	3409.739	3409.739	3409.739	0.0000	0.382

CITY BLOCK - Qtd. Grupos = 3

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.818	0.818	0.818	0.0000	5745.600	5745.600	5745.600	0.0000	0.285
GRASP+PRm	0.818	0.818	0.818	0.0000	5745.600	5745.600	5745.600	0.0000	0.292
GRASP+PRb	0.818	0.818	0.818	0.0000	5745.600	5745.600	5745.600	0.0000	0.363
GRASP+PRf	0.771	0.818	0.815	0.0119	5745.600	5745.600	5745.600	0.0000	0.369
GRASP	0.818	0.818	0.818	0.0000	5745.600	5745.600	5745.600	0.0000	0.347

COSINE - Qtd. Grupos = 3

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.942	0.942	0.942	0.0000	8.166	8.166	8.166	0.0000	0.367
GRASP+PRm	0.942	0.942	0.942	0.0000	8.166	8.166	8.166	0.0000	0.363
GRASP+PRb	0.942	0.942	0.942	0.0000	8.166	8.166	8.166	0.0000	0.432
GRASP+PRf	0.942	0.942	0.942	0.0000	8.166	8.166	8.166	0.0000	0.427
GRASP	0.942	0.942	0.942	0.0000	8.166	8.166	8.166	0.0000	0.409

PEARSON - Qtd. Grupos = 3

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.886	0.886	0.886	0.0000	21.948	21.948	21.948	0.0000	0.754
GRASP+PRm	0.886	0.886	0.886	0.0000	21.948	21.948	21.948	0.0000	0.653
GRASP+PRb	0.886	0.886	0.886	0.0000	21.948	21.948	21.948	0.0000	0.809
GRASP+PRf	0.886	0.886	0.886	0.0000	21.948	21.948	21.948	0.0000	0.808
GRASP	0.886	0.886	0.886	0.0000	21.948	21.948	21.948	0.0000	0.784

Tabela 4.14. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **MultiA**

EUCLIDIANA - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.874	0.874	0.874	0.0000	55453218.807	55453218.807	55453218.807	0.0000	38.072
GRASP+PRm	0.851	0.874	0.873	0.0057	55453218.807	55461621.499	55453750.779	2027.5322	36.034
GRASP+PRb	0.874	0.874	0.874	0.0000	55453218.807	55453218.807	55453218.807	0.0000	34.311
GRASP+PRf	0.874	0.874	0.874	0.0000	55453218.807	55453218.807	55453218.807	0.0000	35.581
GRASP	0.874	0.874	0.874	0.0000	55453218.807	55453218.807	55453218.807	0.0000	34.019

CITY BLOCK - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.851	0.899	0.885	0.0157	1481082000.794	1482742282.048	1481793140.963	332380.2828	13.846
GRASP+PRm	0.851	0.924	0.884	0.0169	1481082000.794	1482657684.057	1481829498.835	324229.1617	14.649
GRASP+PRb	0.851	0.899	0.870	0.0177	1481082000.794	1481843741.913	1481511389.332	337100.6384	14.528
GRASP+PRf	0.851	0.899	0.875	0.0195	1481082000.794	1481929023.612	1481560937.255	323008.0098	14.286
GRASP	0.851	0.899	0.882	0.0139	1481082000.794	1482742282.048	1481745997.223	293855.0615	13.509

COSINE - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.768	0.831	0.829	0.0115	165.416	165.425	165.416	0.0017	66.010
GRASP+PRm	0.831	0.831	0.831	0.0000	165.416	165.416	165.416	0.0000	62.048
GRASP+PRb	0.831	0.831	0.831	0.0000	165.416	165.416	165.416	0.0000	61.552
GRASP+PRf	0.831	0.831	0.831	0.0000	165.416	165.416	165.416	0.0000	56.764
GRASP	0.831	0.831	0.831	0.0000	165.416	165.416	165.416	0.0000	59.108

PEARSON - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.829	0.829	0.829	0.0000	196.102	196.102	196.102	0.0000	113.093
GRASP+PRm	0.740	0.829	0.826	0.0162	196.102	196.110	196.102	0.0015	123.760
GRASP+PRb	0.829	0.829	0.829	0.0000	196.102	196.102	196.102	0.0000	102.278
GRASP+PRf	0.829	0.829	0.829	0.0000	196.102	196.102	196.102	0.0000	108.798
GRASP	0.829	0.829	0.829	0.0000	196.102	196.102	196.102	0.0000	96.462

Tabela 4.15. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **Novartis**

EUCLIDIANA - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.950	0.950	0.950	0.0000	46554.627	46554.627	46554.627	0.0000	7.489
GRASP+PRm	0.950	0.950	0.950	0.0000	46554.627	46554.627	46554.627	0.0000	7.046
GRASP+PRb	0.950	0.950	0.950	0.0000	46554.627	46554.627	46554.627	0.0000	7.115
GRASP+PRf	0.950	0.950	0.950	0.0000	46554.627	46554.627	46554.627	0.0000	7.458
GRASP	0.950	0.950	0.950	0.0000	46554.627	46554.627	46554.627	0.0000	6.504

CITY BLOCK - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.950	0.950	0.950	0.0000	1033080.467	1033080.467	1033080.467	0.0000	3.045
GRASP+PRm	0.950	0.950	0.950	0.0000	1033080.467	1033080.467	1033080.467	0.0000	3.109
GRASP+PRb	0.950	0.950	0.950	0.0000	1033080.467	1033080.467	1033080.467	0.0000	2.973
GRASP+PRf	0.950	0.950	0.950	0.0000	1033080.467	1033080.467	1033080.467	0.0000	3.093
GRASP	0.950	0.950	0.950	0.0000	1033080.467	1033080.467	1033080.467	0.0000	3.024

COSINE - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.950	0.950	0.950	0.0000	895.503	895.503	895.503	0.0000	16.356
GRASP+PRm	0.950	0.950	0.950	0.0000	895.503	895.503	895.503	0.0000	14.762
GRASP+PRb	0.950	0.950	0.950	0.0000	895.503	895.503	895.503	0.0000	15.534
GRASP+PRf	0.950	0.950	0.950	0.0000	895.503	895.503	895.503	0.0000	12.501
GRASP	0.950	0.950	0.950	0.0000	895.503	895.503	895.503	0.0000	14.796

PEARSON - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.950	0.950	0.950	0.0000	894.191	894.191	894.191	0.0000	26.031
GRASP+PRm	0.950	0.950	0.950	0.0000	894.191	894.191	894.191	0.0000	21.785
GRASP+PRb	0.950	0.950	0.950	0.0000	894.191	894.191	894.191	0.0000	23.362
GRASP+PRf	0.950	0.950	0.950	0.0000	894.191	894.191	894.191	0.0000	27.304
GRASP	0.950	0.950	0.950	0.0000	894.191	894.191	894.191	0.0000	23.449

Tabela 4.16. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **Yeast**

EUCLIDIANA - Qtd. Grupos = 9

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.131	0.151	0.143	0.0054	28113.476	28144.615	28121.595	8.6697	1636.252
GRASP+PRm	0.131	0.153	0.144	0.0061	28113.410	28141.691	28122.129	8.1635	1021.809
GRASP+PRb	0.132	0.153	0.144	0.0063	28113.476	28142.715	28125.202	8.4172	2037.720
GRASP+PRf	0.129	0.150	0.138	0.0060	28113.893	28147.426	28131.351	10.3473	1283.363
GRASP	0.131	0.151	0.143	0.0059	28113.410	28141.800	28122.401	8.6668	1363.339

CITY BLOCK - Qtd. Grupos = 7

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.151	0.161	0.156	0.0030	73712.810	73746.750	73723.599	11.2188	1132.612
GRASP+PRm	0.145	0.159	0.156	0.0031	73712.960	73751.840	73723.479	13.0608	801.195
GRASP+PRb	0.151	0.160	0.156	0.0023	73712.810	73739.950	73721.642	9.4262	1535.764
GRASP+PRf	0.147	0.161	0.155	0.0034	73714.170	73771.550	73726.189	13.9272	1032.244
GRASP	0.150	0.161	0.158	0.0019	73712.810	73751.050	73716.264	7.5182	1013.296

COSINE - Qtd. Grupos = 9

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.110	0.137	0.129	0.0105	2102.333	2104.719	2102.849	0.8267	1400.802
GRASP+PRm	0.110	0.137	0.128	0.0115	2102.333	2104.299	2102.902	0.8105	969.030
GRASP+PRb	0.110	0.137	0.126	0.0123	2102.333	2104.591	2103.026	0.8847	983.654
GRASP+PRf	0.110	0.137	0.126	0.0124	2102.333	2104.591	2103.027	0.8847	1377.021
GRASP	0.110	0.136	0.128	0.0111	2102.333	2104.357	2102.970	0.8274	908.616

PEARSON - Qtd. Grupos = 9

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.109	0.138	0.127	0.0129	9658.331	9671.834	9661.686	4.3530	1463.359
GRASP+PRm	0.109	0.138	0.128	0.0126	9658.331	9672.055	9661.460	4.2471	1039.384
GRASP+PRb	0.108	0.138	0.125	0.0134	9658.415	9671.735	9662.449	4.3589	897.521
GRASP+PRf	0.109	0.138	0.124	0.0134	9658.415	9670.969	9662.764	4.3800	896.042
GRASP	0.109	0.138	0.129	0.0124	9658.331	9667.199	9660.914	3.8458	852.922

Tabela 4.17. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **Protein1**

EUCLIDIANA - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.271	0.297	0.288	0.0087	5788404.396	5788679.556	5788528.430	51.8281	91.416
GRASP+PRm	0.271	0.297	0.288	0.0085	5788404.396	5788679.556	5788530.169	46.4123	88.550
GRASP+PRb	0.271	0.294	0.286	0.0083	5788404.396	5788545.015	5788500.582	52.8336	85.419
GRASP+PRf	0.271	0.294	0.286	0.0089	5788404.396	5788558.107	5788501.356	56.3714	89.871
GRASP	0.271	0.294	0.288	0.0089	5788404.396	5788545.015	5788512.961	52.5951	79.166

CITY BLOCK - Qtd. Grupos = 5

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.259	0.310	0.296	0.0135	31980190.600	31987039.000	31982625.343	1834.4075	79.963
GRASP+PRm	0.252	0.310	0.299	0.0123	31980718.200	31988048.000	31982666.413	1940.3799	75.583
GRASP+PRb	0.268	0.309	0.300	0.0080	31980190.600	31985905.500	31981741.580	1319.5710	77.607
GRASP+PRf	0.259	0.309	0.299	0.0084	31980190.600	31985406.400	31981351.963	1190.6153	89.123
GRASP	0.254	0.309	0.297	0.0141	31980107.600	31986867.400	31982059.953	1743.9223	71.245

COSINE - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.339	0.350	0.341	0.0028	1036.075	1036.129	1036.095	0.0191	130.570
GRASP+PRm	0.339	0.348	0.341	0.0022	1036.075	1036.116	1036.093	0.0170	131.986
GRASP+PRb	0.339	0.344	0.340	0.0009	1036.075	1036.114	1036.086	0.0158	132.622
GRASP+PRf	0.340	0.342	0.340	0.0004	1036.075	1036.110	1036.077	0.0065	117.881
GRASP	0.339	0.348	0.341	0.0022	1036.075	1036.115	1036.093	0.0187	120.200

PEARSON - Qtd. Grupos = 4

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.345	0.345	0.345	0.0000	2328.064	2328.126	2328.077	0.0148	201.051
GRASP+PRm	0.344	0.345	0.345	0.0002	2328.064	2328.085	2328.078	0.0078	167.804
GRASP+PRb	0.345	0.345	0.345	0.0000	2328.064	2328.085	2328.070	0.0058	182.977
GRASP+PRf	0.339	0.345	0.345	0.0011	2328.064	2328.586	2328.086	0.0947	205.839
GRASP	0.345	0.345	0.345	0.0000	2328.064	2328.126	2328.076	0.0117	168.529

Tabela 4.18. Valores mínimo, máximo, média e desvio padrão obtidos para o CRand e Função objetivo para a instância **Protein2**

EUCLIDIANA - Qtd. Grupos = 11

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.143	0.169	0.159	0.0054	1761314.964	1762639.545	1761948.348	347.8548	155.813
GRASP+PRm	0.151	0.168	0.158	0.0046	1761298.342	1762541.456	1761924.685	285.4350	165.651
GRASP+PRb	0.150	0.167	0.160	0.0045	1761107.845	1762558.298	1761603.286	310.8329	186.314
GRASP+PRf	0.152	0.169	0.162	0.0047	1761075.771	1762566.510	1761591.723	361.8837	172.177
GRASP	0.152	0.168	0.161	0.0044	1761062.649	1761972.089	1761558.537	268.1833	170.232

CITY BLOCK - Qtd. Grupos = 9

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.150	0.179	0.164	0.0075	16528658.500	16545758.800	16535647.907	4560.7931	117.230
GRASP+PRm	0.148	0.176	0.164	0.0064	16530062.900	16544581.900	16537703.263	4245.5602	127.100
GRASP+PRb	0.153	0.180	0.165	0.0060	16525616.300	16536903.300	16531281.953	2802.9023	129.922
GRASP+PRf	0.146	0.180	0.163	0.0086	16527432.100	16541503.100	16533602.867	3998.9260	124.127
GRASP	0.154	0.175	0.167	0.0054	16526838.300	16542695.500	16532219.613	4058.4162	118.625

COSINE - Qtd. Grupos = 12

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.143	0.170	0.155	0.0063	238.014	238.267	238.111	0.0613	209.951
GRASP+PRm	0.140	0.170	0.156	0.0074	238.018	238.203	238.090	0.0463	218.111
GRASP+PRb	0.147	0.170	0.157	0.0065	238.016	238.170	238.073	0.0367	249.879
GRASP+PRf	0.149	0.170	0.155	0.0048	238.015	238.187	238.094	0.0464	206.689
GRASP	0.144	0.170	0.154	0.0069	238.015	238.166	238.078	0.0433	201.082

PEARSON - Qtd. Grupos = 12

	CRAND				Função Objetivo				Tempo
Algoritmo	**Mínimo**	**Máximo**	**Média**	**DP**	**Mínimo**	**Máximo**	**Média**	**DP**	**Médio**
GRASP+PRgr	0.139	0.172	0.154	0.0073	536.956	537.561	537.221	0.1301	300.791
GRASP+PRm	0.140	0.172	0.155	0.0084	536.987	537.423	537.207	0.1141	253.668
GRASP+PRb	0.129	0.172	0.156	0.0097	536.954	537.269	537.123	0.0981	331.141
GRASP+PRf	0.131	0.169	0.155	0.0083	536.990	537.502	537.170	0.1080	313.532
GRASP	0.130	0.172	0.155	0.0089	536.987	537.436	537.174	0.1111	263.325

As figuras a seguir ilustram os TTTplots para as implementações do GRASP e variantes GRASP+PR para todas as instâncias analisadas neste trabalho. Para cada instância foram realizadas 200 execuções por algoritmo e os tempos estão em segundos. Para 11 das 13 instâncias os gráficos mostram que o GRASP+PR é capaz de encontrar soluções alvo mais rapidamente que o GRASP.

O algoritmo GRASP+PRb apresentou a maior quantidade de melhores tempos de execução. Nas instâncias BreastA (Figura 4.4), BreastB2 (Figura 4.6) e DLBCLB (Figura 4.8) o algoritmo apresentou os melhores tempos de execução com uma diferença considerável em relação as demais variantes. Nas instâncias BreastB1 (Figura 4.5), DLBCLA (Figura 4.7), MultiA (Figura 4.10) e Yeast (Figura 4.12) o GRASP+PRb juntamente com o GRASP+PRf apresentaram resultados similares e superiores as demais variantes.

O algoritmo GRASP+PRgr apresentou os melhores tempos de execução para as instâncias Breast1 (Figura 4.2) e Iris (Figura 4.9). As instâncias Breast2 (Figura 4.3) e Novartis (Figura 4.11) não apresentaram diferença significativa nos tempos de execução dos algoritmos.

A instância Protein1 (Figura 4.13) obteve os melhores tempos com os algoritmos GRASP e GRASP+PRb, e na instância Protein2 (Figura 4.14) o GRASP apresentou o melhor tempo de execução com uma pequena diferença com relação ao algoritmo GRASP+PRb.

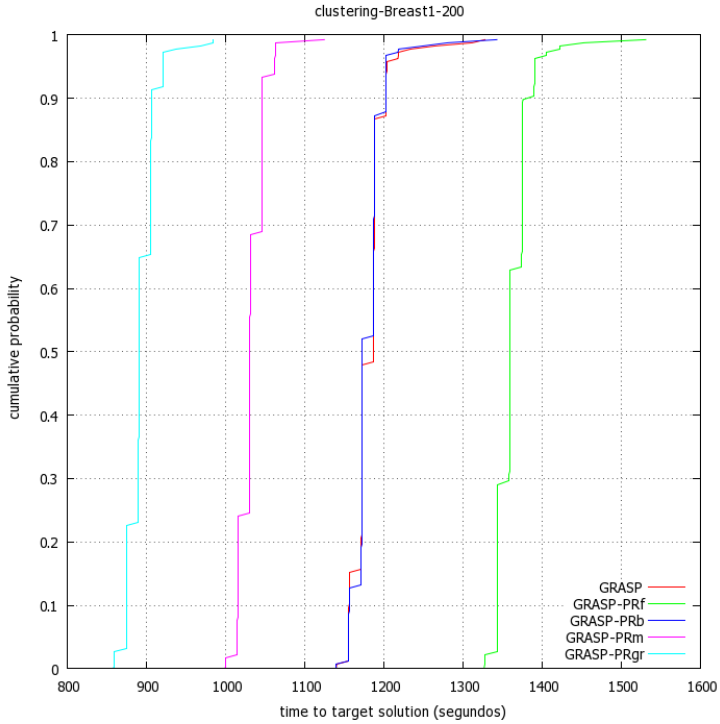

Figura 4.2. Gráfico Time-to-Target para a instância Breast1.

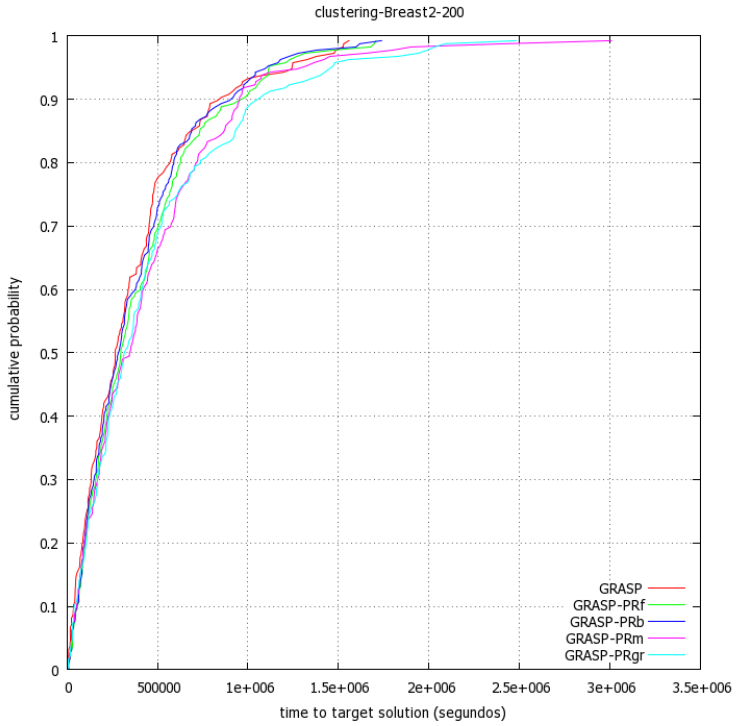

Figura 4.3. Gráfico Time-to-Target para a instância Breast2.

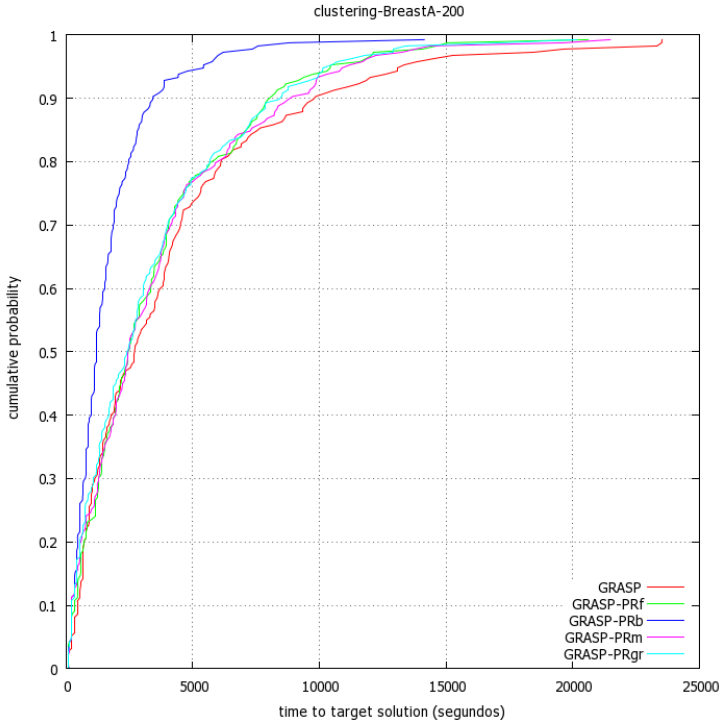

Figura 4.4. Gráfico Time-to-Target para a instância BreastA.

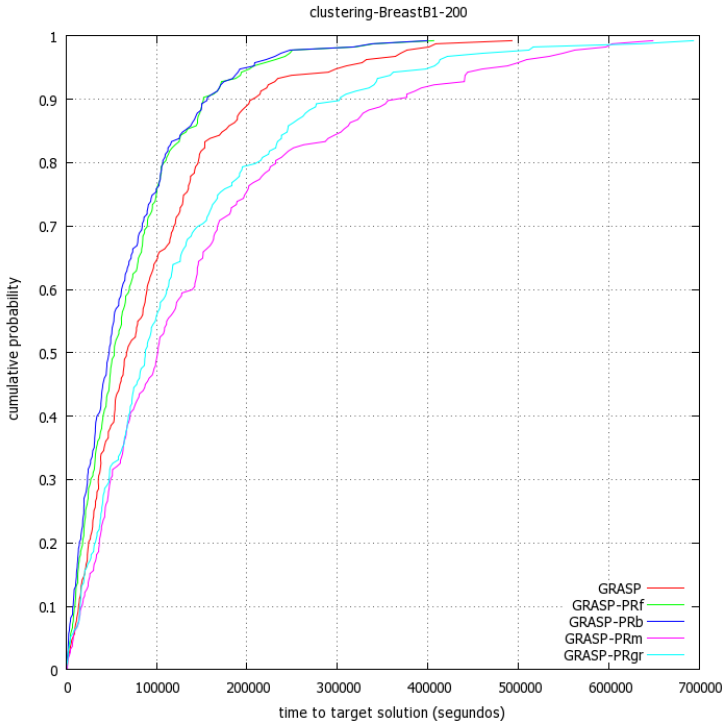

Figura 4.5. Gráfico Time-to-Target para a instância BreastB1.

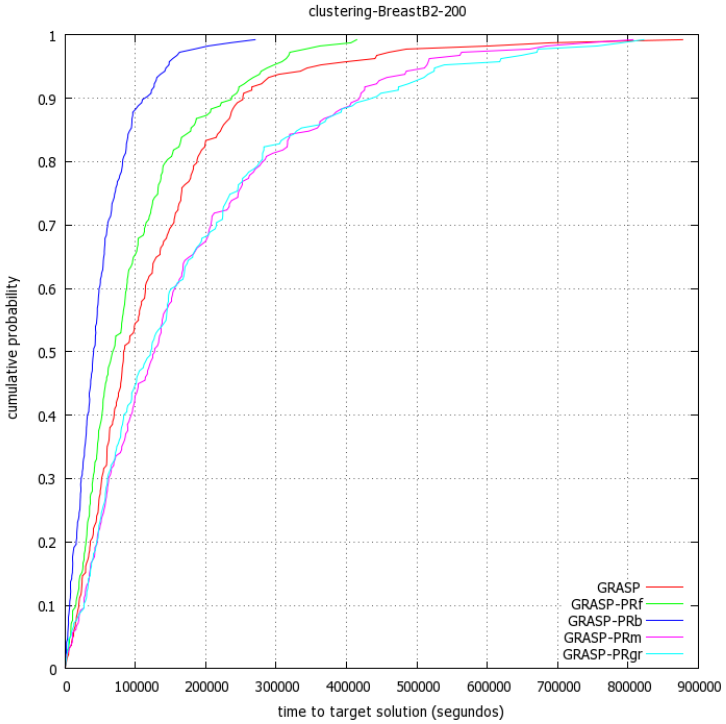

Figura 4.6. Gráfico Time-to-Target para a instância BreastB2.

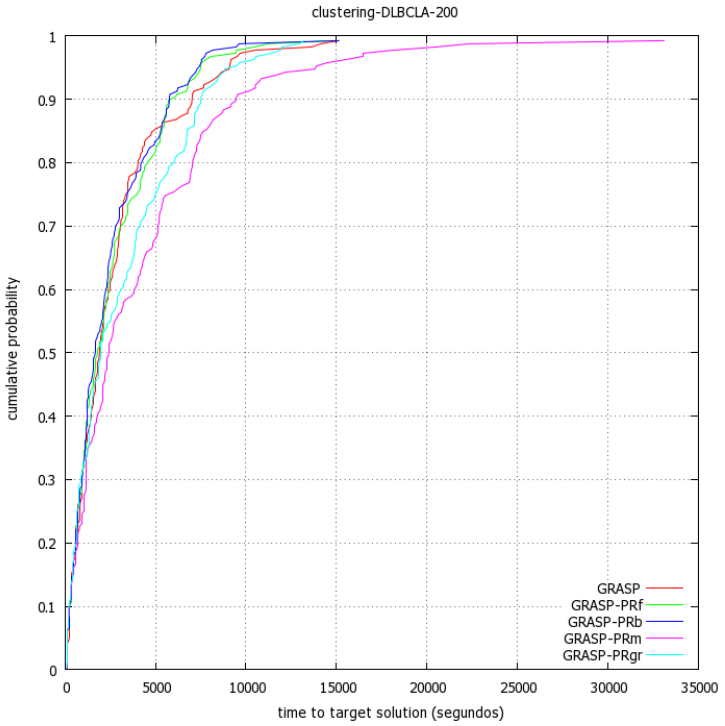

Figura 4.7. Gráfico Time-to-Target para a instância DLBCLA.

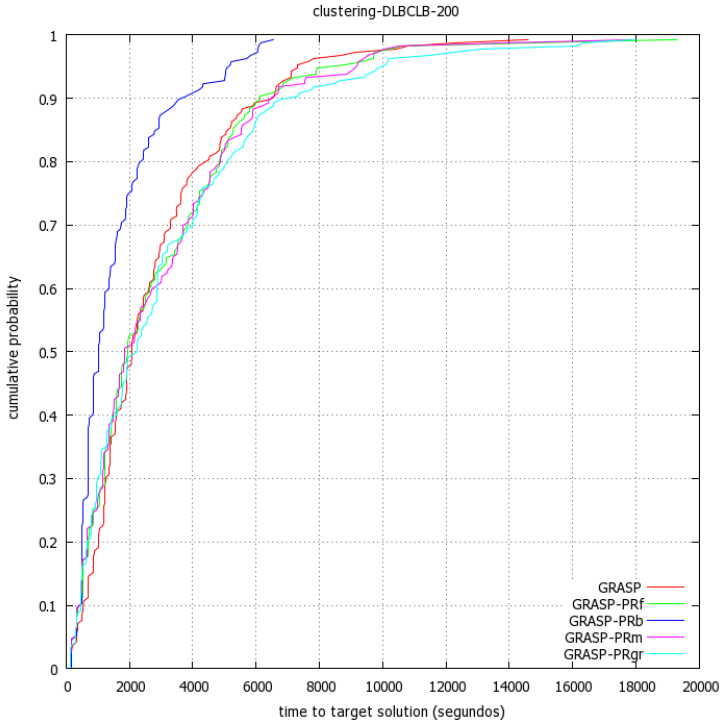

Figura 4.8. Gráfico Time-to-Target para a instância DLBCLB.

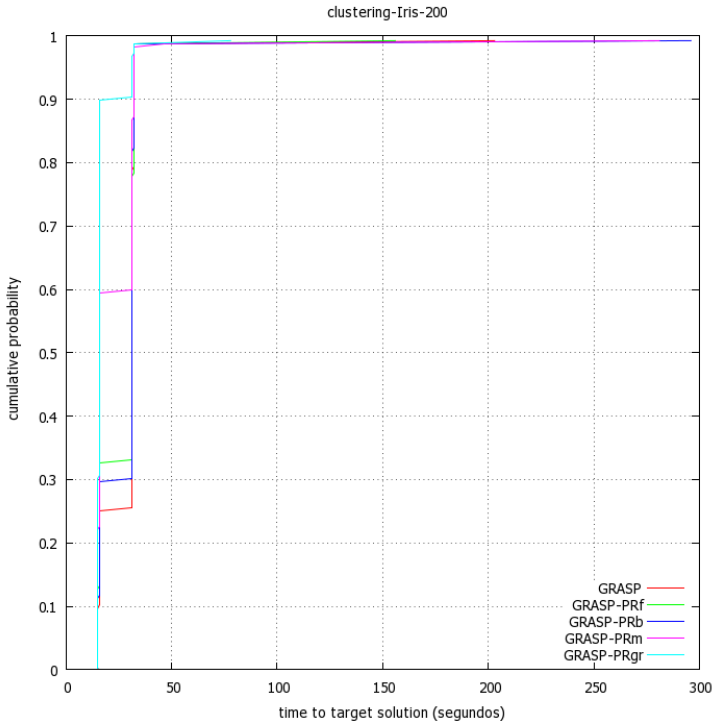

Figura 4.9. Gráfico Time-to-Target para a instância Iris.

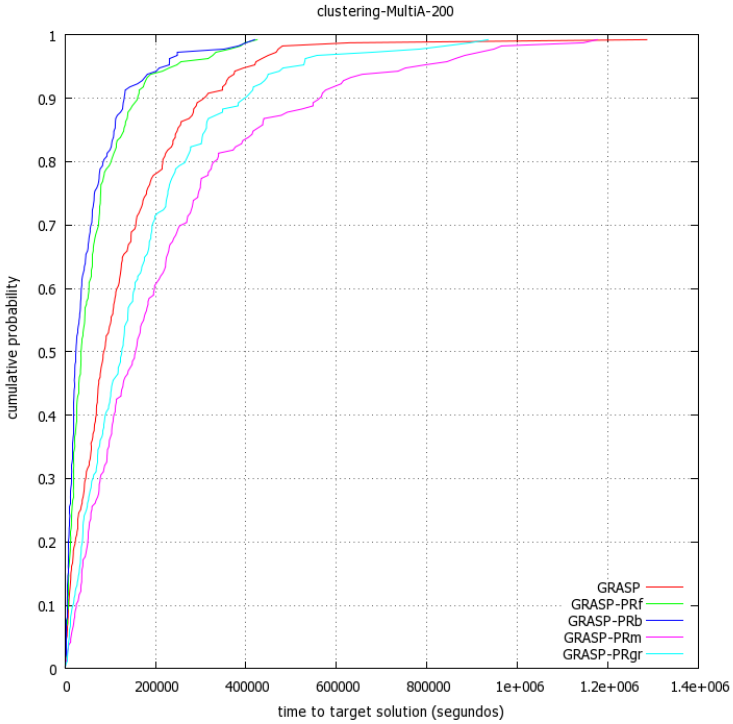

Figura 4.10. Gráfico Time-to-Target para a instância MultiA.

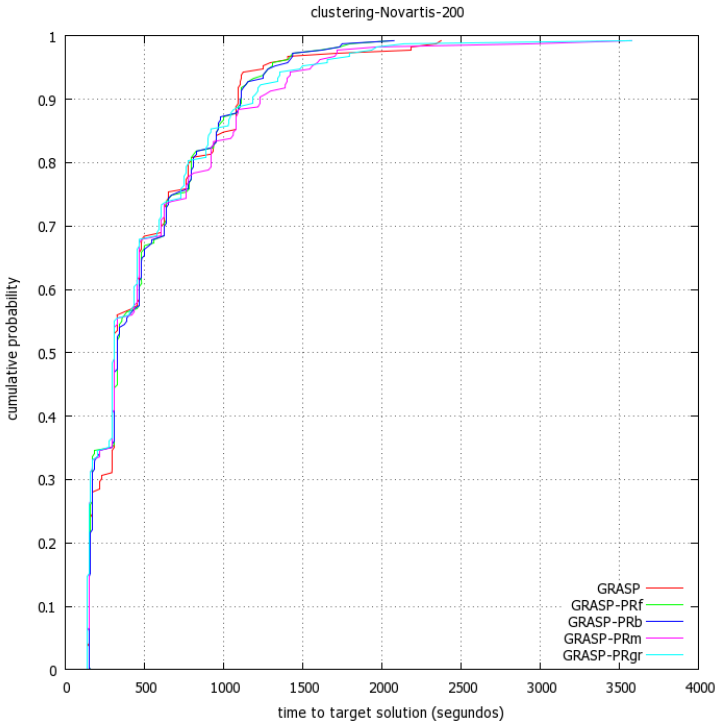

Figura 4.11. Gráfico Time-to-Target para a instância Novartis.

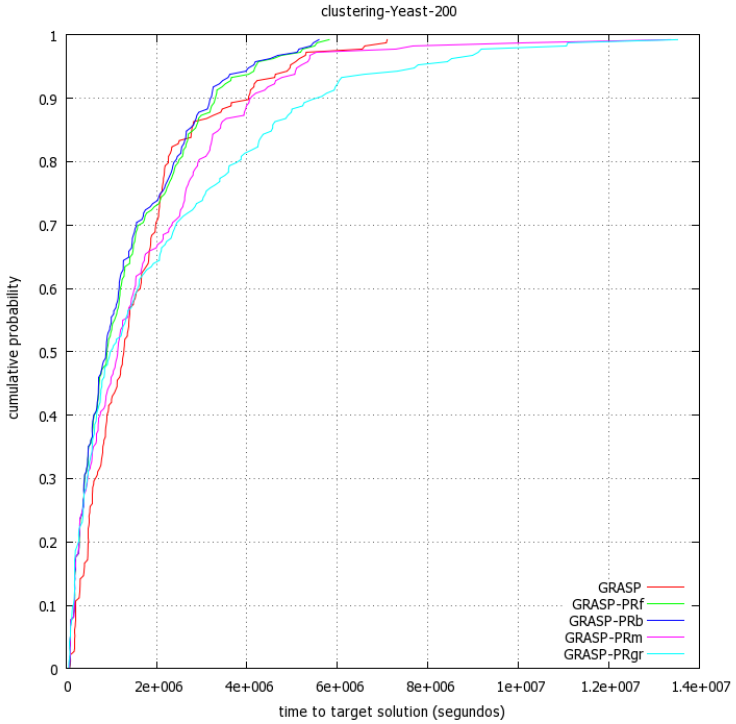

Figura 4.12. Gráfico Time-to-Target para a instância Yeast.

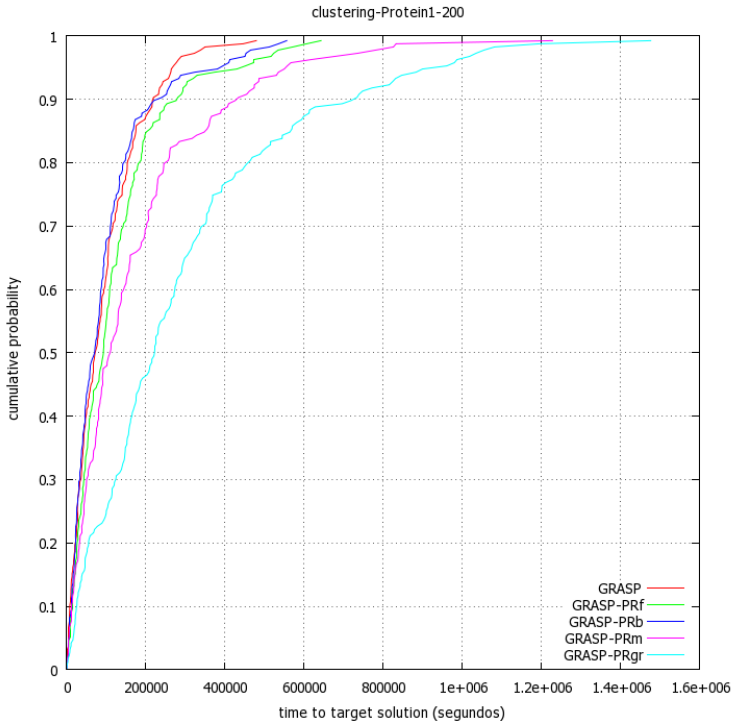

Figura 4.13. Gráfico Time-to-Target para a instância Protein1.

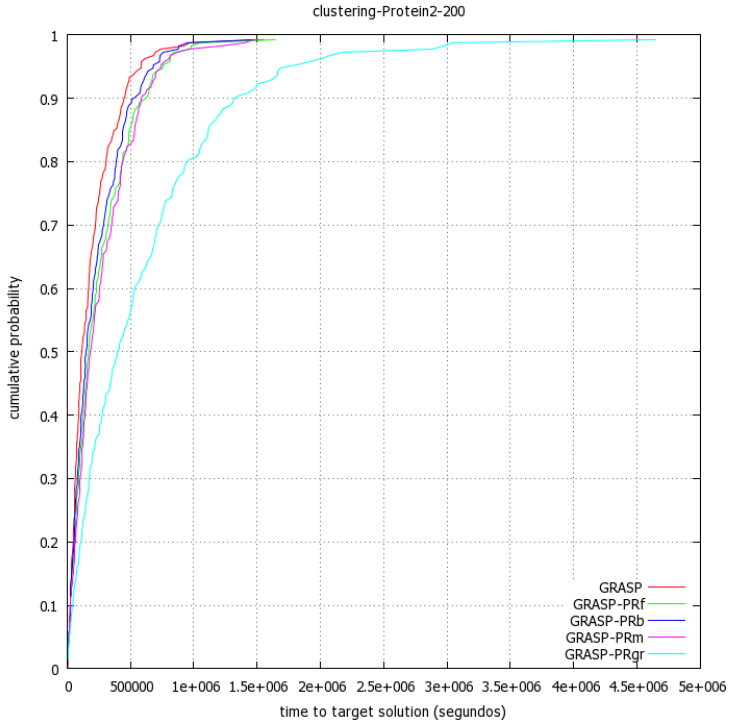

Figura 4.14. Gráfico Time-to-Target para a instância Protein2.

Capítulo 5

Conclusão e Trabalhos Futuros

Este capítulo destaca os principais resultados deste trabalho além de fornecer sugestões para o direcionamento de pesquisas futuras relacionadas à continuidade do tema exposto.

O objetivo desse trabalho foi propor algoritmos híbridos fundamentados nas heurísticas GRASP e *Path-Relinking* para o problema de agrupamento de dados biológicos, com intuito de obter melhores soluções quando considerado unicamente o GRASP na sua forma padrão. Neste trabalho, considerou-se a hipótese que o *Path-Relinking* como estratégia de intensificação do GRASP melhora o desempenho e qualidade das soluções, quando comparadas ao GRASP na sua forma padrão.

Para testar essa hipótese, foi realizada a hibridização do GRASP proposto por Nascimento et al. [2010b], com quatro variações do *Path-Relinking*: *Forward, Backward, Mixed, e Greedy Randomized Adaptative*. A validação das soluções obtidas deu-se por meio da comparação com os algoritmos clássicos k-means, k-medians e PAM e com o algoritmo GRASP proposto por Nascimento et al. [2010b].

Os resultados obtidos com os experimentos computacionais, mostraram que as heurísticas propostas foram bem sucedidas e confirmaram a hipótese que o GRASP com *Path-Relinking* melhora o GRASP na sua forma padrão. Os algoritmos propostos diminuiram o valor da função objetivo, já que exploraram com maior eficiência o espaço de busca. O aumento dos CRands revelou uma maior coesão nos agrupamentos. A redução do desvio padrão aumentou a robustez do método de agrupamento. A redução do tempo computacional mostrou que os híbridos convergem mais rapidamente. No geral, as variantes *Greedy Randomized Adaptative* e *Mixed* apresentaram os melhores resultados. A variante *Backward* gerou soluções mais diversificadas e, embora tenha apresentado menor robustez quando comparada as demais variantes, apresentou o melhor desempenho na convergência do algoritmo.

Para que o *Path-Relinking* seja efetivo, os experimentos destacaram a importância em se considerar a distância mínima (distância *Hamming*) entre os pares participantes. Nesse sentido, em instâncias com pequena quantidade de grupos (2 a 4), observou-se um comportamento determinístico do algoritmo e o *Path-Relinking* não é aproveitado devido à baixa variabilidade de soluções. A pouca variabilidade também é observada para os casos onde a Busca Local é eficiente para melhoria das soluções.

Com relação as medidas de dissimilaridade, para um mesmo algoritmo, observou-

se que o métrica utilizada tem influência na eficiência e eficácia do algoritmo. Para as instâncias estudadas, as métricas baseadas em distância (euclidiana e *City Block*) apresentaram melhor desempenho que as correlações (Cosine e Pearson).

Em algumas instâncias (ex. DLBCLA, DLBCLB, Yeast) observou-se que não existe uma relação direta entre a minimização da função objetivo e a maximização do CRand, ou seja, a menor função objetivo não necessariamente gera uma solução que obtém o melhor CRand. Assim, como trabalhos futuros, propõem-se estudos que melhor elucidem a relação entre a função objetivo e o CRand, ou outras medidas externas de avaliação. Ainda como trabalhos futuros, estudos sobre o comportamento de *overfitting* [Bubeck & Von Luxburg, 2007], utilização de outros modelos de agrupamento [Rao, 1971] ou abordagem multi-objetivo [Handl & Knowles, 2005] são vistos como temas interessantes a serem pesquisados.

Referências Bibliográficas

Aiex, R.; Resende, M. G. C. & Ribeiro, C. C. (2007). TTT plots: A perl program to create time-to-target plots. *Optimization Letters*, 1(4):355–366.

Al-Sultan, K. (1995). A tabu search approach to the clustering problem. *Pattern Recognition*, 28(9):1443–1451.

Alon, U.; Barkai, N.; Notterman, D.; Gish, K.; Ybarra, S.; Mack, D. & Levine, A. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745.

Alvarez-Valdes, R.; Crespo, E.; Tamarit, J. & Villa, F. (2008). GRASP and path-relinking for project scheduling under partially renewable resources. *European Journal of Operational Research*, 189(3):1153–1170.

An, J. & Chen, Y. (2009). Finding rule groups to classify high dimensional gene expression datasets. *Computational Biology and Chemistry*, 33(1):108–113.

Anderberg, M. (1973). *Cluster analysis for applications*. Academic Press.

Areibi, S. & Vannelli, A. (1997). A GRASP clustering technique for circuit partitioning. In Gu, J. & Pardalos, P., editores, *Satisfiability problems*, volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pp. 711–724. American Mathematical Society.

Arroyo, J.; dos Santos Soares, M. & dos Santos, P. (2010). A GRASP heuristic with path-relinking for a bi-objective p-median problem. In *10th International Conference on Hybrid Intelligent Systems (HIS)*, pp. 97–102.

Bandyopadhyay, S. & Maulik, U. (2002). An evolutionary technique based on K-Means algorithm for optimal clustering in RN. *Information Sciences*, 146(1-4):221–237.

Bastos, L.; Ochi, L. & Macambira, E. (2005). GRASP with path-relinking for the SONET Ring Assignment Problem. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, pp. 239–244. IEEE Computer Society.

Bennett, K. & Mangasarian, O. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1(1):23–34.

Berkhin, P. (2006). A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pp. 25–71.

Bertsimas, D. & Shioda, R. (2007). Classification and regression via integer optimization. *Operations Research*, 55(2):252–271.

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):247–253.

Bishop, C. (2006). *Pattern recognition and machine learning*, volume 4. Springer New York.

Blum, C. & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308.

Borgatti, S. (2011). Distance and correlation. `www.analytictech.com/ba762/handouts/distcorr.htm`. Acessado em 15/03/2011.

Braga, L. (2005). *Introdução à mineração de dados*. Editora E-papers.

Bubeck, S. & Von Luxburg, U. (2007). Overfitting of clustering and how to avoid it. *CCSd - Centre pour la Communication Scientifique Directe*.

Bucene, L. C.; Rodrigues, L. H. A. & Meira, C. A. A. (2002). Mineração de dados climáticos para previsão de geada e deficiência hídrica para as culturas do café e da cana-de-açúcar para o estado de São Paulo. Technical Report ISSN 1677-9274, Embrapa - Empresa Brasileira de Pesquisa Agropecuária.

Butenko, S.; Murphey, R. & Pardalos, P. (2004). *Recent developments in cooperative control and optimization*. Springer Netherlands.

Cano, J.; Cordón, O.; Herrera, F. & Sánchez, L. (2000). A greedy randomized adaptive search procedure to the clustering problem. *International Journal of Intelligent and Fuzzy Systems*, 12:235–242.

Cano, J.; Cordón, O.; Herrera, F. & Sánchez, L. (2002). A greedy randomized adaptive search procedure applied to the clustering problem as an initialization process using k-means as a local search procedure. *Journal of Intelligent and Fuzzy Systems*, 12(3):235–242.

Carreto, C. & Baker, B. (2002). A GRASP interactive approach to the vehicle routing problem with backhauls. *Essays and Surveys in Metaheuristics*, pp. 185–199.

Chaves, A. (2009). *Uma meta-heurística híbrida com busca por agrupamentos aplicada a problemas de otimização combinatória*. PhD thesis, INPE - Instituto Nacional de Pesquisas Espaciais.

Davis, L. & Mitchell, M. (1991). *Handbook of genetic algorithms*. Van Nostrand Reinhold.

Delorme, X.; Gandibleux, X. & Rodriguez, J. (2004). GRASP for set packing problems. *European Journal of Operational Research*, 153(3):564 – 580.

DeRisi, J.; Penland, L.; Brown, P.; Bittner, M.; Meltzer, P.; Ray, M.; Chen, Y.; Su, Y. & Trent, J. (1996). Use of a cDNA microarray to analyse gene expression patterns in human cancer. *Nature Genetics*, 14(4):457–460.

Dharan, S. & Nair, A. (2009). Biclustering of gene expression data using greedy randomized adaptive search procedure. In *TENCON 2008-2008 IEEE Region 10 Conference*, pp. 1–5.

Ding, C. & Dubchak, I. (2001). Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358.

Fayyad, U.; Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54.

Feo, T. A. & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem* 1. *Operations Research Letters*, 8(2):67–71.

Feo, T. A. & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

Feo, T. A.; Resende, M. G. C. & Smith, S. (1994). A greedy randomized adaptative search procedure for maximum independent set. *Operations Research*, 42(5):860–878.

Festa, P.; Pardalos, P.; Pitsoulis, L. & Resende, M. G. C. (2007). GRASP with path-relinking for the weighted MAXSAT problem. *Journal of Experimental Algorithmics (JEA)*, 11:2–4.

Festa, P. & Resende, M. G. C. (2002). GRASP: An annotated bibliography. *Essays and Surveys in Metaheuristics*, pp. 325–367.

Festa, P. & Resende, M. G. C. (2009). An annotated bibliography of GRASP - Part II: Applications. *International Transactions in Operational Research*, 16(2):131–172.

Fisher, R. et al. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

Frawley, W.; Piatetsky-Shapiro, G. & Matheus, C. (1992). Knowledge discovery in databases: An overview. *AI Magazine*, 13(3):57–70.

Gibas, C. & Jambeck, P. (2001). *Developing bioinformatics computer skills*. O'Reilly Media, Inc.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Glover, F. (1996). Tabu search and adaptive memory programming: Advances, applications and challenges. *Interfaces in Computer Science and Operations Research*, 1:1–75.

Glover, F. (2000). Multi-start and strategic oscillation methods - Principles to exploitadaptive memory. *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pp. 1–24.

Glover, F. & Kochenberger, G. (2003). *Handbook of metaheuristics*. Kluwer Academic Publishers.

Glover, F.; Laguna, M. & Martí, R. (2000). Fundamentals of scatter search and path-relinking. *Control and Cybernetics*, 39(3):653–684.

Glover, F. & Marti, R. (2006). Tabu search. *Metaheuristic procedures for training neutral networks*, pp. 53–69.

Glover, F. & Taillard, E. (1993). A user's guide to tabu search. *Annals of Operations Research*, 41(1):3–28.

Grabmeier, J. & Rudolph, A. (2002). Techniques of cluster algorithms in data mining. *Data Mining and Knowledge Discovery*, 6(4):303–360.

Hair Jr, J.; Anderson, R.; Tatham, R.; Black, W.; Sant'Anna, A. & Neto, A. (2006). *Análise multivariada de dados*. Bookman Porto Alegre.

Halmenschlager, C. (2000). Utilização de agentes na descoberta de conhecimento. Master's thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

Han, J. & Kamber, M. (2006). *Data mining: Concepts and techniques*. Morgan Kaufmann.

Hand, D. & Heard, N. (2005). Finding groups in gene expression data. *Journal of Biomedicine and Biotechnology*, 2:215–225.

Handl, J. & Knowles, J. (2005). Exploiting the trade-off: The benefits of multiple objectives in data clustering. *Lecture Notes in Computer Science*, 3410:547–560.

Handl, J. & Knowles, J. (2007). An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76.

Hodge, V. & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126.

Hoos, H. H. & Stützle, T. (1998a). Evaluation las vegas algorithms - Pitfalls and remedies. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 238–245.

Hoos, H. H. & Stützle, T. (1998b). On the empirical evaluation of las vegas algorithms - Position paper. Technical report, Computer Science Department, University of British Columbia.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558.

Horton, P. & Nakai, K. (1997). Better prediction of protein cellular localization sites with the k nearest neighbors classifier. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, volume 5, pp. 147–152.

Hoshida, Y.; Brunet, J.; Tamayo, P.; Golub, T. & Mesirov, J. (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(1):193–218.

Ibaraki, T.; Nonobe, K. & Yagiura, M. (2005). *Metaheuristics: Progress as real problem solvers*. Springer Verlag.

Jain, A.; Murty, M. & Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys (CSUR)*, 31(3):264–323.

Kaufman, L. & Rousseeuw, P. (1987). Clustering by means of medoids. *Statistical data analysis based on the L1 Norm*, pp. 405–416.

Kaufman, L. & Rousseeuw, P. (2005). *Finding groups in data: An introduction to cluster analysis*. WileyBlackwell.

Kernighan, B. & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307.

King, B. (1967). Step-wise clustering procedures. *Journal of the American Statistical Association*, 62(317):86–101.

Kitano, H. (2002). Computational systems biology. *Nature*, 420(6912):206–210.

Laguna, M. & Marti, R. (1999). GRASP and path-relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52.

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.

Lee, L. (1999). Measures of distributional similarity. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 25–32. Association for Computational Linguistics.

Lira, S. & Neto, A. (2008). Coeficientes de correlação para variáveis ordinais e dicotômicas derivados do coeficiente linear de Pearson. *Ciência & Engenharia*, 15(1/2).

Ma, P.; Chan, K.; Yao, X. & Chiu, D. (2006). An evolutionary clustering algorithm for gene expression microarray data analysis. *IEEE Transactions on Evolutionary Computation*, 10(3):296–314.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 281–297. University of California Press.

Mateus, G. R.; Resende, M. G. C. & Silva, R. M. A. (2010). GRASP with path-relinking for the generalized quadratic assignment problem. *Journal of Heuristics*, pp. 1–39.

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.

Mavridou, T.; Pardalos, P.; Pitsoulis, L. & Resende, M. G. C. (1998). A GRASP for the biquadratic assignment problem. *European Journal of Operational Research*, 105(3):613 – 621.

Michalski, R.; Michalski, R. & Tecuci, G. (1994). *Machine learning: A multistrategy approach*. Morgan Kaufmann Publishers.

Milligan, G. (1996). Clustering validation: Results and implications for applied analyses. *Clustering and Classification*, pp. 341–375.

Milligan, G.; Soon, S. & Sokol, L. (2009). The effect of cluster size, dimensionality, and the number of clusters on recovery of true cluster structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(1):40–47.

Monti, S.; Savage, K.; Kutok, J.; Feuerhake, F.; Kurtin, P.; Mihm, M.; Wu, B.; Pasqualucci, L.; Neuberg, D.; Aguiar, R. & Others (2005). Molecular profiling of diffuse large B-cell lymphoma identifies robust subtypes including one characterized by host inflammatory response. *Blood*, 105(5):1851–1861.

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):91–118.

Nakai, K. & Kanehisa, M. (1991). Expert system for predicting protein localization sites in gram-negative bacteria. *Proteins: Structure, Function, and Bioinformatics*, 11(2):95–110.

Nakai, K. & Kanehisa, M. (1992). A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14(4):897–911.

Nascimento, M. (2010). *Metaheurísticas para o problema de agrupamento de dados em grafo*. PhD thesis, USP.

Nascimento, M.; Resende, M. G. C. & Toledo, F. (2010a). GRASP heuristic with path-relinking for the multi-plant capacitated lot sizing problem. *European Journal of Operational Research*, 200(3):747–754.

Nascimento, M.; Toledo, F. & de Carvalho, A. (2010b). Investigation of a new GRASP-based clustering algorithm applied to biological data. *Computers & Operations Research*, 37(8):1381–1388.

Osman, I. & Kelly, J. (1996). *Meta-heuristics: Theory and applications*. Kluwer Academic Publishers Norwell, MA, USA.

Pacheco, J. (2005). A scatter search approach for the minimum sum-of-squares clustering problem. *Computers & Operations Research*, 32(5):1325–1335.

Pardalos, P. & Resende, M. G. C. (1994). A greedy randomized adaptive search procedure for the quadratic assignment problem. In *Quadratic assignment and related problems: DIMACS Workshop, 20-21 May 1993*, pp. 237–261. American Mathematical Society.

Pena, J.; Lozano, J. & Larranaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040.

Phyu, T. (2009). Survey of Classification Techniques in Data Mining. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 1:18–20.

Piatetsky-Shapiro, G. & Frawley, W. (1991). *Knowledge discovery in databases*. AAAI Press.

Prais, M. & Ribeiro, C. C. (2000a). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176.

Prais, M. & Ribeiro, C. C. (2000b). Variação de parâmetros em procedimentos GRASP. *Investigación Operativa*, 9:1–20.

Range, M.; Abreu, N. & Boaventura-Netto, P. (2000). GRASP para o PQA: Um limite de aceitação para soluções iniciais. *Pesquisa Operacional*, 20:45–58.

Rao, M. (1971). Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, 66(335):622–626.

Resende, M. G. & Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In Hillier, F. S.; Gendreau, M. & Potvin, J.-Y., editores, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pp. 283–319. Springer US.

Resende, M. G. C.; Martí, R.; Gallego, M. & Duarte, A. (2010a). GRASP and path-relinking for the max-min diversity problem. *Computers & Operations Research*, 37(3):498–508.

Resende, M. G. C. & Ribeiro, C. (2003). Greedy randomized adaptive search procedures. In Hillier, F. S.; Glover, F. & Kochenberger, G., editores, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pp. 219–249. Springer New York.

Resende, M. G. C. & Ribeiro, C. C. (2005). GRASP with path-relinking: Recent advances and applications. *Metaheuristics: Progress as real problem solvers*, pp. 29–63.

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.

Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e aplicações*. Editora Manole Ltda.

Ribeiro, C. C. & Resende, M. G. C. (2010). Path-Relinking intensification methods for stochastic local search algorithms. *Computers & Operations Research*, 37:498–508.

Ribeiro, M. (2005). Incorporando técnicas de mineração de dados à metaheurística GRASP. Mestrado em ciência da computação, Universidade Federal Fluminense.

Rigoutsos, I.; Floratos, A.; Parida, L.; Gao, Y. & Platt, D. (2000). The emergence of pattern discovery techniques in computational biology. *Metabolic Engineering*, 2(3):159–177.

Ripley, B. (2008). *Pattern recognition and neural networks*. Cambridge Univ Pr.

Rodrigues, F. S. (2009). Métodos de agrupamento na análise de dados de expressão gênica. Mestrado em estatística, Universidade Federal de São Carlos, Centro de Ciências Exatas e de Tecnologia, Departamento de Estatística, São Carlos/SP.

Romesburg, C. (2004). *Cluster analysis for researchers*. Lulu Press North Carolina.

Rosenwald, A.; Wright, G.; Chan, W. C.; Connors, J. M.; Campo, E.; Fisher, R. I.; Gascoyne, R. D.; Muller-Hermelink, H. K.; Smeland, E. B. & Staudt, L. M. (2002). The use of molecular profiling to predict survival after chemotherapy for diffuse Large-B-cell lymphoma. *The New England Journal of Medicine*, 346(25):1937–1947.

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.

Sneath, P. (2005). Numerical taxonomy. *Bergeyś Manual of Systematic Bacteriology*, pp. 39–42.

Spaeth, H. (1980). *Cluster analysis algorithms*. Wiley, New York.

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.

Talbi, E. (2009). *Metaheuristics: From design to implementation*. Wiley Online Library.

Turcotte, M.; Muggleton, S. & Sternberg, M. (2001). The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Machine Learning*, 43(1):81–95.

Tuyttens, D. & Vandaele, A. (2010). Using a greedy random adaptative search procedure to solve the cover printing problem. *Computers & Operations Research*, 37(4):640–648.

Van Laarhoven, P. & Aarts, E. (1987). Simulated annealing: Theory and applications. *Mathematics and Its Applications, D. Reidel, Dordrecht*.

Van't, V.; Laura, J.; Hongyue, D.; Van De Vijver, M.; He, Y.; Hart, A. et al. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536.

Villegas, J.; Prins, C.; Prodhon, C. & Medaglia, A.L. andVelasco, N. (2011). A GRASP with evolutionary path-relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38(9):1319–1334.

Wang, J.; Shapiro, B. & Shasha, D. (1999). *Pattern discovery in biomolecular data: Tools, techniques, and applications*. Oxford University Press, USA.

Ward Jr., J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Webb, A. (1999). *Statistical pattern recognition*. A Hodder Arnold Publication.

West, M.; Blanchette, C.; Dressman, H.; Huang, E.; Ishida, S.; Spang, R.; Zuzan, H.; Olson, J. & Marks, J.R. andNevins, J. (2001). Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 98(20):11462–11467.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Witten, I. & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub.

Wu, X. & Kumar, V. (2009). *The top ten algorithms in data mining*. Chapman & Hall/CRC.

Zapfel, G.; Braune, R. & Bogl, M. (2010). *Metaheuristic search concepts: A tutorial with applications to productionand logistics*. Springer Verlag.

Zhang, C. & Zhang, S. (2002). *Association rule mining: models and algorithms*. Lecture notes in computer science. Springer.

Zhao, Q. & Bhowmick, S. (2003). Sequential pattern mining: A survey. *ITechnical Report CAIS Nayang Technological University Singapore*, pp. 1–26.