Ministério da
Ciência e Tecnologia

sid.inpe.br/mtc-m19/2011/02.17.15.47-TDI

ABORDAGENS HEURÍSTICAS PARA PROBLEMAS DE AGRUPAMENTOS

Dalila Ribeiro Serpa

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Luiz Antonio Nogueira Lorena, e Francisco de Assis Corrêa, aprovada em 25 de fevereiro de 2011

URL do documento original:
<http://urlib.net/ 8JMKD3MGP7W/397C9JE >

INPE
São José dos Campos
2011

PUBLICADO POR :

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

Ministério da
Ciência e Tecnologia

sid.inpe.br/mtc-m19/2011/02.17.15.47-TDI

ABORDAGENS HEURÍSTICAS PARA PROBLEMAS DE AGRUPAMENTOS

Dalila Ribeiro Serpa

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Luiz Antonio Nogueira Lorena, e Francisco de Assis Corrêa, aprovada em 25 de fevereiro de 2011

URL do documento original:
<http://urlib.net/ 8JMKD3MGP7W/397C9JE >

INPE
São José dos Campos
2011

Dados Internacionais de Catalogação na Publicação (CIP)

Serpa, Dalila Ribeiro.

Se67a Abordagens heurísticas para problemas de agrupamentos / Dalila Ribeiro Serpa. – São José dos Campos : INPE, 2011.

xxiv+68 p. ; (sid.inpe.br/mtc-m19/2011/02.17.15.47-TDI)

Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2011.

Orientadores : Drs. Luiz Antonio Nogueira Lorena e Francisco de Assis Corrêa.

1. Agrupamentos 2. Particionamento em cliques . 3. Meta-heurísticas. 4. VNS. 5. ILS. 6. CS. I.Título.

CDU 519:9

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de Mestre **em**

Computação Aplicada

Dra. **Sandra Aparecida Sandri**

Presidente / INPE / SJCampos - SP

Dr. **Luiz Antonio Nogueira Lorena**

Orientador(a) / INPE / São José dos Campos - SP

Dr. **Francisco de Assis Corrêa**

Orientador(a) / UNIP / São José dos Campos - SP

Dr. **José Carlos Becceneri**

Membro da Banca / INPE / SJCampos - SP

Dra. **Mariá Cristina Vasconcelos Nascimento**

Aluno (a): Dalila Ribeiro Serpa

São José dos Campos, 25 de fevereiro de 2011

"Um homem deixa de ser um principiante em qualquer ciência e se torna um mestre quando aprende que vai ser um principiante a vida inteira.".

Robin G. Collingwood

A meus pais José Braz e Maria Aparecida, a meus queridos irmãos Rose, Gustavo, Adriana e Paula, e ao meu esposo André Ricardo

AGRADECIMENTOS

Primeiramente, agradeço a Deus pela sabedoria, paciência, força de vontade e principalmente pela vida, o maior presente que recebi de Suas mãos.

Agradeço a minha família pelo apoio e, em especial, ao meu esposo André Ricardo pela paciência e compreensão.

Agradeço ao Instituto Nacional de Pesquisas Espaciais (INPE), em especial ao curso de Computação Aplicada (CAP), por ter me proporcionado o estudo e desenvolvimento desta dissertação.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro concedido.

Agradeço aos meus orientadores Dr. Luiz Antonio Nogueira Lorena e Dr. Francisco de Assis Corrêa pela paciência e dedicação durante esses dois anos.

Agradeço, em especial, ao Dr. Antonio Augusto Chaves por todo apoio, paciência e por responder a todos os meus emails por mais bobas que fossem as minhas dúvidas.

Agradeço ao Ms. Rogério Galante Negri por me ensinar e ajudar a entender o processamento de imagens, bem como ter me ajudado a utilizar o programa ENVI.

Agradeço, também, aos meus colegas do CCS pelo convívio e pelos momentos de descontração que vivemos juntos.

E a todos que direta ou indiretamente colaboraram para a realização deste trabalho.

RESUMO

Os problemas de agrupamentos surgiram da necessidade de se agrupar dados a fim de entender um objeto ou um fenômeno ainda desconhecidos. O agrupamento é feito com base na similaridade entre os objetos de um conjunto de dados, em que os mais similares ficam no mesmo grupo. Este trabalho propõe três novas abordagens heurísticas para problemas de agrupamentos: a meta-heurística Busca em Vizinhança Variável (VNS, do inglês *Variable Neighborhood Search*), a meta-heurística Busca Local Iterativa (ILS, do ingles *(*Iterated Local Search) e o método híbrido Busca por Agrupamentos (CS, do inglês *Clustering Search*). O VNS é caracterizado por realizar buscas em vizinhanças distantes. O ILS realiza perturbações em uma solução gerando novas soluções de partida para a busca local. O CS, chamado de método híbrido por utilizar uma combinação de meta-heurísticas com busca local, é caracterizado por realizar buscas em regiões promissoras à melhoria da solução. Neste trabalho, estes algoritmos utilizarão a abordagem de particionamento em cliques para realizar o *clustering*. Os *clusters* obtidos pelos algoritmos serão avaliados por dois índices de validação externa: Rand e Corrected Rand. Além disso, uma pequena aplicação em classificação de imagens será apresentada. E por fim, os resultados obtidos serão comparados com outros algoritmos da literatura.

HEURISTICS APPROACHES FOR CLUSTERING PROBLEMS

ABSTRACT

The clustering problems arose from the need to group data in order to understand an object or a phenomenon still unknown. Data clustering is based on similarity between objects of a data set, where the most similar objects are in the same group. This work proposes three new heuristics approaches to clustering problems: the *Variable Neighborhood Search* (VNS) metaheuristic, the *Iterated Local Search* (ILS) metaheuristic and the hybrid method called *Clustering Search*(CS). VNS is characterized by performing searches in a distant neighborhoods. ILS performs a perturbation in a solution, generating new starting solutions to the local search. CS, called hybrid method because it uses a combination of metaheuristics with local search, is characterized by performing searches in promising regions to the solution improvement. In this work, these algorithms will use the clique partitioning approach to perform the clustering. The clusters obtained by these algorithms will be evaluated by two external validation indexes: Rand and Corrected Rand. Moreover, a little application in image classification will be presented. At the end, the results obtained will be compared with others algorithms in the literature.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

VNS	–	Variable Neighborhood Search
ILS	–	Iterated Local Search
CRand	–	Correct (Adjusted) Rand Index
Rand	–	Rand Index
CS	–	Clustering Search
ECS	–	Evolutionary Clustering Search
GRASP	–	Greedy Randomized Adaptive Search Procedure
PAM	–	Particionamento em torno de medianas
ILS	–	Iterated Local Search
VND	–	Descida em Vizinhança Variável
PR	–	Path Relinking

LISTA DE SÍMBOLOS

N – número de objetos de um conjunto de dados
L – número de atributos de um objeto
K – número de *clusters* existentes em um conjunto de dados
k – número que identifica um *cluster* ($k \in [1, K]$)
x_i – objeto i de um conjunto de dados
ind_{ij} – índice de proximidade entre os objetos i e j
a_{il} – *l-ésimo* atributo do objeto i
d_{ij} – distância/dissimilaridade entre os objetos x_i e x_j
sim_{ij} – índice de similaridade entre os objetos x_i e x_j
B_{ij} – Correlação de Pearson entre os objetos x_i e x_j
D_{ij} – Correlação de Pearson não-centralizado entre os objetos x_i e x_j
p_{max} – número de estruturas diferentes de vizinhança do VNS
$P_{N \times N}$ – matriz de proximidade
G – grafo de proximidade
M – conjunto de nós do grafo G
H – conjunto de arestas do grafo G
m_i – nó i do grafo G
h_{ij} – aresta que liga o nó m_i ao m_j do grafo G
$P1$ – primeira partição para o cálculo do Rand e CRand
$P2$ – segunda partição para o cálculo do Rand e CRand
ee – o número de pares de objetos que estão no mesmo *cluster*
dd – o número de pares de objetos que estão em *clusters* diferentes
ed – o número de pares de objetos que estão em *clusters* iguais e em *clusters* diferentes
de – o número de pares de objetos que estão em *clusters* diferentes e em *clusters* iguais
f – função objetivo
f_{pmed} – função objetivo do problema de p-medianas
V – vizinhança em uma busca local
med – medianas de uma solução
s – solução corrente
s' – solução vizinha
s'' – solução ótima local
s''' – solução ótima local de uma região do ILS
s^* – melhor solução encontrada
s_i – *i-ésima* posição da solução s que representa o *i-ésimo* objeto do conjunto de dados
s_j – *j-ésima* posição da solução s que representa o

		j-ésimo objeto do conjunto de dados
s_t	–	solução gerada pela meta-heurística no CS
C_j	–	*j-ésimo* grupo do CS
c_j	–	centro do *j-ésimo* grupo do CS
c_j^*	–	melhor solução encontrada para o *j-ésimo* grupo
c_j''	–	melhor vizinho de c_j
v_j	–	volume do *j-ésimo* grupo do CS
λ	–	limitante do volume do CS
r_j	–	índice de ineficácia do *j-ésimo* grupo do CS
r_{max}	–	limite do índice de ineficácia do CS
$d(s_t, c_j)$	–	medida de distância entre a solução s_t e o centro c_j
$d(s_t, c)$	–	medida de distância entre a solução s_t e cada centro c
R	–	valor do índice Rand
CR	–	valor do índice CRand
k_i	–	*i-ésimo cluster* de uma partição
k_j	–	*j-ésimo cluster* de uma partição
n_{ij}	–	número de objetos comuns aos *clusters* k_i de $P1$ e k_j de $P2$
$n_{i.}$	–	número de objetos no *cluster* k_i de $P1$
$n_{.j}$	–	número de objetos no *cluster* k_j de $P2$
K_{P1}	–	quantidade de *clusters* existentes em $P1$
K_{P2}	–	quantidade de *clusters* existentes em $P2$
β	–	porcentagem da perturbação do ILS
S^*	–	subespaço formado por ótimos locais
Q	–	conjunto de soluções aleatórias geradas no método para criar grupos
W	–	conjunto de soluções diversas que constituem os centros dos grupos iniciais
$Rd_{q \times q}$	–	matriz de diversidade entre as soluções de Q
Rd_{soma}	–	soma das diversidades entre um candidato e todas as soluções de W
$s_{inicial}$	–	solução inicial do PR
s_{guia}	–	solução final do PR
$med_{inicial}$	–	medianas da solução inicial do PR
med_{guia}	–	medianas da solução final do PR
mv	–	movimento do PR
mv^*	–	melhor movimento em uma iteração do PR
gap	–	amplitude da diferença entre soluções geradas em diferentes execuções
s_{min}	–	solução com menor função objetivo para cálculo do *gap*
s_{max}	–	solução com maior função objetivo para cálculo do *gap*

SUMÁRIO

Pág.

1 INTRODUÇÃO

Pessoas do mundo todo se deparam com milhares de informações a todo instante, seja na televisão, na rua, no jornal, na internet, etc. Muitas vezes essas informações não são claras, dependendo do assunto que tratam. O fato de não proporcionarem entendimento imediato obriga as pessoas a buscarem mais dados e analisá-los, seja separando os que interessam ou organizando-os de forma a compreender e colocar em prática o que as informações trazem de aplicação prática. Da mesma forma acontece com pesquisadores de diferentes áreas de estudo que coletam grande número de informações e em muitos casos não as compreendem. Por isso, as representam como dados e utilizam de mecanismos para organizá-los em grupos como uma forma de descobrir conjuntos de categorias entre eles e assim entender um objeto ou um fenômeno (XU; WUNSCH, 2005). A necessidade de se agrupar ou classificar conjuntos de dados deu origem aos problemas de agrupamentos.

O agrupamento de dados é utilizado em diversas áreas como: biologia, medicina, sensoriamento remoto, *marketing*, visão computacional. Uma área que tem se beneficiado recentemente das pesquisas de análise de agrupamentos de dados é a bioinformática (KOCHENBERGER et al., 2005; NASCIMENTO et al., 2010).

Como exemplo de problema de agrupamentos, imagine um pesquisador estudando células cancerígenas. Sabendo que elas podem caracterizar cânceres malignos ou benignos, ele monta um conjunto de dados com as propriedades de todas as células e organiza-o em dois grupos, em que as células que compartilham das mesmas propriedades ficam juntas em um mesmo grupo. Assim, numa interpretação com base em conhecimento prévio da área, o pesquisador consegue entender qual grupo corresponde a tal tipo de câncer. Com isso, futuramente, ao coletar dados de células pertencentes a um câncer desconhecido, o pesquisador faz uma comparação com os dados das células já agrupadas, verificando a qual grupo o novo câncer pertence, e assim o classifica como maligno ou benigno.

O agrupamento descrito acima é facilmente feito por humanos quando se tem poucos dados. Mas, quando o conjunto de dados é extenso, há a necessidade de se aplicar um algoritmo para diminuir o tempo e aumentar eficiência do agrupamento. Técnicas ou algoritmos de agrupamento são importantes ferramentas para tratar esses problemas. Um algoritmo de agrupamento visa encontrar grupos (*clusters*) entre os objetos de um conjunto de dados de forma a manter os mais similares no mesmo *cluster*. Essa

similaridade pode ser representada por alguma característica que seja comum entre os objetos do mesmo *cluster*.

O processo de agrupamento é composto por várias etapas que devem ser cuidadosamente seguidas, pois em cada uma delas são feitas definições importantes para o resultado final. A escolha de um algoritmo de agrupamento é uma etapa importante desse processo, visto que existem muitos tipos de algoritmos para este fim, os quais se diferem em vários pontos que vão desde o tipo de agrupamento até mesmo a abordagem que utilizam para tal função. Um dos algoritmos de agrupamento mais conhecidos é o k-Médias (MACQUEEN, 1967), que usa uma abordagem estocástica para realizar o agrupamento, ou seja, é um algoritmo baseado em um modelo matemático com o objetivo de minimizar ou maximizar o valor de uma função. No caso do k-Médias, a função objetivo é a minimização do erro quadrático entre os objetos e seus centros de *cluster*. Em outras palavras, a função objetivo do k-Médias é a minimização da soma das distâncias entre os objetos e seus centros de *cluster*.

Recentemente, as meta-heurísticas vêm sendo estudadas na resolução de problemas de agrupamentos. Meta-heurísticas são algoritmos de busca que visam encontrar a melhor solução dentre muitas existentes e são baseados em minimizar ou maximizar uma função objetivo (abordagem estocástica, assim como k-Médias). Dois trabalhos são destaque nessa área: Nascimento et al. (2010) e Chang et al. (2009). Em Nascimento et al. (2010) os autores propoem um modelo matemático baseado em particionamento em cliques, aplicam a meta-heurística GRASP (FEO; RESENDE, 1995) na resolução do agrupamento e comparam os resultados com outros algoritmos de agrupamento. Já em Chang et al. (2009), é proposto um rearranjo de genes com a meta-heurística Algoritmo Genético (GOLDBERG, 1989) para o k-Médias, além disso, os autores comparam os resultados com outros algoritmos de agrupamento.

O objetivo deste trabalho é aplicar três meta-heurísticas a problemas de agrupamentos. A primeira é a meta-heurística VNS (MLADENOVIć; HANSEN, 1997) caracterizada por realizar buscas em vizinhanças distantes. Já a segunda, é a meta-heurística ILS caracterizada por focar sua busca em um subespaço S^* formado por soluções que são ótimos locais de determinados métodos de otimização. E a terceira é a meta-heurística híbrida Busca por Agrupamentos (CHAVES, 2009), caracterizada por realizar buscas em regiões promissoras por meio do agrupamento de soluções. Além de utilizar a abordagem heurística, neste trabalho o agrupamento será feito por meio do particionamento em cliques. Para realizar o particionamento em cliques é preciso

que os dados sejam representados como um grafo completo e assim particioná-lo em K cliques, em que o número de cliques é igual ao número de *clusters* existentes. A explanação sobre essas abordagens será vista no Capítulo 2. Os testes das meta-heurísticas serão feitos com bases de dados reais encontradas na literatura. Os *clusters* obtidos com as meta-heurísticas serão avaliados por dois índices de validação: Rand (RAND, 1971) e Corrected Rand (HUBERT; ARABIE, 1985). Os resultados do agrupamento serão comparados com dois trabalhos dessa mesma área. Além da aplicação em conjuntos de dados, será apresentada uma pequena aplicação em classificação de imagem, a qual utilizará uma amostra de uma imagem coletada pelo satélite Landsat-5 e obtida através do catálogo de imagens do INPE.

1.1 Organização da dissertação

O trabalho está organizado da seguinte forma: o Capítulo 2 descreve o processo de agrupamento, suas etapas e definições. O Capítulo 3 traz uma revisão bibliográfica de alguns trabalhos que utilizaram meta-heurísticas na solução de agrupamentos. O Capítulo 4 faz uma introdução ao termo meta-heurística e traz algumas definições importantes referente aos problemas de agrupamentos. No Capítulo 5 estão explicadas as duas meta-heurísticas VNS e ILS. No Capítulo 6 é descrita a meta-heurística híbrida Busca por Agrupamentos. O Capítulo 7 traz uma análise de robustez e eficiência das meta-heurísticas abordadas neste trabalho, além de mostrar os resultados de comparações com outros trabalhos da literatura e uma pequena aplicação em classificação de imagens. E por último, o Capítulo 8 traz algumas considerações finais, sugestões de melhorias e propostas para trabalhos futuros.

2 PROBLEMA DE AGRUPAMENTOS

Neste capítulo é feita uma descrição do problema abordado neste trabalho. A solução de um problema de agrupamentos envolve várias etapas. Em cada etapa desse processo são feitas definições importantes para o resultado final. Este trabalho utilizará algumas definições que serão mostradas juntamente à explicação das etapas do processo de agrupamento.

2.1 Definições

Os problemas de agrupamentos são caracterizados pela necessidade de se agrupar objetos de um conjunto de dados de forma a manter os mais semelhantes no mesmo (*cluster*). Essa semelhança é representada por alguma característica que seja comum entre os objetos do *cluster*.
Não existe uma única definição para *cluster*, apesar da ideia ser intuitiva. Algumas definições comuns são encontradas em Barbara (2000), são elas:

***cluster* bem separado:** um *cluster* é um conjunto de pontos tal que qualquer ponto em um determinado *cluster* está mais próximo (ou mais similar) a cada ponto nesse *cluster* do que a qualquer ponto não pertencente a ele;

***cluster* baseado em centro:** um *cluster* é um conjunto de pontos, tal que qualquer ponto em um dado *cluster* está mais próximo (ou mais similar) ao centro desse *cluster* do que ao centro de qualquer outro *cluster*;

***cluster* contínuo (vizinho mais próximo ou agrupamento transitivo):** um *cluster* é um conjunto de pontos tal que qualquer ponto em um dado *cluster* está mais próximo (ou mais similar) a um ou mais pontos nesse *cluster* do que a qualquer ponto que não pertence a ele;

***cluster* baseado em densidade:** um *cluster* é uma região densa de pontos, separada de outras regiões de alta densidade por regiões de baixa densidade;

***cluster* baseado em similaridade:** um *cluster* é um conjunto de pontos que são similares, enquanto pontos em *clusters* diferentes não são similares.

Para se realizar o agrupamento é preciso seguir algumas etapas que vão desde a preparação dos dados até a interpretação dos *clusters* obtidos. Esse processo é explicado a seguir.

2.2 Processo de agrupamento

O processo de agrupamento é realizado por meio de uma sequência de etapas a serem seguidas. Essas etapas compreendem: a preparação dos dados, a definição de uma medida de proximidade, a escolha de um algoritmo de agrupamento, a validação do agrupamento feito pelo algoritmo e por fim a interpretação dos *clusters* obtidos.
As subseções posteriores trazem a descrição de cada etapa de acordo com as informações obtidas em Jain et al. (1999), Hansen e Jaumard (1997), Xu e Wunsch (2005).

2.2.1 Preparação dos dados

Os objetos a serem agrupados podem representar um objeto físico, como uma bola ou um fenômeno, como uma doença. Esses objetos também são comumente chamados de instâncias, pontos, padrões ou amostras. A preparação dos dados envolve vários aspectos relacionados à forma de representação e ao pré-processamento pelo qual eles passam antes de serem agrupados. As formas de representação se dividem em três e são descritas a seguir.

- **Matriz de dados**

  Na matriz de dados, os objetos são representados por vetores em um espaço multidimensional, onde cada dimensão representa um atributo (variável, medida) que descreve o objeto (BARBARA, 2000). Assim, um conjunto de dados é representado por uma matriz $N \times L$, sendo N o número total de objetos do conjunto e L o total de atributos de um objeto. Com isso, uma linha e uma coluna da matriz correspondem, respectivamente, a um objeto e um atributo desse objeto (JAIN; DUBES, 1988).

- **Matriz de proximidade**

  A matriz de proximidade representa os dados por meio das similaridades ou dissimilaridades entre os objetos (BARBARA, 2000). Considere uma matriz de proximidade P_{NxN}, em que N é o número total de objetos. Sejam x_i e x_j o *i-ésimo* e *j-ésimo* objetos de um conjunto de dados, então a *i-ésima* linha e a *j-ésima* coluna da matriz P receberá o valor da

similaridade ou dissimilaridade entre os objetos x_i e x_j.

- **Grafo de proximidade**

  A partir da matriz de proximidade é possível definir o grafo de proximidade. Considere um grafo $G = (M, H)$, em que M é o conjunto de nós e H o conjunto de arestas não direcionadas que ligam cada nó a todos os outros nós, caracterizando um grafo completo. Cada nó desse grafo representa um objeto do conjunto de dados e cada aresta recebe um valor de peso que corresponde ao valor da proximidade entre os objetos. Por exemplo, sejam m_i e m_j dois nós de G que representam os dois objetos x_i e x_j, então o peso da aresta h_{ij} que liga os nós m_i e m_j será o valor da proximidade entre os objetos x_i e x_j, exatamente o valor encontrado na *i-ésima* linha e *j-ésima* coluna da matriz de proximidade.

O pré-processamento dos dados pode envolver normalizações, conversões de tipos, redução de atributos por meio de seleção ou extração de características (JAIN et al., 1999), entre outras transformações necessárias.

2.2.2 Medidas de proximidade

Nesta etapa define-se a medida de proximidade apropriada ao agrupamento em questão. A medida escolhida deve levar em consideração os tipos e as escalas dos atributos que definem os objetos e também as propriedades dos dados mais relevantes ao pesquisador. Em geral, as medidas de proximidade consideram que todos os atributos são importantes.
De acordo com Barbara (2000) e Jain e Dubes (1988) os tipos de atributos são os seguintes:

- **Binários:** compostos por apenas dois valores, por exemplo, 0 ou 1, sim ou não.
- **Discretos:** compostos por um número finito de valores, por exemplo, meses do ano.
- **Contínuos:** compostos por um número infinito de valores.

E as escalas dos atributos são:

a) **Qualitativos**

- **Nominal:** os valores são simples nomes, por exemplo, cores.
- **Ordinal:** os valores representam uma ordem, por exemplo, (1,2,3) ou (10, 20, 30)

b) **Quantitativos**

- **Intervalar:** uma unidade de medida existe, ou seja, a diferença entre os valores é significativa. Por exemplo, a medida de 90º Fahrenheit é diferente de 90º Celsius.
- **Proporcional:** a escala tem um zero absoluto, de modo que as proporções sejam significativas. Por exemplo, a distância entre duas cidades pode ser medida em quilômetros, metros ou centímetros que a proporção será a mesma, ou seja, se uma pessoa for dirigindo um carro de uma cidade a outra, percorrerá a mesma distância.

Sabendo, então, os tipos e escalas dos atributos, é possível escolher uma ou mais medidas de proximidade.
Segundo Jain e Dubes (1988) um índice de proximidade deve satisfazer algumas propriedades. Seja ind_{ij} o índice de proximidade entre os objetos x_i e x_j, suas propriedades são:

a) $ind_{ii} = 0, \quad \forall\, i$, esta propriedade caracteriza uma medida de dissimilaridade;

b) $ind_{ii} > ind_{ij} \quad \forall\, i, j$, esta propriedade caracteriza uma medida de similaridade;

c) $ind_{ij} = ind_{ji} \quad \forall\, i, j$, garante a simetria entre os objetos;

d) $ind_{ij} \geq 0, \quad \forall\, i, j$, garante que um índice de proximidade seja sempre positivo;

Para que uma medida de proximidade seja considerada uma medida de distância, ela também deve satisfazer a mais duas propriedades:

e) $ind_{ij} = 0$, se e somente se $i = j$

f) $ind_{ij} \leq ind_{i,k} + ind_{k,j} \quad \forall\, i, j, k$, o que garante a desigualdade triangular, ou seja, a soma de dois lados do triângulo formado entre três objetos não pode ser menor que um dos lados.

Neste trabalho, serão utilizadas quatro medidas de proximidade: duas medidas de distância e duas medidas de correlação, descritas a seguir.

2.2.2.1 Medidas de distância

As distâncias Euclidiana e City-block (ou Manhattan) são as mais comuns (BARBARA, 2000; JAIN; DUBES, 1988; XU; WUNSCH, 2005). Essas métricas consideram cada atributo como uma dimensão de um espaço multidimensional e cada ponto desse espaço, um objeto. As métricas de distância também são consideradas medidas de dissimilaridade. Sejam a_{il} e L respectivamente o *l-ésimo* atributo do objeto i e o número total de atributos de um objeto, então as dissimilaridades (distâncias) d_{ij} são definidas por:

- **Euclidiana:** é a distância geométrica Euclidiana entre os objetos.

$$d_{ij} = \sqrt{\sum_{l=1}^{L} (a_{il} - a_{jl})^2} \tag{2.1}$$

- **City-block ou Manhattan:** é a soma das diferenças absolutas entre os atributos de dois objetos.

$$d_{ij} = (\sum_{l=1}^{L} |a_{il} - a_{jl}|) \tag{2.2}$$

2.2.2.2 Medidas de correlação

Um coeficiente de correlação também pode ser uma medida de proximidade. As medidas de correlação são consideradas medidas de similaridade. E por não satisfazerem a propriedade de desigualdade triangular são consideradas semimétricas (METZ, 2006). Neste trabalho serão utilizados dois coeficientes de correlação: Pearson e Pearson não-centralizado, que são descritos a seguir.

- **Correlação de Pearson**

  Esse coeficiente mede o grau e a direção de correlação entre dois objetos. Ele assume valores entre -1 e 1. Quando igual a 1 gera uma associação positiva (relação forte ou perfeita) entre duas variáveis. Se igual a -1 gera relação linear perfeita negativa (correlação inversa), ou seja, se uma variável aumenta a outra diminui. Se igual a zero significa que não há dependência linear entre as variáveis. A formulação dela é dada por:

$$B_{ij} = \frac{L\sum a_{il}a_{jl} - \sum a_{il}a_{jl}}{\sqrt{L\sum a_{il}^2 - (\sum a_{il})^2}\sqrt{L\sum a_{jl}^2 - (\sum a_{jl})^2}} \tag{2.3}$$

  Para este trabalho, as medidas utilizadas serão de dissimilaridade. Por isto, assim como descrito em Everitt et al. (2001), para transformar essa similaridade sim_{ij} em dissimilaridade d_{ij} calcula-se $d_{ij} = 1 - sim_{ij}$. Além disso, para trabalhar numa escala entre [0,1] a dissimilaridade entre os objetos, para Correlação de Pearson, será dada por: $d_{ij} = 1 - |B_{ij}|$.

- **Correlação de Pearson não-centralizado**

  Esse coeficiente mede a correlação geométrica definida pelo ângulo entre dois objetos. O valor deste coeficiente varia entre -1 e 1. Assim, quando a correlação entre um par de objetos é igual a 1 significa que ângulo entre esses objetos é 0^o, agora quando for igual -1 o ângulo entre esses objetos é 180^o. Esta correlação é equivalente ao cosseno do ângulo entre dois vetores. A formulação dessa correlação é a seguinte:

$$D_{ij} = \frac{\sum_{l=1}^{L} a_{il}a_{jl}}{\sqrt{\sum_{l=1}^{L} a_{il}^2}\sqrt{\sum_{l=1}^{L} a_{jl}^2}} \tag{2.4}$$

  Assim como a Correlação de Pearson descrita anteriormente, a dissimilaridade entre os objetos, para Correlação de Pearson não-centralizado utilizada neste trabalho, é dada por $d_{ij} = 1 - |D_{ij}|$.

Em Metz (2006), Xu e Wunsch (2005) são encontrados mais detalhes sobre essas e outras medidas de proximidade.

2.2.3 Agrupamento

Esta etapa compreende a escolha de um algoritmo de agrupamento adequado ao problema.
O agrupamento de dados é um tipo especial de classificação de dados que também pode seguir três gêneros: exclusivos ou não exclusivos, supervisionados ou não supervisonados, hierárquicos ou particionais (JAIN; DUBES, 1988). A Figura 2.1 ilustra esses gêneros e, logo abaixo, encontra-se a explicação de cada um deles.

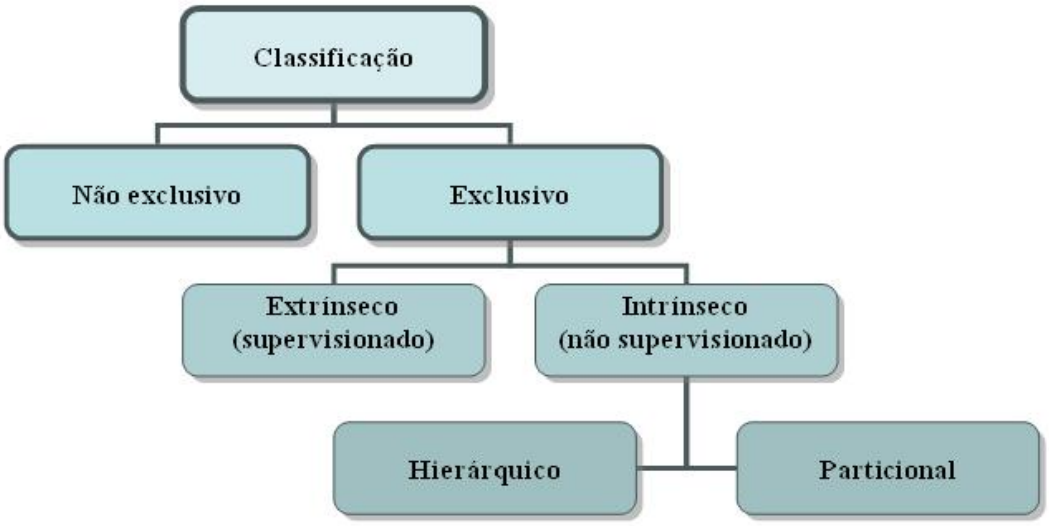

Figura 2.1 - Gêneros de Classificação
Fonte: Jain e Dubes (1988)

- **Exclusivo x Não exclusivo:** na classificação exclusiva, ao particionar os objetos de um conjunto de dados, cada objeto pertencerá a exatamente um *cluster*. Ao contrário, a classificação não exclusiva permite que um mesmo objeto pertença a vários *clusters*. Um exemplo de algoritmo de classificação não exclusiva são aqueles que usam o método de agrupamento nebuloso (do inglês *fuzzy clustering*) (SILVA, 2003).

- **Intrínseco x Extrínseco:** a classificação intrínseca, também chamada de não supervisionada, utiliza apenas a matriz de proximidade e não necessita de nenhum conhecimento prévio sobre a classificação dos objetos. Ao

contrário, a classificação extrínseca pode utilizar tanto a matriz de proximidade como objetos classificados a *priori*. É como se a extrínseca, além de saber a proximidade entre os objetos, também tivesse um "professor" guiando o agrupamento, enquanto a intríseca baseia-se apenas na proximidade entre os objetos.

- **Hierárquico x Particional:** esta classificação envolve a estrutura imposta aos dados. A classificação hierárquica gera uma sequência de partições aninhadas. Já a classificação particional faz um único particionamento dos dados. A Figura 2.2 ilustra as duas classificações: hierárquica e particional.

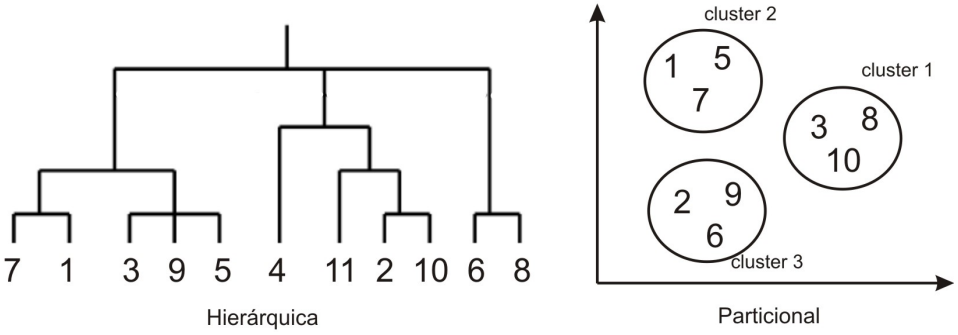

Figura 2.2 - Classificação Hierárquica e Particional

Neste trabalho, os *clusters* serão feitos de forma exclusiva, intrínseca (não supervisionada) e particional. Além dessas características, os algoritmos utilizados terão outras duas características importantes para seu funcionamento: serão estocásticos (heurísticos) e farão o particionamento dos dados em cliques. Abaixo encontram-se os detalhes de cada um dos algoritmos utilizados neste trabalho:

- **heurísticos:** se trata da transformação de um problema de agrupamento em um problema de otimização, ou seja, o algoritmo de agrupamento é baseado em uma formulação matemática em que o objetivo é maximizar ou minimizar o valor de uma função (BARBARA, 2000; JAIN et al., 1999). Um exemplo clássico é o algoritmo k-Médias (MACQUEEN, 1967), que faz o agrupamento tentando minimizar a soma das distâncias entre os objetos e seus centros de *cluster*. Recentemente, as meta-heurísticas vêm

sendo aplicadas a problemas de agrupamentos. O funcionamento básico delas é realizar buscas num espaço de possíveis soluções com intuito de encontrar a melhor solução de acordo com uma função objetivo. Assim como k-Médias, elas também não garantem a convergência ao ótimo global, mas encontram boas soluções em tempo computacional aceitável. Neste trabalho, três meta-heurísticas serão aplicadas a problemas de agrupamentos e são detalhadas nos Capítulo 5 e 6. Os trabalhos que também aplicaram métodos heurísticos a esses problemas são comentados no Capítulo 3.

- **particionamento em cliques:** a ideia é representar os dados como um grafo de proximidade e particioná-lo em K cliques (subgrafos completos), sendo K o número de *clusters* existentes. Assim, ao particionar o grafo em K cliques encontram-se os K *clusters*. Esta abordagem também foi trabalhada em Amorim et al. (1992), Mehrotra e Trick (1998) e Kochenberger et al. (2005).

Depois de agrupar os dados, é necessário avaliar os *clusters* obtidos. Esse processo de validação é descrito na próxima subseção.

2.2.4 Validação

A validação em um processo de agrupamento é responsável por avaliar os *clusters* obtidos, determinando se os resultados são ou não significativos para o conjunto de dados analisado. Segundo Jain et al. (1999), uma estrutura de agrupamento só é válida caso não tenha ocorrido por acaso, já que um algoritmo de agrupamento encontrará *clusters*, independente de haver similaridade ou não entre os objetos.

Em geral, a validação é feita com base em índices estatísticos que julgam, de maneira quantitativa e objetiva, o mérito das estruturas encontradas (JAIN; DUBES, 1988). Um índice é um valor que expressa a qualidade do agrupamento.

Um índice de validação deve obedecer a um critério de validação, o qual define a estratégia utilizada para validar o agrupamento, visto que o índice é apenas a estatística pela qual a validade é testada (JAIN; DUBES, 1988). Os critérios de validação de agrupamentos podem ser divididos em 3 tipos (JAIN; DUBES, 1988; HALKIDI et al., 2002a). São eles:

- **critérios internos:** medem a qualidade do agrupamento apenas com base nos dados originais, seja matriz de dados/proximidade ou grafo de proximidade.

- **critérios externos:** medem a qualidade do agrupamento partindo de duas estruturas: uma é a estrutura gerada pelo algoritmo de agrupamento e a outra é a estrutura que contém os *clusters* reais construídos por um especialista da área com base em conhecimento prévio.

- **critérios relativos:** nesses critérios, diversos *clusterings* são comparados a fim de decidir qual deles é o melhor em algum aspecto. Pode ser usado para comparar *clusterings* gerados por diferentes algoritmos.

Neste trabalho, serão utilizados os critérios externos para validação. Os *clusters* gerados pelas meta-heurísticas serão avaliados pelos índices Rand e Corrected Rand (CRand) propostos, respectivamente, por Rand (1971) e Hubert e Arabie (1985). O índice Rand determina a similaridade entre duas partições (uma é a gerada pelo algoritmo de agrupamento e a outra é a que contém os *clusters* reais) por meio de penalizações nas associações incorretas de pares de objetos aos *clusters*. Por exemplo, se os objetos x_1 e x_2 são associados a um mesmo *cluster* em uma partição e a *clusters* diferentes em outra partição, isso resulta em uma penalização, fazendo um decréscimo no valor do índice. Sejam $P1$ e $P2$ duas partições, o Rand é calculado da seguinte forma (HALKIDI et al., 2002a):

*ee***:** o número de pares de objetos que estão no mesmo *cluster* em $P1$ e $P2$;

*dd***:** o número de pares de objetos que estão em *clusters* diferentes em $P1$ e $P2$;

*ed***:** o número de pares de objetos que estão em *clusters* iguais em $P1$ e em *clusters* diferentes em $P2$;

*de***:** o número de pares de objetos que estão em *clusters* diferentes em $P1$ e em *clusters* iguais em $P2$;

Assim, o índice Rand, R, é dado por:

$$R = \frac{ee + dd}{ee + dd + ed + de} \tag{2.5}$$

O valor de R varia entre [0,1], sendo que quanto mais próximo de 0, mais diferentes são as duas partições (os *clusters* encontrados em uma são diferentes dos encontrados na outra). E quanto mais próximo de 1, mais semelhantes são as partições (os *clusters* encontrados em uma são mais semelhantes aos encontrados na outra), constatando a combinação perfeita nas duas partições. O índice Rand não é corretamente encontrado quando as partições têm diferentes números de *clusters* (JAIN; DUBES, 1988).

O CRand é uma normalização do índice Rand (HUBERT; ARABIE, 1985; FACELI et al., 2005). O resultado do CRand varia entre [-1,1], sendo que valores negativos são menos significativos (HUBERT; ARABIE, 1985). A diferença é que o CRand não é sensível ao número de *clusters*, ou seja, pode ser aplicado a partições que tenham números de *clusters* diferentes.
Seja $P1$ e $P2$ duas partições, n_{ij} o número de objetos comuns aos *clusters* k_i de $P1$ e k_j de $P2$, $n_{i.}$ o número de objetos no *cluster* k_i de $P1$, $n_{.j}$ o número de objetos no *cluster* k_j de $P2$, K_{P1} e K_{P2} respectivamente a quantidade de *clusters* existentes nas partições $P1$ e $P2$, o valor de CRand CR é dado por:

$$CR = \frac{\sum_{i=1}^{K_{P1}}\sum_{j=1}^{K_{P2}}\binom{n_{ij}}{2} - \left[\sum_{i=1}^{K_{P1}}\binom{n_{i.}}{2}\sum_{j=1}^{K_{P2}}\binom{n_{.j}}{2}\right]/\binom{N}{2}}{\left[\sum_{i=1}^{K_{P1}}\binom{n_{i.}}{2} + \sum_{i=1}^{K_{P2}}\binom{n_{.j}}{2}\right]/2 - \left[\sum_{i=1}^{K_{P1}}\binom{n_{i.}}{2} + \sum_{i=1}^{K_{P2}}\binom{n_{.j}}{2}\right]/\binom{N}{2}} \tag{2.6}$$

O CR é calculado entre a partição gerada pelo algoritmo de agrupamento e a partição que contem a classificação real dos objetos.

É importante ressaltar que além do Rand e CRand existem outros índices (inclusive com outros critérios de validação) e podem ser encontrados em Halkidi et al. (2002a), Halkidi et al. (2002b), Metz (2006) e Jain e Dubes (1988).

2.2.5 Interpretação

A interpretação compreende o processo de examinar cada *cluster*, a fim de encontrar significados relacionados a um domínio de aplicação. Nesta etapa, há a necessidade da participação de um especialista do domínio para avaliar o agrupamento e descobrir algum significado prático, considerando os objetos que compõem cada *cluster*.

3 REVISÃO BIBLIOGRÁFICA

Dentre os diversos trabalhos publicados na área de agrupamento estão aqueles que utilizam meta-heurísticas e aqueles que utilizam particionamento em cliques. Alguns trabalhos aplicam uma junção dessas duas abordagens: usam uma meta-heurística baseada num modelo matemático desenvolvido para particionamento em cliques. Este capítulo apresenta alguns trabalhos que usam essas abordagens.

Kochenberger et al. (2005) propõem um modelo matemático para solução de problemas de agrupamentos por meio do particionamento dos dados em cliques. O modelo matemático é testado por meio da meta-heurística Busca Tabu (GLOVER, 1986). Os dados utilizados nos testes foram 6 conjuntos de dados referentes à expressão gênica.

Nascimento et al. (2010) propõem um modelo matemático para solução de problemas de agrupamentos por meio do particionamento dos dados em cliques. Com base nesse modelo proposto, a meta-heurística GRASP (do inglês *Greedy Randomized Adaptive Search Procedures*) (FEO; RESENDE, 1995) é aplicada a esses problemas. Os testes do algoritmo são feitos utilizando-se de dez conjuntos de dados biológicos. Além disso, os autores geraram *clusters* com outros três algoritmos: k-Médias (MACQUEEN, 1967), k-Medianas (KAUFMAN; ROUSSEEUW, 1989) e Particionamento em torno de medianas (PAM, do inglês *Partitioning Around Medoids*) (KAUFMAN; ROUSSEEUW, 1989). Os resultados do agrupamento dos quatro algoritmos foram avaliados e comparados com o índice CRand.

Dentre os trabalhos que utilizam meta-heurísticas estão aqueles que oferecem melhorias ao k-Médias. Eles aplicam meta-heurísticas na solução de problemas de agrupamentos utilizando a mesma função objetivo do k-Médias. Os trabalhos apresentados a seguir são alguns dos que utilizam essa abordagem.

Em Chang et al. (2009), é proposto um rearranjo de genes com Algoritmo Genético (Goldberg, 1989) chamado Algoritmo Genético com Rearranjo de Genes (GAGR, do inglês *Genetic Algorithm with Gene Rearrangement*), modelado para otimizar a mesma função objetivo do k-Médias. Os autores realizaram testes em conjuntos de dados reais e também em imagens de sensoriamento remoto. Geraram *clusters* com mais três algoritmos: k-Médias (MACQUEEN, 1967), GA-clustering (MURTHY; CHOWDHURY, 2003), KGA-clustering (BANDYOPADHYAY; MAULIK, 2002). Os *clus-*

ters obtidos com os algoritmos foram avaliados pelo índice de validação Rand (Rand, 1971), o qual foi utilizado na comparação dos quatro algoritmos.
Assim como o GAGR, os outros algoritmos utilizados na comparação também oferecem melhorias ao k-Médias utilizando-se de meta-heurísticas, especificamente o Algoritmo Genético.

Al-Sultan (1995) oferece uma melhoria ao k-Médias por meio da meta-heurística Busca Tabu (GLOVER, 1986). Os autores geraram *clusters* com Busca Tabu, k-Médias e Simulated Annealing (KIRKPATRICK et al., 1983). Por fim, fizeram a comparação dos três algoritmos por meio das funções objetivo obtidas por eles.

Durante a pesquisa bibliográfica não foram encontrados trabalhos que propusessem as mesmas meta-heurísticas tratadas neste trabalho como algoritmos de agrupamento para classificação particional.

4 META-HEURÍSTICAS

Neste capítulo é feita uma introdução ao termo meta-heurística e são feitas definições importantes para as abordagens que serão descritas nos próximos capítulos.

O termo meta-heurística é derivado do termo heurística. Uma heurística é uma técnica de busca local que procura boas soluções em um tempo computacionalmente aceitável, mas, não garante que a solução encontrada seja o ótimo global nem mesmo quão próxima ela está da ótima. Uma grande dificuldade de uma heurística é fugir dos ótimos locais, visto que o objetivo é alcançar o ótimo global. Para suprir essa dificuldade, uma outra metodologia foi criada: a meta-heurística. Uma meta-heurística é um algoritmo aproximativo que tenta combinar métodos heurísticos básicos com mecanismos de alto nível para realizar uma busca eficiente no espaço de soluções (BLUM; ROLI, 2003). Esses mecanismos são responsáveis pela fuga dos ótimos locais e tem como objetivo realizar buscas em regiões mais promissoras.

Neste trabalho serão utilizadas três meta-heurísticas diferentes: VNS, ILS e CS. Antes de descrever as meta-heurísticas, é necessário realizar duas definições importantes para o problema de agrupamento. A primeira delas é a representação da solução, pois é por meio da solução que o problema é representado e trabalhado durante a execução das meta-heurísticas. A outra definição a ser feita é a função objetivo, a qual faz a avaliação de cada solução gerada pelas meta-heurísticas.
As próximas seções explicam essas duas definições.

4.1 Representação das soluções

Neste trabalho, os conjuntos de dados serão representados como grafos de proximidade, por isso, nas meta-heurísticas, o conjunto de nós desse grafo será representado por um vetor $s = \{s_1, s_2, s_3, ..., s_N\}$, sendo N o número total de nós (que é o mesmo número de objetos do conjunto de dados). A cada posição i desse vetor é atribuído um *cluster* $k \in \{1, K\}$, numericamente representado de acordo com a quantidade de clusters K definida a *priori*. Assim o objeto s_i correspondente à posição i do vetor s pertencerá ao *cluster* k atribuído a essa posição. Com isso, esse vetor s representará uma solução. A Figura 4.1 ilustra um exemplo de solução para 5 objetos e 2 classes.

A representação das soluções é numérica e não leva em consideração o número do *cluster*, mas sim o agrupamento dos objetos. Por exemplo, um *cluster* de número 2 formado pelos objetos 3, 4 e 5 em uma solução, pode ter qualquer outro número

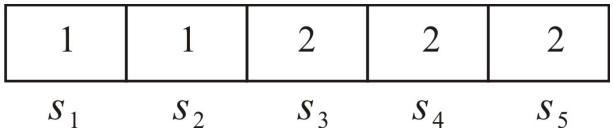

Figura 4.1 - Exemplo de solução

para representá-lo em outra solução desde que os objetos pertencentes a ele sejam os mesmos 3, 4 e 5. A Figura 4.2 ilustra duas soluções iguais com 5 objetos e 2 *clusters*. Em uma solução os *clusters* são representados pelos números 1 e 2 e na outra solução por 3 e 4, mas os objetos agrupados são os mesmos, mostrando que as duas soluções são iguais.

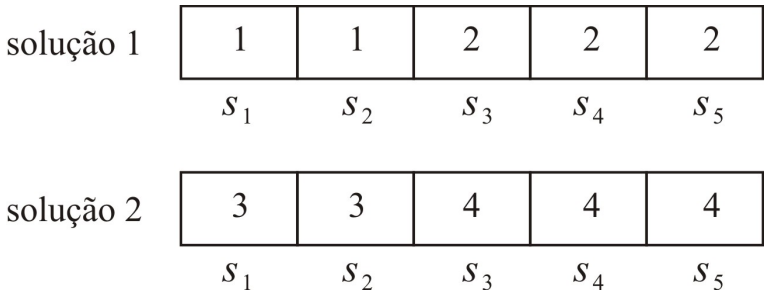

Figura 4.2 - Exemplo de duas soluções iguais

4.2 Função objetivo

A função objetivo deste problema se trata da soma das dissimilaridades entre os objetos pertencentes a um mesmo *cluster*. Seja d_{ij} a dissimilaridade entre o objeto s_i e s_j (posições da solução s que representam os objetos x_i e x_j), as soluções serão avaliadas com base na seguinte função:

$$f(s) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}, \ \forall \, s_i = s_j \tag{4.1}$$

onde, as dissimilaridades entre os objetos s_i e s_j só serão somadas se $s_i = s_j$, ou seja, se o objeto x_i e o objeto x_j (do conjunto de dados) pertencerem ao mesmo *cluster*. Assim, $f(s)$ receberá a soma das dissimilaridades entre os objetos pertencentes ao mesmo *cluster*. Este é um problema de minimização, então, o valor desta função objetivo deverá ser minimizado pelas meta-heurísticas.

5 BUSCA EM VIZINHANÇA VARIÁVEL E BUSCA LOCAL ITERATIVA

Neste capítulo serão apresentadas as meta-heurísticas clássicas propostas como algoritmos de agrupamento. Antes, serão apresentadas definições importantes para o funcionamento destas meta-heurísticas. E serão mostrados os pseudocódigos dos algoritmos utilizados nos experimentos.

5.1 Solução inicial e Busca Local

O método de geração da solução inicial e o de busca local são partes importantes de uma meta-heurística. Nesta seção, serão descritos os dois procedimentos propostos neste trabalho: o que gera a solução inicial e o que realiza a busca local. Esses dois métodos serão utilizados pelas duas meta-heurísticas descritas neste capítulo.

5.1.1 Método de geração da solução inicial

O processo de geração da solução inicial será feito aleatoriamente, ou seja, será atribuído um *cluster* aleatório a cada posição de uma solução. Dessa forma, são geradas várias soluções e somente a que obtiver o menor valor na função objetivo é que será considerada solução inicial para as meta-heurísticas. A Figura 5.1 ilustra o pseudocódigo do método que gera a solução inicial.

```
Algoritmo - Gerar solução inicial()
   Inicialize a solução inicial s;
   Inicialize a solução auxiliar sAux;
   Seja K o número de clusters existentes;
   Enquanto (condição de parada não satisfeita) faça
      Para cada posição de sAux faça
         Atribua randomicamente um cluster k ∈ {1, K};
      fim-para;
      Se (f(sAux) < f(s)) então
         s ← sAux;
      fim-se;
   fim-enquanto;
   Retorne s;
fim-algoritmo;
```

Figura 5.1 - Pseudocódigo de geração da solução inicial

O método inicia atribuindo a cada posição de *sAux* um *cluster* k definido aleatoriamente. Dessa forma, são geradas 100 soluções e somente que obtiver o menor valor na função objetivo é que será considerada a solução inicial para as meta-heurísticas.

5.1.2 Busca Local

O funcionamento da busca local utilizada pelas meta-heurísticas propostas é baseado na atribuição de um objeto a todos os *clusters* diferentes do qual ele já pertence com intuito de descobrir o *cluster* mais coerente. Assim, se o *cluster* atribuído oferecer melhora à solução, esta solução será considerada. A Figura 5.2 ilustra o pseudocódigo da busca local utilizada neste trabalho.

```
Algoritmo - Busca local(solução s')
   Seja K a quantidade de clusters existentes;
   Inicialize uma solução s'' para representar o ótimo local;
   s'' ← s';
   Enquanto (condição de parada não satisfeita) faça
      Selecione uma posição aleatória s'_i da solução s';
      Guarde essa posição para que não seja visitada novamente;
      Para cada cluster k ∈ {1, K} faça
         Atribua k à posição s'_i ;
         Se (f'(s') < f'(s'')) então
            s'' ← s';
         Senão
            s' ← s'';
         fim-se;
      fim-para;
   fim-enquanto;
fim-algoritmo;
```

Figura 5.2 - Pseudocódigo da busca local

Esta rotina realiza testes atribuindo todos os *clusters* a uma posição aleatória da solução, ou seja, testa o mesmo objeto em todos os *clusters* existentes, aceitando o *cluster* que obtiver menor valor na função objetivo. Essa posição é guardada para que não seja visitada novamente durante a mesma iteração da meta-heurística.

Para que o esforço computacional exigido pela busca local seja menor, sua função objetivo sofreu uma simplificação. Quando a solução s' chega à rotina da busca lo-

cal, sabe-se seu valor de função objetivo $f(s')$. Com isso, ao se modificar s', sabe-se também em qual s'_j foi realizada a mudança de *cluster*. Seja k_{ant} e k_{novo}, respectivamente, o antigo e o novo *cluster* de s'_j, $fo = f(s')$ e d_{ij} a dissimilaridade entre s'_i e s'_j, a função $f'(s')$ para a busca local é dada por:

$$f'(s') = fo, \quad \text{sendo que,} \quad \forall\, i, \quad fo = \begin{cases} fo - d_{ij}, & \text{se} \quad s'_i = k_{ant}, \\ fo + d_{ij}, & \text{se} \quad s'_i = k_{novo}, \end{cases} \tag{5.1}$$

assim, a dissimilaridade entre o objeto s_i e s_j será somada quando s_j for igual ao novo *cluster* e subtraída se s_j for igual ao *cluster* anterior.
Quando a busca local termina sua execução, calcula-se a função $f()$, pois a função $f'()$ apresenta problemas de arredondamento impedindo sua utilização no restante do algoritmo.

5.2 Busca em Vizinhança Variável - VNS

A meta-heurística Busca em Vizinhança Variável (VNS, do inglês *Variable Neigborhood Search*) proposta inicialmente por Mladenović e Hansen (1997), é caracterizado por mudar de vizinhança constantemente, a fim de escapar dos ótimos locais. O VNS explora vizinhanças cada vez mais distantes da solução corrente e só aceita uma nova solução se e somente se acontecer alguma melhora.
A Figura 5.3 mostra o pseudocódigo do VNS.

Seja um conjunto pré-definido de estruturas de vizinhança V^p, onde $p = \{1, ..., p_{max}\}$, e $V^p(s^*)$ o conjunto de soluções da *p-ésima* vizinhança de s^*, o algoritmo parte de uma solução inicial qualquer e a cada iteração gera um vizinho aleatório s' pertencente à vizinhança V^p da solução corrente ($s' \in V^p(s^*)$). Então, aplica uma busca local no vizinho s' e encontra uma solução s'' representando um ótimo local. Se este ótimo local for melhor que a solução corrente, a busca continua a partir dele, recomeçando da primeira estrutura de vizinhança. Caso contrário, a busca continua da próxima estrutura de vizinhança V^{p+1}. O VNS finaliza quando atinge a condição de parada.
O processo descrito acima é o funcionamento de um VNS básico. Na próxima seção, encontra-se a explicação do VNS aplicado a problemas de agrupamentos.

```
Algoritmo - VNS()
   Gere uma solucao inicial s;
   Seja p_max o número de estruturas diferentes de vizinhança;
   Inicialize a solução s* para a melhor solução encontrada;
   s* ← s;
   Enquanto ( critério de parada não for satisfeito ) faça
      p ← 1;
      Enquanto ( p ≤ p_max ) faca
         Gere um vizinho s' aleatoriamente da p-ésima vizinhança de s* (s' ∈ V_p(s));
         Aplique uma busca local em s' obtendo um ótimo local ( s'' );
         Se ( s'' for melhor que s* ) então
            s* ← s'';
            p ← 1;
         Senao
            p ← p + 1;
      fim-enquanto;
   fim-enquanto;
fim-algoritmo.
```

Figura 5.3 - Pseudocódigo do VNS
Fonte: Adaptado de Hansen e Mladenović (2001)

5.2.1 VNS aplicado a problemas de agrupamentos

Neste trabalho, os problemas de agrupamentos são solucionados por meio do particionamento em cliques. Por isso, o VNS é modelado para realizar o particionamento do grafo de proximidade em cliques. A disposição dos *clusters* na estrutura de solução é que caracteriza o particionamento dos objetos e a avaliação dessa partição é feita por meio da função objetivo descrita no capítulo anterior, a qual deve ser minimizada.
A Figura 5.4 ilustra o pseudocódigo do VNS modelado para solucionar o agrupamento.

A seguir, é descrito o processo de geração da vizinhança variável do VNS.

5.2.1.1 Vizinhança variável

A principal característica do VNS é a busca em vizinhos distantes a partir de inúmeras estruturas diferentes de vizinhança. Para se chegar a essas vizinhanças, neste trabalho, o VNS contará basicamente com dois movimentos: troca dos *clusters* en-

```
Algoritmo - VNS(solução inicial s)
   Inicialize a solução s* para a melhor solução encontrada;
   Seja p_max o número de estruturas diferentes de vizinhança;
   p ← 1;
   s* ← s;
   Enquanto (critério de parada não satisfeito) faça
      Gere um vizinho s' da vizinhança V_p(s*);
      Aplique a Busca Local em s' obtendo ótimo local s'';
      Se (f(s'') < f(s*)) então
         s* ← s";
         p ← 1;
      Senão
         p ← p + 1;
      fim-se;
      Se (p > p_max) então
         p ← 1;
      fim-se;
   fim-enquanto;
fim-algoritmo.
```

Figura 5.4 - Estrutura do VNS aplicado a problemas de agrupamentos

tre dois objetos (*swap*) e mudança de um objeto de *cluster*(*shift*). As estruturas de vizinhança são geradas a partir dos movimentos realizados. Com isso, quanto mais movimentos a realizar, mais distantes serão as vizinhanças. A Figura 5.5 mostra o pseudocódigo da rotina que gera os vizinhos a partir das 6 vizinhanças ($p_{max} = 6$), definidas pelos movimentos de *shift* e *swap*.

O *swap* é feito a partir da seleção aleatória de 1, 2 ou 3 pares de objetos (posições) da solução s'. Como exemplo, imagine que na seleção de 1 par de objetos sejam selecionados os objetos 5 e 10, o *swap* se dá pela atribuição do *cluster* da posição 5 à posição 10 e vice-versa. Com isso, o objeto 5 passa a pertencer ao antigo *cluster* do objeto 10 e este passa a pertencer ao antigo *cluster* do objeto 5.
O *shift* se dá pela seleção aleatória de uma posição da solução s' e a atribuição de um *cluster* aleatório a essa posição, excluindo o *cluster* anterior. Com esse movimento, o objeto selecionado sai de um *cluster* para pertencer a outro.

O VNS tem como condição de parada o número de iterações que será igual a 50.

```
Algoritmo - Gerar vizinho(solução s, vizinhança p)
   Inicialize a solução vizinha s';
   s' ← s;
   Caso p seja
      1 : Troque os clusters entre um par aleatório de objetos em s';
      2 : Mude aleatoriamente um objeto de cluster em s';
      3 : Troque os clusters entre 10% de pares aleatórios de objetos em s';
      4 : Mude aleatoriamente 20% dos objetos de cluster em s';
      5 : Troque os clusters entre 30% de pares aleatórios de objetos em s';
      6 : Mude aleatoriamente 50% dos objetos de cluster em s';
   fim-caso;
   Retorne s';
fim-algoritmo;
```

Figura 5.5 - Pseudocódigo de geração dos vizinhos

5.3 Busca Local Iterativa - ILS

A meta-heurística Busca Local Iterativa (ILS, do inglês *Iterated Local Search*), proposta inicialmente por Lourenço et al. (2003) e Stützle (1999), é caracterizada por focar a busca em um subsespaço S^* formado por soluções que são ótimos locais de determinado procedimento de otimização.

O algoritmo ILS consiste em um processo iterativo que, a cada iteração, realiza uma perturbação aleatória em uma solução gerando novas soluções de partida para um método de busca local. Essa perturbação tem uma importância muito grande, pois se for pequena pode fazer com que o algoritmo não consiga escapar da região de atração de um ótimo local já pesquisado. Porém, se for muito grande o algoritmo tem um reinício aleatório (BLUM; ROLI, 2003). A Figura 5.6 mostra o pseudocódigo do ILS.

O algoritmo ILS inicializa gerando uma solução inicial qualquer para o problema (s). Em seguida, é aplicada uma heurística de busca local em s obtendo uma solução ótima local (s''). A cada iteração realiza-se um procedimento de perturbação aleatória da solução s'' produzindo um novo ponto de partida para a busca local (s'). A heurística de busca local é então aplicada sobre s' encontrando a solução ótima local desta região (s'''). Um critério de aceitação é aplicado para decidir a partir de qual solução a busca continuará. O ILS finaliza seu processo quando a condição de

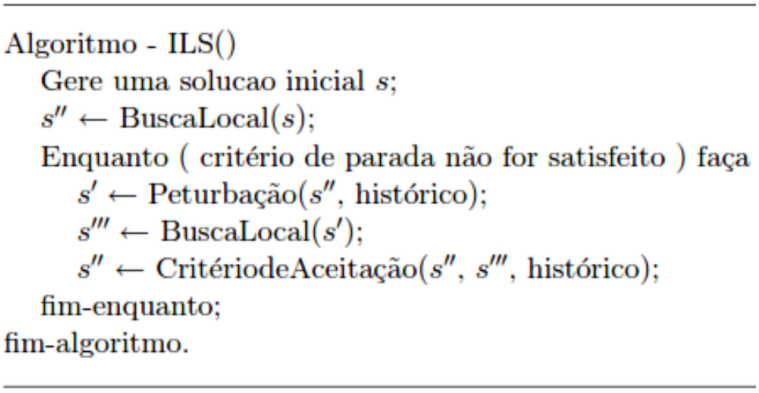

Figura 5.6 - Pseudocódigo do ILS
Fonte: Adaptado de Lourenço et al. (2003)

parada é atingida.
O processo descrito acima é o funcionamento básico de um ILS. A seguir, encontra-se a descrição do ILS aplicado a problemas de agrupamentos.

5.3.1 ILS aplicado a problemas de agrupamentos

Assim como o VNS, o ILS também é modelado com a finalidade de particionar o grafo de proximidade em cliques. Para isso, utilizará a representação de solução descrita no capítulo anterior, onde a disposição dos *clusters* na solução é que determina o particionamento dos objetos em cliques.
A Figura 5.7 ilustra o ILS aplicado a problemas de agrupamentos.

A perturbação realizada nas soluções é bastante importante, pois precisa ser forte o suficiente para que sempre se explore novas regiões e fuja dos ótimos locais, mas também precisa ser fraca o suficiente para guardar características do ótimo local corrente. A quantidade de alterações que ela realiza nas soluções, também chamada de intensidade da perturbação, pode ser fixa ou variável. No caso dos problemas de agrupamentos tratados neste trabalho, a intensidade da perturbação será variável. A Figura 5.8 ilustra o pseudocódigo da perturbação utilizada no ILS deste trabalho.

Outra característica do ILS é a dependência de um *histórico*, ou seja, a realização da perturbação e o critério de aceitação dependem das soluções visitadas anteriormente no processo de busca. Neste trabalho, o ILS foi implementado de forma que não dependa de um *histórico*, assim o critério de aceitação terá uma regra fixa que será

```
Algoritmo - ILS()
   Inicialize s* para a melhor solução encontrada;
   Gere uma solução inicial s;
   s'' ← BuscaLocal(s);
   Enquanto ( critério de parada não for satisfeito ) faça
      s' ← Peturbação(s'');
      s''' ← BuscaLocal(s');
      Se (f(s''') < f(s'')) então
         s'' ← s''';
         s* ← s''';
      fim-se;
   fim-enquanto;
fim-algoritmo.
```

Figura 5.7 - Pseudocódigo do ILS aplicado a problemas de agrupamentos

```
Algoritmo - Perturbação(s'')
   Seja N o número total de objetos;
   Inicialize s' para a solução perturbada;
   Gere um valor aleatório β ∈ [0.4, 0.7];
   Inicialize uma variável perturb para a intensidade da perturbação;
   perturb ← N × β;
   Enquanto (perturb > 0)
      Gere um valor aleatório aux ∈ {0, 1};
      Se aux = 0 então
         Troque dois clusters entre um par aleatório de objetos em s'' e atribua a s';
      fim-se;
      Se aux = 1 então
         Mude aleatoriamente um objeto de cluster em s'' e atribua a s';
      fim-se;
      perturb = perturb − 1;
   fim-enquanto;
   Retorne s';
fim-algoritmo;
```

Figura 5.8 - Pseudocódigo da perturbação no ILS

baseada no valor da função objetivo encontrada para as soluções. Se a solução s''' obtiver menor valor na função objetivo, ela passa a ser a solução que retornará ao processo iterativo (s'') e s^* a guardará como melhor solução encontrada até então.

O ILS tem como condição de parada o número de iterações que será igual a 50.

6 BUSCA POR AGRUPAMENTOS - CS

Este capítulo tem o objetivo de mostrar a meta-heurística híbrida CS. Para que fosse aplicada em problemas de agrupamentos, o CS precisou de várias adaptações que são explicadas nas próximas seções. Por fim, serão mostrados os algoritmos VNS-CS e ILS-CS que serão utilizados no experimento computacional.

6.1 Busca por agrupamentos - CS

O algoritmo Busca por Agrupamentos (CS, do inglês *Clustering Search*), proposto por Chaves (2009), é uma generalização do algoritmo Busca Evolutiva por Agrupamentos (ECS, do inglês *Evolutionary Clustering Search*) proposto inicialmente por Oliveira (2004). O objetivo do CS é intensificar as buscas somente em regiões que sejam promissoras à melhoria da solução, selecionando as soluções que passarão pela busca local, ao invés de realizá-la em todas as soluções ou escolhê-las aleatoriamente (CHAVES, 2009).
O CS já foi aplicado a vários problemas de otimização combinatória. Além dos problemas abordados em Chaves (2009), é possível destacar as aplicações do CS ao Problema do Caixeiro Viajante com Coleta de Prêmios (CHAVES; LORENA, 2008), ao Problema da Atribuição e Balanceamento de Trabalhadores à Linhas de Montagem (CHAVES et al., 2009), ao Problema de Máxima Cobertura Probabilística Localização-Alocação (CORRÊA et al., 2007), Problema de Agrupamentos Centralizados e Capacitados (CHAVES; LORENA, 2010) e Problema P-Medianas Capacitado (CHAVES et al., 2007).

6.1.1 Funcionamento do CS

A ideia do CS é dividir o espaço de busca de forma a agrupar as regiões promissoras em grupos. No CS, um grupo pode ser definido por três atributos $C = (c, v, r)$. Cada grupo C_i terá uma solução que o representará e servirá como sua localização no espaço de busca. Essa solução será chamada de centro c_i. Um grupo C_i não "abrigará" fisicamente todas as soluções agrupadas, mas terá o seu centro c_i, que é uma solução, formado por parte das informações dessas soluções. O total de soluções agrupadas no grupo C_i será chamado de volume v_i. Quando o volume v_i atingir um limitante λ um grupo se torna promissor. Para saber se a busca local está melhorando ou não o centro do grupo C_i um índice de ineficácia r_i será utilizado. O valor r_i será o número de vezes consecutivas que a busca local foi aplicada no grupo C_i sem

melhorar a solução. Este índice é importante no CS, pois evita que a busca local seja executada mais de r_{max} vezes em regiões ruins ou que já tenham sido suficientemente exploradas.
A Figura 6.1 ilustra o método CS para um problema de minimização.

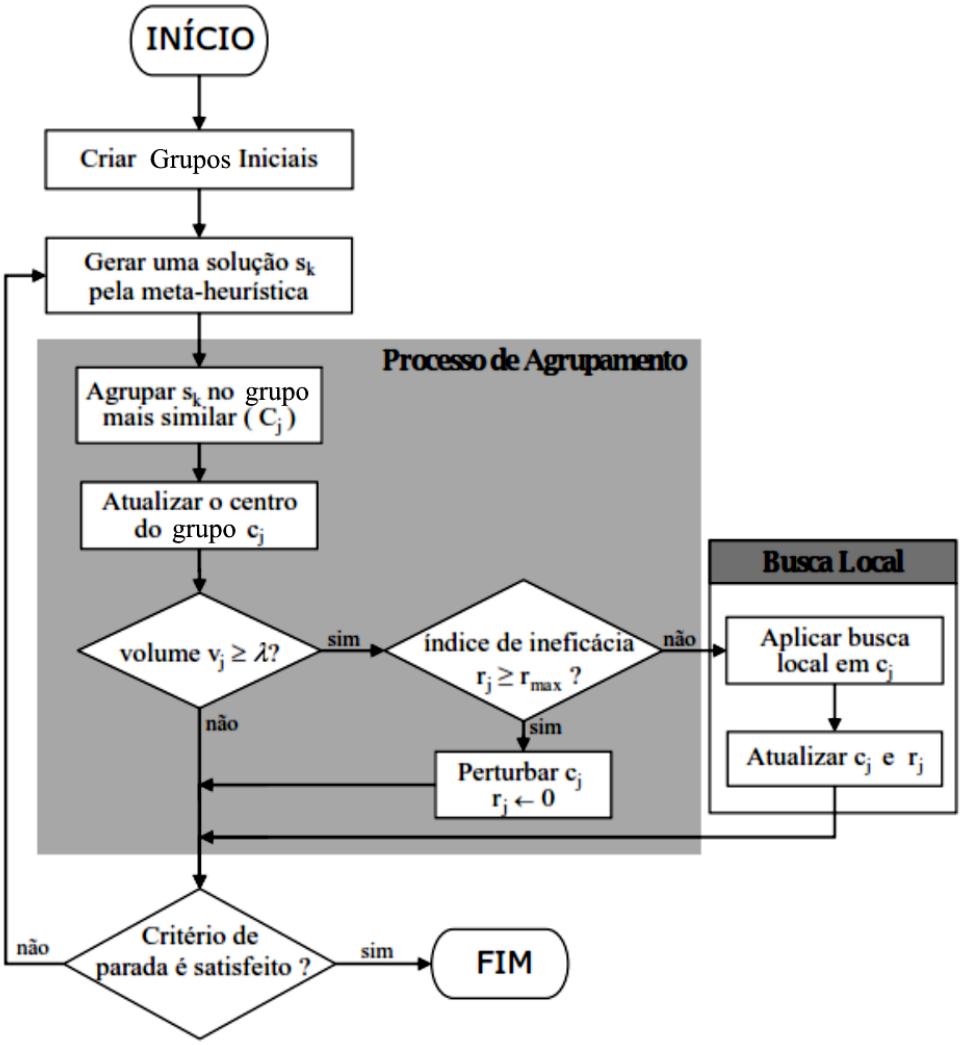

Figura 6.1 - Fluxograma do CS
Fonte: Adaptado de Chaves (2009)

O CS é chamado de método híbrido (ou meta-heurítica híbrida) justamente por combinar diferentes métodos heurísticos. É composto por três componentes principais: uma meta-heurística, um processo de agrupamento e um método de busca local. O

```
Algoritmo - CS
   Crie os grupos iniciais;
   {meta-heurística}
   Enquanto (critério de parada não satisfeito) faça
      Gere uma solução s_t pela meta-heurística;
      {processo de agrupamento}
      Encontre o grupo mais similar a s_t (C_j|d(s_t, c_j) ≡ min{d(s_t, c)});
      Agrupe s_t no grupo mais similar C_j (v_j ← v_j + 1);
      Atualize o centro do grupo (c_j ← Assimilação(c_j, s_t));
      Se (v_j ≥ λ) então
         Reduza o volume v_j ← 0;
         Se (r_j ≥ r_max) então
            Aplique uma perturbação aleatória em c_j;
            Zere o índice de ineficácia r_j ← 0;
         Senão
            {heurística de busca local}
            Encontre o melhor vizinho c''_j de c_j;
            Se (f(c''_j) < f(c_j)) então
               Atualize c_j ← c''_j;
            Se (f(c''_j) < f(c*_j)) então
               Zere o índice de ineficácia r_j ← 0;
               Atualize c*_j ← c''_j;
            Senão
               Atualize o índice de ineficácia r_j ← r_j + 1;
         fim-se;
      fim-se;
   fim-enquanto;
fim-algoritmo;
```

Figura 6.2 - Pseudocódigo do CS
Fonte: Adaptado de Chaves (2009)

CS é um método iterativo e a cada iteração executa os seguintes passos:

- a meta-heurística gera uma solução s_t que é enviada ao processo de agrupamento. Então, calcula-se a distância entre a solução s_t e os grupos existentes para saber a qual grupo ela será agregada. Supondo que o grupo mais próximo dela seja o C_j, o centro c_j é atualizado com informações dessa nova solução por meio do processo de assimilação;
- o próximo passo é analisar o volume v_j. Caso ele já tenha atingido o limitante λ, definido a *priori*, o grupo C_j pode estar numa região de busca

promissora.

- se $v_j \geq \lambda$, o índice de ineficácia r_j é verificado. Se r_j ainda não tiver atingido o limite r_{max}, a busca local é aplicada ao grupo C_j, intensificando a busca na vizinhança deste grupo. Mas, se $r_j \geq r_{max}$ é realizada uma perturbação aleatória no centro c_j, a fim de fugir desta região do espaço de busca.

- o sucesso da busca em um grupo se dá pela melhor solução (c_j^*) encontrada neste grupo até o momento.

- o processo de agrupamento é encerrado, retornando para a meta-heurística que irá gerar outra solução, partindo para a próxima iteração do CS.

O critério de parada do CS pode ser definido pela quantidade de iterações do processo de agrupamento ou o mesmo critério de parada da meta-heurística utilizada. A Figura 6.2 ilustra o pseudocódigo do CS para um problema de minimização.

Na próxima seção será apresentado o CS aplicado a problemas de agrupamentos.

6.2 CS aplicado a problemas de agrupamentos

Como visto anteriormente, o CS é formado por três componentes: uma meta-heurística, um processo de agrupamento e uma busca local. Nas próximas subseções são definidos os componentes do CS.

6.2.1 Meta-heurísticas

Para a aplicação em problemas de agrupamentos, o CS foi implementado utilizando duas abordagens diferentes. A primeira utilizou o VNS como meta-heurística e a segunda utilizou o ILS, ambas já mostradas no capítulo anterior. A abordagem do CS com a meta-heurística VNS, será chamada VNS-CS e a abordagem com ILS será chamada ILS-CS.

A utilização do VNS no VNS-CS e do ILS no ILS-CS da forma como as meta-heurísticas foram descritas no capítulo anterior, apresentou alto custo computacional. Para contornar este problema, foi feita uma adaptação no componente referente às meta-heurísticas. A adaptação faz com que cada componente otimize uma função objetivo, diminuindo o custo computacional. Assim, as meta-heurísticas farão a otimização da função objetivo do problema de p-medianas enquanto o componente

referente a busca local irá otimizar a função objetivo dos problemas de agrupamentos, apresentada no Capítulo 4.

O problema de p-medianas, clássico na literatura, tem o objetivo de encontrar as p facilidades (denominadas medianas) em uma rede de nós de modo a minimizar a soma das distâncias entre cada nó e a mediana mais próxima (SENNE; LORENA, 2003). Com base nisso, as meta-heurísticas tem o objetivo de encontrar os objetos centrais (medianas dos *clusters*) entre todos os objetos de forma a minimizar a soma das dissimilaridades entre cada objeto e o objeto central.

Para isso, foi feita uma alteração na representação das soluções. Além do vetor de soluções, também terá um vetor de medianas $med = \{med_1, ..., med_K\}$, formado por K posições, onde K é quantidade de *clusters* existentes. Dessa forma, os objetos centrais serão guardados no vetor de medianas e os objetos que pertencerem a uma determinada mediana, terão em sua posição do vetor de solução o valor dessa mediana (a posição do objeto central).

A Figura 6.3 ilustra a representação de uma solução com 6 objetos e 2 *clusters*, sendo 2 objetos centrais: 1 e 4. Assim, o vetor de medianas, de tamanho 2 por serem 2 *clusters*, recebe os valores 1 e 4 em suas posições. Os objetos pertencentes ao *cluster* que tem o objeto 1 como mediana, terão o valor 1 em suas posições. Já, os objetos pertencentes ao *cluster* que tem o objeto 4 como mediana, terão o valor 4 em suas posições.

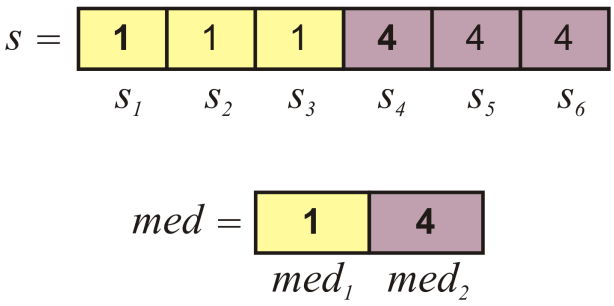

Figura 6.3 - Exemplo do vetor de medianas

Seja, N o total de objetos, K a quantidade de *clusters* e $d(med_j, i)$ a dissimilaridade entre a *j-ésima* mediana e o objeto i, a função objetivo f_{pmed} utilizada pelas meta-

heurísticas em ILS-CS e VNS-CS é dada por:

$$f_{pmed} = \sum_{i=1}^{N} \min_{1 \leq j \leq K} \{d(med_j, i)\} \tag{6.1}$$

onde, só será somada, para todos os N objetos, a dissimilaridade da mediana que for mais próxima ao ponto i.

Para trabalhar com a função objetivo do p-medianas, o algoritmo de geração da solução inicial também teve de ser adaptado, pois a solução inicial para as meta-heurísticas deve trazer além do vetor de solução, um vetor de medianas. Para isso, o algoritmo inicia um vetor de medianas $med = \{med_1, med_2, .., med_K\}$ e atribui a cada posição dele um objeto aleatório $x \in [1, N]$. Após esse passo, o algoritmo entra num processo iterativo com objetivo de atribuir a cada posição i da solução $sAux$ a mediana que tiver a menor distancia entre med_j e o objeto x_i. Esse processo é realizado 100 vezes, gerando 100 soluções, mas apenas a que obtiver o menor valor da função objetivo f_{pmed} é que será a solução inicial s das meta-heurísticas. Esse algoritmo é utilizado no componente meta-heurística do VNS-CS e do ILS-CS. A Figura 6.4 ilustra o pseudocódigo do método de geração da solução inicial.

A busca local utilizada pelo componente meta-heurística trata-se de um *shift* aleatório no vetor de medianas med da uma solução s'. O objetivo é realizar a busca das melhores medianas para a solução s' selecionando aleatoriamente um objeto da solução s' e assumindo-o como mediana. Com isso, todos os objetos de s' que antes eram pertencentes a mediana anterior, passam a pertencer a nova mediana. Depois disso, é calculada a função objetivo do problema de p-medianas f_{med} e guardada em s_t a melhor solução encontrada (solução que será enviada para o processo de agrupamento do CS). O critério de parada assumido é a quantidade de iterações, que neste caso será igual a 10% do total de objetos (N). A Figura 6.5 ilustra o pseudocódigo dessa busca local.

```
Algoritmo - Gerar solução inicial()
   Seja N o total de objetos e K o número de clusters existentes;
   Inicialize a solução inicial s;
   Inicialize a solução auxiliar sAux;
   Seja K o número de clusters existentes;
   Enquanto (condição de parada não satisfeita) faça
      Para cada posição de med faça
         Atribua aleatoriamente um objeto x ∈ {1, N};
      fim-para;
      Para cada i = 1, .., N faça
         sAux_i = min_{med_j} {d(med_j, x_i), j = 1, .., K};
      fim-para;
      Se (f_pmed(sAux) < f_pmed(s)) então
         s ← sAux;
      fim-se;
   fim-enquanto;
   Retorne s;
fim-algoritmo;
```

Figura 6.4 - Pseudocódigo para gerar solução inicial

```
Algoritmo - BuscaLocalMH(s)
   Inicialize uma solução s_t;
   Enquanto (condição de parada não satisfeita) faça
      Atribua a uma posição aleatória do vetor med uma objeto aleatório de s';
      Realocar à nova mediana todos objetos de s' pertencentes a mediana anterior;
      Se (f_pmed(s') < f_pmed(s_t)) então
         s_t ← s';
      fim-se;
   fim-enquanto;
   Retorne s_t;
fim-algoritmo;
```

Figura 6.5 - Pseudocódigo da busca local do componente meta-heurística

6.2.2 Criar grupos iniciais

Depois de definidas as meta-heurísticas, é preciso, então, definir a quantidade de grupos do CS, ou seja, a quantidade de regiões na qual o espaço de busca será

dividido. Neste trabalho, são utilizados 10 grupos. Os grupos iniciais são criados contendo uma solução c_i como centro do grupo. Esses centros de grupo devem ser gerados de forma a representar diferentes regiões do espaço de busca. Para tal, é utilizado o método baseado na diversidade máxima entre centros de grupos, assim como proposto em Chaves (2009).

A medida de diversidade entre centros de grupos pode ser medida pela distância entre eles. Neste trabalho, a medida de diversidade utilizada para calcular a distância entre centros de grupos será o índice Rand (R). Assim, quanto menor for o valor de R maior será a diversidade entre dois centros de grupos, visto que quanto menor o valor de R mais diferentes são as soluções.

Seja Q o conjunto de q soluções aleatórias, $Rd(i, j)$ o valor do índice Rand entre a solução i e j de Q, e W um conjunto diverso com w soluções ($w < q$). O método para criar os centros dos grupos iniciais consiste em selecionar w soluções de Q, de forma a minimizar a soma dos Rands entre essas soluções.
Inicialmente é necessário montar uma matriz simétrica $Rd_{q \times q}$ que armazena os Rands calculados entre cada uma das soluções de Q. O método inicia selecionando aleatoriamente entre as soluções de Q a primeira solução de $W(w_1)$. A segunda solução de $W(w_2)$ deve ser a solução $j \in Q - w_1$ que forneça maior diversidade entre w_1 e j. A partir da terceira solução é necessário calcular $Rd_{soma}(i), \forall i \in Q - W$, que é a soma das diversidades entre o candidato i e todas as soluções pertencentes ao conjunto parcial W. Por estar utilizando o valor de Rand como medida de diversidade, a solução i que obtiver menor valor de Rd_{soma} é selecionada para ser incluída no conjunto W. Esse processo é repetido até que as w soluções sejam selecionadas. Cada solução de W será o centro de um grupo do CS. A Figura 6.6 ilustra o pseudocódigo de criação dos centros de grupos.

6.2.3 Processo de agrupamento

O processo de agrupamento do CS é responsável por direcionar a busca para regiões supostamente promissoras. O primeiro passo é selecionar o grupo C_j que seja mais próximo da solução s_t enviada pela meta-heurística. O objetivo é reunir soluções similares mantendo a mais significativa como centro do grupo. Para isso, realiza-se um processo de assimilação entre a solução s_t e o centro c_j, para que algumas características de s_t sejam inseridas em C_j.

Algoritmo - Criar Grupos Inicias(q, w)
  Gere um conjunto de q soluções aleatórias (Q);
  $W \leftarrow \emptyset$;
  Escolha aleatoriamente a primeira solução $w_1 \in Q$;
  $W \leftarrow W \cup \{w_1\}$;
  Para cada ($j \in Q - w_1$) faça
    Calcule $Rd(j, w_1)$;
  fim-para;
  $w_2 \leftarrow \min\{Rd(u, w_1), u \in Q - w_1\}$;
  $W \leftarrow W \cup \{w_2\}$;
  Para ($e = 3$ até w) faça
    Calcule $Rd_{soma}(i), \forall i \in Q - W$;
    $w_e \leftarrow \min\{Rd_{soma}(u), u \in Q - W\}$;
    $W \leftarrow W \cup \{w_e\}$;
  fim-para;
  Retorne W;
fim-algoritmo;

Figura 6.6 - Pseudocódigo para criar grupos iniciais
Fonte: Adaptado de Chaves (2009)

Segundo Oliveira (2004) existem três formas de realizar a assimilação entre soluções: simples, por recombinação e por caminho. Neste trabalho, será a utilizada a assimilação por caminho que analisa um conjunto de soluções que são geradas no caminho entre s_t e c_j, podendo gerar diversas soluções. Assim como em Chaves (2009), o processo de assimilação deste trabalho será feito através do método de Reconexão por caminho (PR, do inglês *Path-Relinking*)(GLOVER, 1996) que analisa o trajeto que conecta duas soluções.

O PR inicia a partir de duas soluções. A primeira é a solução s_t ($s_{inicial}$) gerada pela meta-heurística e a segunda é o centro do grupo mais similar c_j (s_{guia}). O método inicializa calculando a diferença simétrica entre as duas soluções ($\Delta(s_{inicial}, s_{guia})$) que é o conjunto de movimentos necessários para alcançar s_{guia} a partir de $s_{inicial}$. Essa diferença simétrica é calculada analisando se as posições i das soluções $s_{inicial}$ e s_{guia} são iguais ou não numericamente, para que, a partir disso, se forme o caminho de soluções que conecta $s_{inicial}$ a s_{guia}. Como a representação das soluções, neste caso, é numérica e não leva em consideração o número do *cluster* mas sim os objetos pertencentes a cada *cluster*, o PR é aplicado no vetor de medianas *med* das soluções.

Seja $med_{inicial}$ o vetor medianas de $s_{inicial}$ e med_{guia} o vetor medianas de s_{guia}, a diferença simétrica $\Delta(med_{inicial}, med_{guia})$ é calculada entre os vetores de medianas de cada solução. A cada movimento realizado no vetor de medianas, os objetos da solução pertencentes a mediana anterior são atualizados com a nova mediana. A cada passo, o método analisa todos os movimentos $mv \in \Delta(med_{inicial}, med_{guia})$ de med_s e seleciona o movimento que resultar na melhor solução, aplicando o melhor movimento (mv^*) em med_s ($med_s \oplus mv^*$). O conjunto de possíveis movimentos é atualizado. O PR termina quando med_{guia} for alcançado (fazendo com que $s_{inicial} = s_{guia}$) ou quando uma porcentagem do caminho for alcançada. A melhor solução neste caminho será o novo centro c_j. A Figura 6.7 ilustra o pseudocódigo do PR.

Algoritmo - Reconexão por caminho($s_{inicial}$,s_{guia})
  Calcule a diferença simétrica $\Delta(med_{inicial}, med_{guia})$;
  Inicialize uma solução s;
  $med_s \leftarrow med_{inicial}$;
  Enquanto (critério de parada não satisfeito) faça
    $mv^* \leftarrow \min\{f(med_s \oplus mv) : mv \in \Delta(med_{inicial}, med_{guia})\}$;
    $\Delta(med_s \oplus mv^*, med_{guia}) \leftarrow \Delta(med_s, med_{guia}) \setminus \{mv^*\}$;
    $med_s \leftarrow med_s \oplus mv^*$;
  fim-enquanto;
fim-algoritmo;

Figura 6.7 - Pseudocódigo reconexão por caminho
Fonte: Adaptado de Chaves (2009)

Neste trabalho, o PR percorre apenas 10% do caminho entre s_t e c_j para que não se mova o centro para soluções muito distantes do centro atual.

Depois de realizada a assimilação, é preciso verificar se o grupo ativado já se tornou promissor, analisando se $v_j \geq \lambda$. Se já tiver atingido o limitante λ e o índice de ineficácia r_j ainda não tiver atingido r_{max} iterações ($r_j \leq r_{max}$), a busca é intensificada no grupo promissor, aplicando uma heurística de busca local no centro c_j. Por outro lado, se $r_j \geq r_{max}$ é aplicada uma perturbação aleatória no centro c_j, que neste trabalho foi feita selecionando uma mediana aleatória do vetor med do centro c_j e atribuindo-a a uma posição aleatória de c_j, ou seja, um objeto do centro c_j passa a pertencer a outra mediana. Depois disso, o índice de ineficácia r_j é zerado para que

esse grupo possa ser promissor numa iteração qualquer. Para este trabalho foram assumidos os valores de $r_{max} = 10$ e $\lambda = 20$.

A heurística de busca local aplicada em c_j tem o objetivo de analisar a vizinhança próxima desse centro. Para este trabalho, a heurística de busca local visa testar um objeto i da solução c_j em todos os *clusters* existentes. O *cluster* que oferecer a menor soma da dissimilaridade entre o objeto i e todos os outros pertencentes a esse *cluster* passa a ser o novo *cluster* desse objeto. Como a representação das soluções é baseada no vetor de medianas, a busca local foi implementada de forma que seja atribuída a cada posição i da solução s (que na verdade é o centro c_j) a posição j do vetor de medianas med (medianas de s) que oferecer a menor dissimilaridade entre o objeto i e todos os outros pertencentes à mediana j. Esse processo é realizado até que não haja mais melhora na solução s ou até que seja executado 50 vezes. A Figura 6.8 ilustra o pseudocódigo da busca local do CS.

Algoritmo - BuscaLocalCS(s)
  Seja N o número de objetos e K o número de *clusters* existentes;
  $iteracoes \leftarrow 1$;
  Repita
    Para i de 1 até N faça
      $s_i = \min_{med_k}\{\sum_{j=1}^{N} d(i,j),\ \ \forall s_j = med_k\}, k = 1,..,K;$
    fim-para;
    $iteracoes = iteracoes + 1$;
  Até que não haja mais melhora ou $iteracoes > 50$;
  Retorne s;
fim-algoritmo;

Figura 6.8 - Pseudocódigo da busca local do CS

Depois de aplicada a heurística de busca local, é verificado, através da função objetivo dos problemas de agrupamentos $f()$, se o ótimo local c''_j é melhor que o centro atual c_j. Se sim, $c_j \leftarrow c''_j$. Verifica-se também, se c''_j melhor que c^*_j (considerada a melhor solução encontrada pelo CS). Se sim, $c^*_j \leftarrow c''_j$.

O processo de agrupamento do CS, neste trabalho, tem como critério de parada o número de iterações do componente meta-heurística que é igual a 100. A Figura 6.9

ilustra o pseudocódigo do processo de agrupamento do CS utilizado neste trabalho.

Algoritmo - Processo Agrupamento(s_t)
  Encontre o grupo mais similar a s_t ($C_j | R(s_t, c_j) \equiv \min\{R(s_t, c)\}$);
  $v_j \leftarrow v_j + 1$;
  c_j = Reconexão por caminho(s_t, c_j);
  Se ($v_j \geq \lambda$) então
    $v_j \leftarrow 0$;
    Se ($r_j \geq r_{max}$) então
      Selecione uma mediana aleatória do vetor *med* pertencente
        a c_j e atribua a uma posição aleatória de c_j;
      $r_j \leftarrow 0$;
    Senão
      c_j'' = BuscaLocalCS(c_j);
      Se ($f(c_j'') < f(c_j)$) então
        $c_j \leftarrow c_j''$;
      fim-se;
      Se ($f(c_j'') < f(c_j^*)$) então
        $r_j \leftarrow 0$;
        $c_j^* \leftarrow c_j''$;
      Senão
        $r_j \leftarrow r_j + 1$;
      fim-se;
    fim-se;
  fim-se;
fim-algoritmo;

Figura 6.9 - Pseudocódigo do processo de agrupamento

Com base no que foi descrito anteriormente, as Figuras 6.10 e 6.11 ilustram, respectivamente, o pseudocódigo dos algoritmos VNS-CS e ILS-CS. O método de geração dos vizinhos no VNS-CS e o método de perturbação no ILS-CS são os mesmos utilizados pelas meta-heurísticas descritas no capítulo anterior.

```
Algoritmo - VNS-CS()
   Inicialize a solução s* para a melhor solução encontrada pela meta-heurística;
   Seja p_max o número de estruturas diferentes de vizinhança;
   p ← 1;
   s* ← s;
   Enquanto (critério de parada não satisfeito) faça
      Gere um vizinho s' da vizinhança V_p(s*);
      s'' = BuscaLocalMH(s');
      Processo de Agrupamento(s'');
      Se (f_pmed(s'') < f_pmed(s*))
         s* ← s'';
         p ← 1;
      Senão
         p ← p + 1;
      fim-se;
      Se (p > p_max) então
         p ← 1;
      fim-se;
fim-algoritmo;
```

Figura 6.10 - Pseudocódigo do algoritmo VNS-CS

```
Algoritmo - ILS-CS()
   Inicialize s* para a melhor solução encontrada pela meta-heurística;
   Gere uma solução inicial s;
   s'' ← BuscaLocalMH(s);
   Enquanto ( critério de parada não for satisfeito ) faça
      s' ← Peturbação(s'');
      s''' ← BuscaLocalMH(s');
      Processo Agrupamento(s''');
      Se (f(s''') < f(s'')) então
         s'' ← s''';
         s* ← s''';
      fim-se;
   fim-enquanto;
fim-algoritmo.
```

Figura 6.11 - Pseudocódigo do ILS aplicado a problemas de agrupamentos

7 EXPERIMENTO COMPUTACIONAL E ANÁLISE DOS RESULTADOS

Este capítulo aborda os resultados obtidos com as abordagens heurísticas. Os conjuntos de dados a serem utilizados nos testes foram divididos em dois grupos: A e B. Essa divisão foi feita para que o capítulo fosse dividido em diferentes etapas, iniciando por uma análise de robustez e eficiência das meta-heurísticas e realizando comparações com outros trabalhos da literatura. O capítulo termina com a aplicação em classificação de uma imagem de satélite, comparando com o algoritmo clássico k-Médias.

7.1 Conjuntos de dados

Como os índices utilizados na validação do agrupamento seguem o critério de validação externa, os conjuntos de dados selecionados trazem a classificação real de cada objeto. Foram divididos em dois grupos, descritos a seguir.

7.1.1 Grupo A

Os conjuntos de dados do Grupo A correspondem a quatro dos dez conjuntos utilizados nos testes de Nascimento et al. (2010). São eles:

Íris: esse conjunto de dados consiste de 150 flores (objetos) caracterizadas por quatro atributos: largura e comprimento da sépala e da pétala. De acordo com suas características, as flores são divididas em 3 tipos (classes): íris virgínica, íris setosa e íris versicolor.

Yeast: é formado por 1484 proteínas de leveduras com oito atributos cada. É dividido em 10 *clusters* de acordo com a localização dos sítios de proteínas.

Breast: consiste de 699 células cancerígenas e com nove atributos cada uma. Este conjunto possui duas estruturas diferentes de classificação. Em uma estrutura, as células são divididas em duas classes que caracterizam câncer maligno ou benigno, e na outra estrutura, são divididas em oito *clusters* de maneira cronológica. A estrutura utilizada nos testes será a que contem dois *clusters*.

Proteínas: é composto por 698 objetos correspondentes à dobraduras de proteínas com 125 atributos cada. Esse conjunto também é formado por duas estruturas de classificação, em que na primeira estrutura as dobraduras são divididas em quatro

classes compostos pelas dobraduras principais. Na segunda estrutura, as dobraduras são divididas em 27 subtipos de dobraduras (27 classes). Nos testes será utilizada apenas a estrutura de quatro classes.

Os conjuntos Iris, Yeast e Breast podem ser obtidos em Asuncion e Newman (2007) e o conjunto Proteínas no sítio *http://ranger.uta.edu/∼chqding/protein*. Mais detalhes sobre cada um desses conjuntos de dados são encontrados em Nascimento et al. (2010).

Na Tabela 7.1 é mostrado um resumo das estruturas de cada conjunto de dados deste grupo. A primeira coluna, Conjunto de dados, refere-se ao nome de cada conjunto. A segunda coluna, Objetos, refere-se ao número total de objetos de cada conjunto. A terceira coluna, Atributos, mostra a quantidade de atributos de cada objeto. E a última coluna, *Classes*, refere-se a quantidade de *classes* existentes.

Tabela 7.1 - Estruturas das bases do Grupo A

Conjunto de dados	Objetos	Atributos	*Classes*
Iris	150	4	3
Yeast	1484	8	10
Breast	699	9	2
Protein	698	125	4

7.1.2 Grupo B

O Grupo B é composto por cinco dos seis conjuntos de dados utilizados nos testes de Chang et al. (2009). São eles:

Wine: é composto por informações sobre reconhecimento de vinhos, com 178 objetos e treze atributos. É dividido em 3 tipos de vinhos (3 classes). Seus atributos são classificados como contínuos.

Glass: esse conjunto corresponde a dados de identificação de vidros com 214 objetos e nove atributos. É dividido em 6 tipos de vidros. O estudo de identificação de vidros foi motivado por investigações criminais. Seus atributos são classificados como contínuos.

Balance: composto por 625 objetos com quatro atributos caracterizados pelos resultados de experimentos psicológicos. É dividido em 3 classes.

Liverdisorder: composto por 345 homens (objetos) com seis atributos cada. É dividido em duas classes. Os cinco primeiros atributos correspondem a todos os exames de sangue que caracterizam distúrbios hepáticos por conta do consumo excessivo de álcool. Seus atributos são classificados como contínuos.

O quinto conjunto de dados corresponde a Íris, já descrito no Grupo A.

Na Tabela 7.2 é mostrado um resumo das estruturas de cada conjunto de dados deste grupo, com exceção do conjunto Íris. A primeira coluna, Conjunto de dados, refere-se ao nome de cada conjunto. A segunda coluna, Objetos, refere-se ao número total de objetos de cada conjunto. A terceira coluna, Atributos, mostra a quantidade de atributos de cada objeto. E a última coluna, *Clusters*, refere-se a quantidade de *clusters* existentes.

Tabela 7.2 - Estruturas das bases do Grupo B

Conjunto de dados	Objetos	Atributos	*Classes*
Wine	178	13	3
Glass	214	9	6
Balance	625	4	3
Liverdisorder	345	6	2

7.2 Análise da robustez e eficiência dos algoritmos

Nesta seção, foi realizada uma análise de robustez e eficiência das meta-heurísticas estudadas neste trabalho. A primeira parte da análise tem o objetivo de verificar se um algoritmo executado diversas vezes chega a resultados parecidos. Isso porque as abordagens heurísticas tratadas neste trabalho contêm componentes aleatórias em suas várias fases, e se executados diversas vezes podem chegar a soluções diferentes. A idéia é que, mesmo se executado diversas vezes, um algoritmo não apresente soluções tão diferentes. A segunda parte da análise aborda a eficiência dos algoritmos através da comparação do índice CRand (CR) encontrado por todos os algoritmos para algumas bases de dados. Além do valor de CR, também será comparado o tempo de

execução deles.

7.2.1 Análise da robustez

Para realizar a análise da robustez, os quatro algoritmos (VNS, ILS, VNS-CS, ILS-CS) particionaram os conjuntos de dados do Grupo B(ver Tabela 7.2). Cada algoritmo foi executado 20 vezes para cada conjunto de dados, sendo K igual a classe real de cada conjunto. Dessas 20 execuções, foram selecionadas a solução máxima (s_{max}) e a solução mínima (s_{min}), ou seja, as soluções com maior e menor valor na função objetivo, respectivamente. E com esses valores foi possível avaliar o *gap* das soluções da seguinte forma:

$$gap = \frac{s_{max} - s_{min}}{s_{min}}. \tag{7.1}$$

Com o valor do *gap* é possível avaliar a amplitude da diferença entre as 20 soluções encontradas pelo algoritmo. Na Tabela 7.3 encontram-se os valores dos *gaps* calculados para cada algoritmo e para cada conjunto de dados. Os menores *gaps* estão em negrito.

Tabela 7.3 - Valores dos *gaps* encontrados

Conjunto de dados	VNS	ILS	VNS-CS	ILS-CS
Íris	0	0	0	0
Wine	0.001	0.022	**0**	**0**
Glass	0.032	0.027	**0.024**	**0.024**
Balance	0.001	0.001	0.001	0.001
Liverdisorder	0.001	0.011	**0**	**0**

Analisando os *gaps* é possível perceber que a amplitude da diferença entre as soluções é pequena em todos os algoritmos. Mas, mesmo assim, os algoritmos VNS-CS e ILS-CS se sobressaem pois conseguem $gap = 0$ para os conjuntos Wine e Liverdisorder, diferentemente dos outros. Com isso, pode-se concluir que, em comparação aos outros, os algoritmos VNS-CS e ILS-CS são os mais robustos.

7.2.2 Análise de eficiência

Para realizar a análise de eficiência, os quatro algoritmos (VNS, ILS, VNS-CS, ILS-CS) particionaram os conjuntos de dados do Grupo A (ver Tabela 7.1). Cada algoritmo foi executado 20 vezes para cada conjunto de dados com K *clusters*, onde K varia de 2 a 12.

Além disso, os algoritmos usaram quatro medidas de dissimilaridade diferentes: Distancia Euclidiana, Distância City-block, Cosseno e Correlação de Pearson, ambas descritas no Capítulo 2. Dentre as 20 execuções, somente a solução que obteve o menor valor na função objetivo (FO) é que foi considerada, pois trata-se de um problema de minimização. E com essa solução, foi calculado o CR. Essa metodologia de testes foi inspirada em Nascimento et al. (2010).

Os resultados reportados são aqueles em que K obteve o maior valor de CR. O valor de CR é calculado através do *software* R-Project desenvolvido para cálculos estatísticos. Ele é gratuito e pode ser obtido em http://www.r-project.org/.

As Tabelas 7.4, 7.6, 7.8 e 7.10 mostram os valores de K, FO e CR respectivamente, para Distância Euclidiana, Distância City-block, Cosseno e Correlação de Pearson, encontrados pelos quatro algoritmos. Os melhores CR estão em negrito.

As Tabelas 7.5, 7.7, 7.9 e 7.11 mostram, respectivamente, o tempo (em segundos) que o algoritmo leva para chegar na melhor solução ($tMelhor$) e o tempo total (em segundos) de execução do algoritmo ($tTotal$).

Tabela 7.4 - Resultados para Distância Euclidiana

Conjuntos de dados	VNS			ILS			VNS-CS			ILS-CS		
	K	FO	CR	K	FO	CR	K	FO	CR	K	FO	CR
Yeast	8	32455.49	0.143	9	28115.90	0.148	9	28113.65	**0.150**	9	28113.47	**0.150**
Breast	2	687524.83	0.877	2	687524.83	0.877	2	687524.83	0.877	2	687524.83	0.877
Proteínas	4	5860490.76	0.316	4	5860526.03	**0.322**	4	5860509.71	**0.322**	4	5860490.76	**0.322**
Íris	3	3409.73	0.756	3	3409.73	0.756	3	3409.73	0.756	3	3409.73	0.756

Tabela 7.5 - Tempos de execução - Distância Euclidiana

Conjuntos de dados	VNS		ILS		VNS-CS		ILS-CS	
	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$
Yeast	29.063	32.282	8.781	37.812	179.718	208.218	190.593	282.140
Breast	0.344	3.437	0.453	3.625	10.172	15.000	10.609	16.375
Proteínas	2.843	4.890	0.703	5.265	18.875	29.281	17.594	21.500
Íris	0.015	0.187	0.016	0.188	0.421	0.656	0.468	0.672

Analisando os resultados para Distância Euclidiana, pode-se concluir que tanto o $tMelhor$ quanto $tTotal$ para os algoritmos VNS-CS e ILS-CS é elevado, porém, esses algoritmos conseguem chegar ao melhor CR do conjunto Yeast em comparação aos outros algoritmos. O ILS-CS consegue os menores valores de função objetivo para Proteínas e Yeast.

Tabela 7.6 - Resultados para Distância City-block

Conjuntos de dados	VNS			ILS			VNS-CS			ILS-CS		
	K	FO	CR	K	FO	CR	K	FO	CR	K	FO	CR
Yeast	7	73713.17	**0.157**	7	73734.25	0.152	7	73712.80	**0.157**	7	73712.80	**0.157**
Breast	2	1414121	0.877	2	1414121	0.877	2	1414121	0.877	2	1414121	0.877
Proteínas	5	32402165.5	0.294	5	32404046.1	**0.302**	5	32402209.99	0.295	5	32402209.99	0.295
Íris	3	5745.6	0.818	3	5745.6	0.818	3	5745.59	0.818	3	5745.59	0.818

Tabela 7.7 - Tempos de execução - Distância City-block

Conjuntos de dados	VNS		ILS		VNS-CS		ILS-CS	
	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$
Yeast	23.938	29.922	7.125	32.922	170.718	186.811	146.203	216.406
Breast	0.281	3.438	0.453	3.641	10.375	15.235	10.125	15.156
Proteínas	1.547	5.515	1.812	6.016	24.813	41.860	25.360	31.750
Íris	0.016	0.172	0.016	0.187	0.453	0.719	0.453	0.687

Para a Distância City-block, novamente encontram-se tempos elevados nos algoritmos VNS-CS e ILS-CS, mas eles conseguem o melhor CR para Yeast, apesar do VNS também chegar nesse valor de CR em menor tempo. No conjunto Proteínas, o ILS consegue o melhor CR, mas é possível notar que VNS-CS e ILS-CS chegam no mesmo CR e nos menores valores de função objetivo.

Tabela 7.8 - Resultados para Cosseno

Conjuntos de dados	VNS			ILS			VNS-CS			ILS-CS		
	K	FO	CR	K	FO	CR	K	FO	CR	K	FO	CR
Yeast	9	2102.33	0.135	9	2102.33	0.135	9	2102.33	0.135	9	2102.33	0.135
Breast	3	8340.24	0.293	3	8340.24	0.293	3	8340.24	0.293	3	8340.24	0.293
Proteínas	4	1049.95	0.340	4	1050.24	**0.358**	4	1049.95	0.338	4	1049.95	0.349
Íris	3	8.16	0.941	3	8.16	0.941	3	8.16	0.941	3	8.16	0.941

Tabela 7.9 - Tempos de execução - Cosseno

Conjuntos de dados	VNS		ILS		VNS-CS		ILS-CS	
	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$
Yeast	32.610	34.688	13.125	40.156	56.344	232.156	135.546	207.296
Breast	0.532	4.391	0.484	4.594	12.218	21.264	14.125	19.297
Proteínas	0.485	4.906	3.313	5.157	17.577	22.000	14.797	18.000
Íris	0.016	0.188	0.016	0.203	0.406	0.656	0.500	0.641

Nos resultados para Cosseno, pode-se notar que os algoritmos chegam nos mesmos valores de CR, com exceção do ILS que consegue o melhor CR para o conjunto Proteínas.

Tabela 7.10 - Resultados para Correlação de Pearson

Conjuntos de dados	VNS			ILS			VNS-CS			ILS-CS		
	K	FO	CR	K	FO	CR	K	FO	CR	K	FO	CR
Yeast	9	9244.76	0.131	7	13325.39	0.131	7	13325.39	0.131	7	9244.76	**0.133**
Breast	3	34235.23	0.283	3	34235.23	0.287	3	34235.23	**0.290**	3	34235.23	0.288
Proteínas	4	2358.05	0.344	4	2358.05	0.344	4	2358.05	0.344	4	2358.05	0.344
Íris	3	21.94	0.886	3	21.94	0.886	3	21.94	0.886	3	21.94	0.886

Tabela 7.11 - Tempos de execução - Correlação de Pearson

Conjuntos de dados	VNS		ILS		VNS-CS		ILS-CS	
	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$	$tMelhor$	$tTotal$
Yeast	9.031	34.734	6.281	34.702	94.110	195.813	248.796	266.375
Breast	1.828	4.375	0.578	4.610	12.860	22.125	12.125	24.750
Proteínas	0.672	4.891	0.610	5.219	13.203	26.719	17.437	19.250
Íris	0.015	0.171	0.031	0.187	0.422	0.656	0.438	0.657

Para Correlação de Pearson, os algoritmos VNS-CS e ILS-CS apresentam melhores valores de CR. O VNS-CS consegue o melhor índice para o conjunto Breast e o ILS-CS para o Yeast.

Apesar do tempo de execução dos algoritmos VNS-CS e ILS-CS ser maior que do VNS e ILS, eles conseguem convergir para melhores resultados. Alguns conjuntos de dados apresentam bastante dificuldade em serem agrupados e isso pode ser observado nos valores de CR encontrados para Proteínas e Yeast, por exemplo. Se os algoritmos VNS-CS e ILS-CS necessitam de um pouco mais de tempo, mas conseguem chegar a melhores valores de CR, pode-se concluir que eles são mais eficientes, não por causa do tempo mas sim pela convergência.

7.3 Comparação de resultados

Nesta seção, os resultados do particionamento de cada conjunto de dados serão comparados com outros trabalhos da literatura. A metodologia e os resultados para cada grupo de conjuntos de dados são diferentes e por isso serão descritos em duas subseções.

Como analisado na seção anterior, os algoritmos VNS-CS e ILS-CS podem ser considerados mais robustos e eficientes em comparação aos outros. Mas, o algoritmo a ser utilizado nas comparações é o ILS-CS.

São duas metodologias de testes diferentes pois cada uma é referente a um trabalho da literatura. Essas metodologias já foram utilizadas nas análises de robustez e eficiência. Os testes foram executados em um Processador AMD Athlon Dual Core com memória de 2.5 GB de RAM e Sistema Operacional Windows XP.

O índice Rand R também foi calculado no *software* R-Project.

7.3.1 Comparação I

Neste primeira comparação, foram particionados os conjuntos de dados do grupo A. Como a metodologia dos testes foi inspirada em Nascimento et al (2010), também foram utilizadas diferentes medidas de dissimilaridade. São elas: Distância Euclidiana, Distância City-block, Correlação de Pearson não centralizado (Cosseno) e Correlação de Pearson. Essas medidas já foram apresentadas no Capítulo 2.

O ILS-CS foi comparado com GRASP (NASCIMENTO et al., 2010), k-Médias (MACQUEEN, 1967), k-Medianas (KAUFMAN; ROUSSEEUW, 1989) e PAM (KAUFMAN; ROUSSEEUW, 1989), todos utilizando as mesmas medidas de dissimilaridade.

Para cada conjunto de dados o ILS-CS gerou partições com K *clusters*, onde K varia de 2 a 12. O algoritmo foi executado 20 vezes e dentre essas execuções, somente a solução que obteve menor função objetivo foi considerada, pois este é um problema de minimização. Esta solução, então, foi avaliada pelo índice CRand (CR).

As Tabelas 7.12, 7.13, 7.14 e 7.15 mostram, respectivamente, os valores de K e CR para Distância Euclidiana, Distância City-block, Cosseno e Correlação de Pearson, encontrados pelo ILS-CS para cada conjunto de dados juntamente com os resultados obtidos pelos algoritmos GRASP, k-Médias, k-Medianas e PAM em Nascimento et al (2010).

Os resultados apresentados são aqueles em que o valor de K obteve maior valor do CR. A coluna Conjunto de dados indica o nome dos conjuntos de dados, enquanto as colunas K e CR indicam, respectivamente, o número de clusters e o CR encontrado para K. Os melhores valores de CR estão escritos em negrito.

Tabela 7.12 - Resultados para Distância Euclidiana

Conjuntos de dados	ILS-CS		GRASP		k-Médias		k-Medianas		PAM	
	K	CR	K	CR	K	CR	K	CR	K	CR
Yeast	9	0.150	9	0.150	7	0.170	6	**0.173**	8	0.143
Breast	2	**0.877**	2	**0.877**	2	0.803	2	0.782	2	0.828
Proteínas	4	**0.322**	4	**0.322**	7	0.322	7	0.313	6	0.250
Íris	3	**0.756**	3	**0.756**	3	0.730	3	0.744	3	0.730

Tabela 7.13 - Resultados para Distância City-block

Conjuntos de dados	ILS-CS		GRASP		k-Médias		k-Medianas		PAM	
	K	CR	K	CR	K	CR	K	CR	K	CR
Yeast	7	0.157	7	0.157	7	**0.181**	6	0.167	7	0.152
Breast	2	**0.877**	2	**0.877**	2	0.770	2	0.765	2	0.807
Proteínas	5	**0.295**	5	0.293	8	0.223	7	0.229	3	0.192
Íris	3	**0.818**	3	**0.818**	3	0.717	3	0.717	3	0.772

Tabela 7.14 - Resultados para Cosseno

Conjuntos de dados	ILS-CS		GRASP		k-Médias		k-Medianas		PAM	
	K	CR	K	CR	K	CR	K	CR	K	CR
Yeast	9	0.135	9	0.135	9	0.138	6	0.132	7	**0.146**
Breast	3	0.293	3	0.293	4	0.258	3	0.306	3	**0.332**
Proteínas	4	**0.349**	4	**0.349**	7	0.320	6	0.304	6	0.247
Íris	3	**0.941**	3	**0.941**	3	0.904	3	**0.941**	3	0.904

Analisando os resultados, percebe-se que o ILS-CS obteve os melhores resultados para alguns conjuntos, principalmente com a Distância City-block e de maneira geral, obteve bons resultados. Na Tabela 7.12, o ILS-CS teve 3 melhores resultados: Proteínas, Breast e Íris, encontrando o mesmo número de *clusters* que o número real para os três conjuntos de dados. Na Tabela 7.13, o ILS-CS obteve 3 melhores

Tabela 7.15 - Resultados para Correlação de Pearson

Conjuntos de dados	ILS-CS		GRASP		k-Médias		k-Medianas		PAM	
	K	CR	K	CR	K	CR	K	CR	K	CR
Yeast	7	0.133	9	0.131	8	0.135	8	0.133	7	**0.145**
Breast	3	0.288	3	0.284	2	**0.441**	2	0.368	2	0.289
Proteínas	4	**0.344**	4	**0.344**	7	0.313	7	0.306	6	0.245
Íris	3	0.886	3	0.886	3	0.886	3	**0.941**	3	0.886

resultados, sendo que para Proteínas, ele foi o único a obter o melhor resultado. Para a Distância Cosseno, Tabela 7.14, o ILS-CS encontra 2 melhores resultados: Proteínas e Iris. Na Tabela 7.15, o ILS-CS também encontra apenas 1 melhor resultado (Proteínas) e o mesmo número de *clusters* que o real.

Como se pode ver, o ILS-CS tem desempenho semelhante aos melhores algoritmos, chegando a ser o único a apresentar o melhor resultado em um dos casos. Comparando de forma geral, o ILS-CS é melhor que o GRASP, pois consegue os melhores valores de CR 9 vezes enquanto o GRASP consegue 8 vezes.

7.3.2 Comparação II

Neste segundo experimento, os conjuntos de dados utilizados foram os do grupo B. A métrica de distância, neste caso, foi a Distância Euclidiana.

Para cada conjunto de dados, o algoritmo foi executado 20 vezes, gerando 20 soluções. Essas soluções foram avaliadas através do índice Rand R. Para todos os conjuntos foram selecionados o R máximo, mínimo e o médio, entre as 20 soluções geradas. Como o Rand é sensível ao número de *clusters* das duas partições utilizadas no cálculo, o ILS-CS gerou as partições com o número de *clusters* real de cada conjunto de dados.

Para verificar o desempenho do ILS-CS, seus resultados foram comparados com os de Chang et al. (2009). No trabalho citado foram utilizados quatro algoritmos: k-Médias (MACQUEEN, 1967), GA-clustering (MURTHY; CHOWDHURY, 2003), KGA-clustering (BANDYOPADHYAY; MAULIK, 2002) e GAGR-clustering, sendo este último proposto pelos autores. Os conjuntos de dados utilizados pelos autores foram os

mesmos do grupo B.
A Tabela 7.16 mostra os resultados dos máximos, mínimos e médios valores de R encontrados. Os melhores resultados estão em negrito.

Analisando a Tabela 7.16, percebe-se que o ILS-CS tem desempenho melhor que os outros algoritmos, pois consegue os melhores Rands em 3 dos 5 conjuntos de dados.

Tabela 7.16 - Resultados do máximo, médio e mínimo R

Base de dados	k-Médias	GA-clustering	KGA-clustering	GAGR-clustering	ILS-CS
Íris					
Max	0.8623	0.8622	0.8737	0.8737	**0.8922**
Médio	0.8292	0.8518	0.8588	0.8711	**0.8922**
Min	0.7716	0.8564	0.8464	0.8623	**0.8922**
Wine					
Max	0.9265	0.9415	0.9415	0.9415	0.7191
Médio	0.8413	0.9368	0.9319	**0.9388**	0.7191
Min	0.7339	0.9246	0.9120	**0.9349**	0.7191
Glass					
Max	0.5107	0.5308	0.5367	0.5367	**0.7450**
Médio	0.3807	0.3878	0.3966	0.4047	**0.7356**
Min	0.2556	0.2759	0.2683	0.2736	**0.6714**
Balance					
Max	0.5826	0.5827	0.5834	0.6075	**0.6250**
Médio	0.5603	0.5642	0.5642	0.5646	**0.5859**
Min	0.5215	0.5345	0.5296	0.5320	**0.5667**
Liverdisorder					
Max	0.5004	0.5014	0.5043	**0.5050**	0.5016
Médio	0.4989	0.5004	0.5022	**0.5031**	0.5016
Min	0.4984	0.4993	0.5021	0.5021	0.5016

Além de ter obtido resultados melhores, o ILS-CS demandou menos tempo para os conjuntos de dados desse grupo, conforme pode ser visto na Tabela 7.17. Como foram geradas 20 soluções neste teste, os tempos mostrados na tabela são a média dos tempos das soluções, assim $tMelhor$ é a média dos 20 melhores tempos (tempo em que o algoritmo encontra a melhor solução) e $tTotal$ é a média do tempo total das 20 soluções. O conjunto que demandou mais tempo foi o Balance, justamente o

conjunto com maior número de objetos.

Tabela 7.17 - Tempos de execução - Grupo B

Conjuntos de dados	Iterações VNS	tMelhor(s)	tTotal(s)
Íris	50	0.490	0.664
Wine	50	0.696	1.006
Glass	50	1.648	2.285
Balance	50	11.932	18.033
Liverdisorder	50	2.675	4.292

7.4 Classificação de imagens

Uma outra aplicação interessante para essas abordagens heurísticas é a classificação de imagens, ou seja, trabalhar a classificação de uma imagem como um problema de agrupamentos, fazendo com que os objetos sejam os *pixels* da imagem e as regiões com diferentes cores (chamadas de padrões) sejam os *clusters*.

O objetivo da classificação de imagens é atribuir os *pixels* a determinados padrões, definidos através de algum método de classificação (neste caso, um algoritmo de agrupamento). A Figura 7.1 ilustra uma imagem com 4 padrões (4 *clusters*).

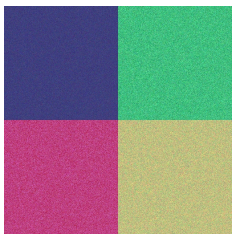

Figura 7.1 - Exemplo de uma imagem com 4 *clusters*

Para isso, foi utilizada uma imagem multiespectral (ou multibandas) coletada pelo satélite Landsat-5, no dia 29/06/2010 correspondente a um trecho da Amazônia, próximo a cidade de Santarém no Pará. Essa imagem foi obtida pelo sítio *http://www.dgi.inpe.br/CDSR*, onde fica o catálogo de imagens do Inpe. Ela vem composta por um número muito grande de *pixels*, por isso é necessário fazer um recorte e utilizar apenas uma amostra, que neste caso contem 100x100 *pixels*.

A Figura 7.2 ilustra a imagem em RGB a ser utilizada no experimento computacional composta pela banda 5 como vermelho (R), banda 4 como verde (G) e banda 3 como azul (B). As cores da imagem representam cada padrão, sendo o verde escuro uma região de floresta secundária, o verde mais claro, uma região de floresta primária, o rosa uma região de solo exposto, o preto um rio e a região formada por uma mistura de tons roxos e verdes (parecendo um azul claro) trata-se de terreno de pastagem.

Figura 7.2 - Amostra da imagem Landsat-5

O primeiro passo para o agrupamento da imagem é processá-la afim de extrair sua matriz de *pixels*. Essa matriz corresponde aos dados de entrada para o algoritmo de agrupamento (os objetos). O algoritmo a ser utilizado nessa aplicação é o ILS-CS.

Uma métrica de distância deve ser definida, pois o algoritmo de agrupamento a utiliza durante sua execução. Neste caso, a métrica a ser utilizada será a Distância Euclidiana dada pela Equação 2.1.

Os atributos utilizados para o agrupamento são os valores de RGB de cada pixel encontrados na matriz de *pixels*, ou seja, cada *pixel* terá 3 atributos: um valor para vermelho (R), um valor para verde (G) e um valor para azul (B). Esses valores variam de 0 a 255.

Como a imagem é composta por uma matriz de 100x100 *pixels* e o ILS-CS representa as soluções em um vetor, essa matriz é transformada em um vetor, assim obtendo 10.000 objetos.

O experimento computacional é feito com dois algoritmos: ILS-CS e k-Médias. A

imagem foi classificada com K variando de 2 a 10. Para cada K, o ILS-CS foi executado 5 vezes e somente a solução que obteve o menor valor na função objetivo é que foi utilizada. Para executar o k-Médias foi utilizada uma implementação já pronta e disponível no programa ENVI (ferramenta para processamento de imagens) e foi executado uma única vez para cada K. A Figura 7.3 ilustra o resultado para cada algoritmo e para cada K.

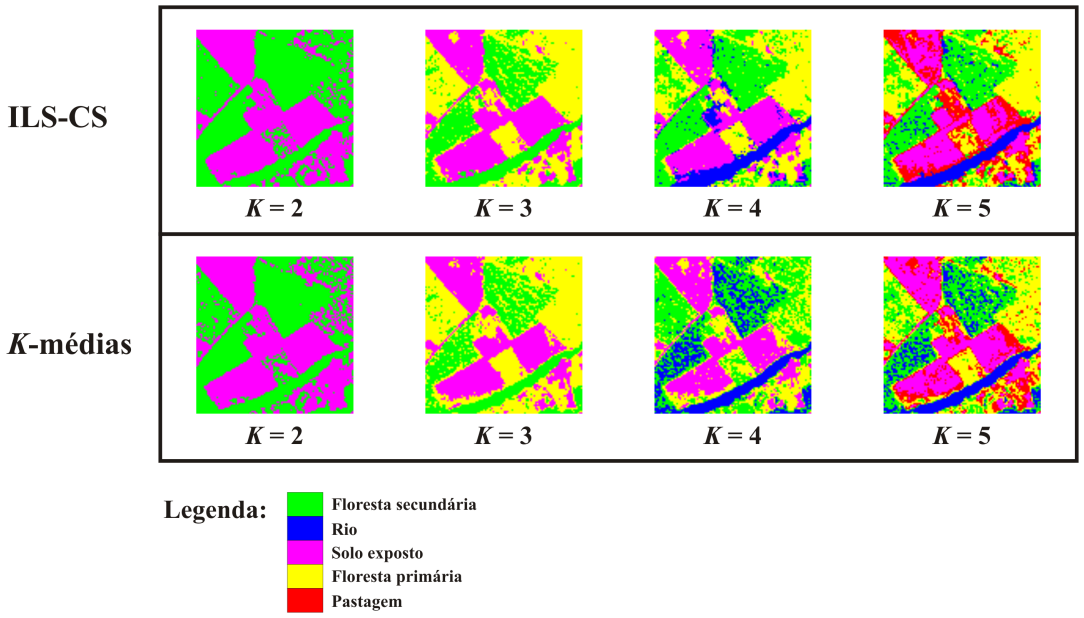

Figura 7.3 - Resultados da classificação da imagem

Nos resultados reportados, o valor de K foi até 5 porque os resultados com K acima deste valor apresentaram dificuldades em definir, visualmente, qual cor representa qual padrão.

Para validar o algoritmo e saber qual dos resultados é o melhor (qual K é melhor), foi utilizado o índice de concordância Kappa (CONGALTON; GREEN, 2008). Este índice é calculado entre a imagem original e a imagem classificada. Seu valor varia entre 0 e 1. Neste trabalho, o índice Kappa foi calculado no *software* ENVI. Quanto mais próximo de 1, mais concordantes são as duas imagens, ou seja, a classificação tem bom resultado.

A Tabela 7.18 mostra os resultados do índice Kappa para o ILS-CS e para o k-

Médias.

Tabela 7.18 - Índices Kappa encontrados para cada algoritmo e cada K

Algoritmo	$K = 2$	$K = 3$	$K = 4$	$K = 5$
k-Médias	0.6943	0.8962	0.6543	0.6671
ILS-CS	**0.8205**	**0.9386**	**0.9929**	**0.8554**

Os resultados mostram que o ILS-CS consegue os melhores valores para o índice Kappa. A imagem classificada com $K = 4$ obteve o maior valor de Kappa e com isso, conclui-se que com 4 padrões essa imagem é melhor classificada.

O tempo de execução do ILS-CS para $K = 4$ foi de 2145.35 segundos (aproximadamente 35 minutos) para chegar na melhor solução. E o tempo total de execução foi de 5999.04 segundos (aproximadamente 2 horas). Analisando o tempo de execução pode-se dizer que é bastante elevado. Para imagens maiores, esse tempo seria ainda maior. Como o k-Médias foi executado no ENVI e ele não fornece informações referente aos tempos de execução, não pode ser comparado.

Mesmo o ILS-CS tendo apresentado alto custo computacional, os resultados dele para essa imagem foram muito satisfatórios.

8 CONCLUSÃO E TRABALHOS FUTUROS

Os problemas de agrupamentos vem sendo bastante explorados na literatura. Diversos algoritmos foram e ainda vem sendo propostos para solucionar estes problemas como, por exemplo, o k-Médias que é um dos algoritmos clássicos e muito utilizado para este fim.

Recentemente, foi proposto o uso de meta-heurísticas na solução de problemas de agrupamentos, fazendo com que um problema de agrupamentos fosse transformado em um problema de otimização. E apresentaram bons resultados.

O objetivo deste trabalho foi propor diferentes abordagens heurísticas para solucionar problemas de agrupamentos, tanto para conjuntos de dados pequenos quanto para conjuntos grandes (como no caso do agrupamento em imagens). Depois de apresentados os resultados dos testes computacionais, foi possível perceber que para grandes conjuntos de dados, os algoritmos demandam mais tempo. Isso acontece porque com grande número de objetos (o que leva a ter grande número de posições no vetor de solução) o algoritmo necessita realizar um número maior de movimentos para se chegar em boas soluções e isto implica em aplicar busca local, função objetivo, processo de agrupamento, e etc, mais vezes, fazendo com que o algoritmo consiga convergir, mas com alto custo computacional.

Pode-se exemplificar usando as duas meta-heuristicas clássicas, VNS e ILS, e as duas meta-heurísticas híbridas, VNS-CS e ILS-CS. Note que para as meta-heurísticas clássicas o tempo de execução dos algoritmos é menor em comparação as outras. Isso se justifica pelo fato de que elas realizam menos movimentos nas soluções, ou seja, oferecem menor diversificação nas soluções e tem menor intensificação da busca local, fazendo com que o tempo de execução seja baixo, mas que nem sempre cheguem a bons resultados. Já as meta-heurísticas híbridas oferecem grande diversificação e uma boa intensificação do processo de busca, obtendo bons resultados (alguns até melhores que das meta-heurísticas clássicas) porém demandam bastante tempo.

Outro fator importante no agrupamento são os atributos dos conjuntos de dados. Esses atributos são fundamentais pois é por meio deles (são utilizados no cálculo da medida de proximidade) que se faz a separação dos objetos em *clusters*. Em alguns casos, os resultados ruins não são justificados pela ineficiência dos algoritmos mas sim pela dificuldade que os atributos apresentam em separar os objetos em *clusters*.

Isso é observado através dos resultados que os algoritmos obtêm para os conjuntos Íris e Yeast, por exemplo. O conjunto Íris tem atributos que permitem realizar o agrupamento facilmente (ver gráficos da Figura 8.1), visto que para todas as meta-heurísticas abordadas neste trabalho, os resultados do CR para ele foram os mesmos e quando comparados com outros algoritmos da literatura, os resultados para Íris são bons, em sua maioria. Ao analisar os resultados para o conjunto Yeast, percebe-se a grande dificuldade em separar seus objetos em *clusters* (ver gráficos da Figura 8.2). Para Yeast, as meta-heurísticas abordadas neste trabalho encontram baixos valores de CR (somente as meta-heurísticas híbridas conseguem obter bons resultados em todos os testes) e quando comparados com outros trabalhos da literatura percebe-se que eles também encontram baixos valores de CR, o que leva a concluir que Yeast realmente apresenta dificuldade em ser particionado.

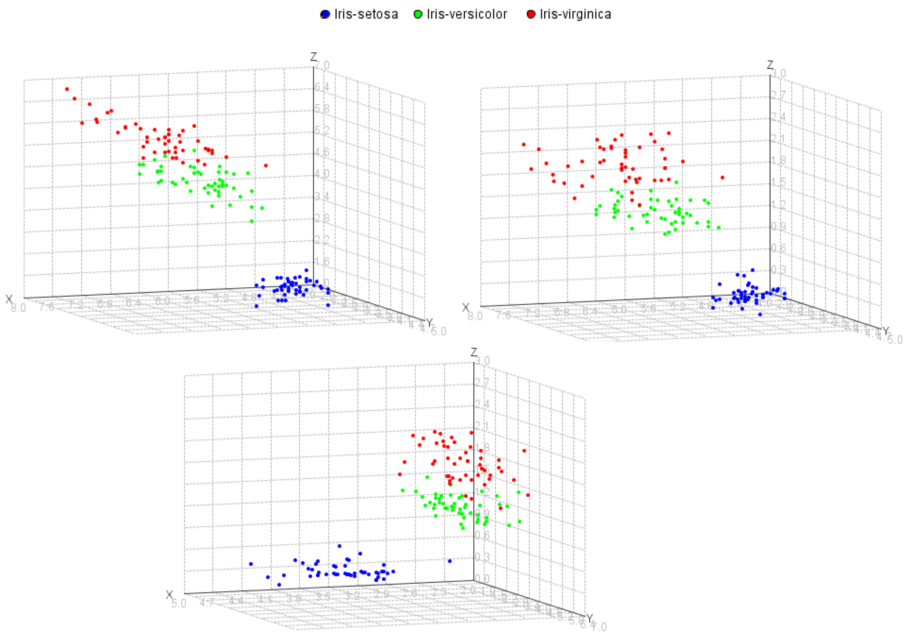

Figura 8.1 - Gráficos gerados a partir de atributos diferentes do conjunto Íris

Com relação ao teste em agrupamento de imagens, a meta-heurística híbrida ILS-CS obteve resultados bastante satisfatórios quando comparada ao k-Médias. O teste em imagens demanda bastante tempo, pois o número de objetos é grande. Para se ter uma idéia, uma imagem de 100x100 *pixels* resulta em 10.000 objetos. Já uma imagem de 200x200 resultaria em 40.000 objetos, necessitando de mais tempo ainda

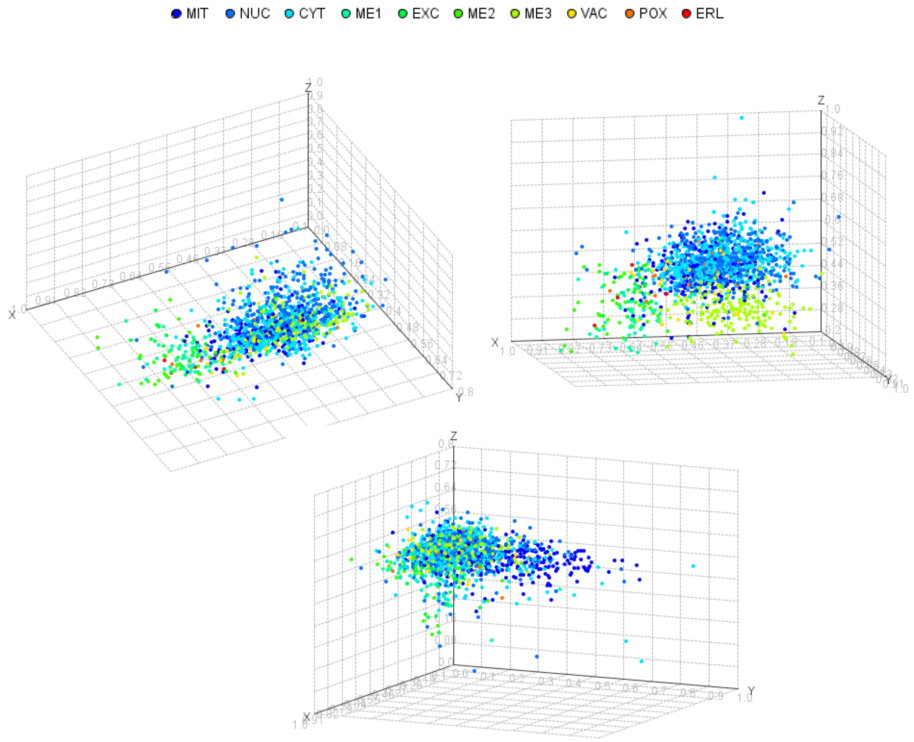

Figura 8.2 - Gráficos gerados a partir de atributos diferentes do conjunto Yeast

para ser classificada.

Diante disso, pode-se concluir que as abordagens heurísticas propostas neste trabalho, em especial as meta-heurísticas híbridas, podem ser consideradas bons algoritmos de agrupamento.

8.1 Trabalhos futuros

Primeiramente, seria importante fazer melhorias nas meta-heurísticas híbridas (que obtiveram melhores resultados) afim de diminuir o tempo computacional que elas demandam nos particionamentos. Para isso, é necessário intensificar ainda mais o processo de busca dentro do processo de agrupamento (do CS), implementando outras rotinas de busca local. Uma sugestão é combinar diferentes métodos de busca local através da meta-heurística Descida em Vizinhança Variável (VND, do inglês *Variable Neighborhood Descent*)(MLADENOVIć; HANSEN, 1997) que consiste em explorar o espaço de busca por meio de trocas sistemáticas de estruturas de vizinhança (neste caso seria a troca dos métodos de busca local), aceitando somente soluções de melhora da solução corrente e retornando a primeira estrutura (retornando ao

primeiro método) quando uma solução melhor é encontrada.

Outra sugestão é fazer uma otimização nos algoritmos com relação à memória utilizada por eles. Isso seria interessante pois poderiam realizar o agrupamento em imagens com maior número de *pixels* sem a necessidade de aumentar a capacidade do *hardware*.

A aplicação em classificação de imagens rendeu bons resultados o que motiva a aplicar o ILS-CS em outras imagens e comparar com outros algoritmos de classificação de imagens.

REFERÊNCIAS BIBLIOGRÁFICAS

AL-SULTAN, K. A tabu search approach to the clustering problem. **Inf. Sci. Appl.**, v. 28, p. 1443–1451, 1995. 18

AMORIM, S.; BARTHéLEMY, J.; RIBEIRO, C. Clustering and clique partitioning: Simulated annealing and tabu search approaches. **Journal of Classification**, v. 9, p. 17–41, 1992. 13

ASUNCION, A.; NEWMAN, D. **UCI Machine learning repository**. 2007. Disponível em: <http://www.ics.uci.edu/~mlearn/MLRepository.html>. Acesso em: 05 de maio de 2010. 46

BANDYOPADHYAY, S.; MAULIK, U. An evolutionary technique based on k-means algorithm for optimal clustering in RN. **Information Sciences**, v. 146, p. 221–237, 2002. 17, 55

BARBARA, D. **An introduction to cluster analysis for data mining**. 2000. Disponível em: <http://www-users.cs.umn.edu/~han/dmclass/cluster_survey_10_02_00.pdf>. Acesso em: 26 de abril de 2010. 5, 6, 7, 9, 12

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization overview and conceptual comparison. **ACM Computing Surveys**, v. 35, n. 3, p. 268–308, 2003. 19, 26

CHANG, D.-X.; ZHANG, X.-D.; ZHENG, C.-W. A genetic algorithm with gene rearrangement for k-means clustering. **Pattern Recognition**, v. 42, n. 7, p. 1210 – 1222, 2009. 2, 17, 46, 55

CHAVES, A. **Uma Meta-heurística híbrida com Busca por Agrupamentos aplicada a problemas de otimização combinatória**. Tese (Doutorado em Computação Aplicada) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos - SP, 2009. 2, 31, 32, 33, 38, 39, 40

CHAVES, A. A.; CORRÊA, F. de A.; ; LORENA, L. A. N. Clustering search heuristic for the capacitated p-median problem. **Springer Advances in Software Computing Series**, v. 44, p. 136–143, 2007. 31

CHAVES, A. A.; LORENA, L. A. N. Hybrid metaheuristic for the prize collecting travelling salesman problem. **Lecture Notes in Computer Science**, v. 4972, p. 123–134, 2008. 31

_____. Clustering search algorithm for the capacitated centered clustering problem. **Computers & Operations Research**, v. 37, p. 552–558, 2010. 31

CHAVES, A. A.; LORENA, L. A. N.; MIRALLES, C. Hybrid metaheuristic for the assembly line worker assignment and balancing problem. **Lecture Notes in Computer Science**, v. 5818, p. 1–14, 2009. 31

CONGALTON, R. G.; GREEN, K. **Assessing the accuracy of remotely sensed data : principles and practices**. USA: CRC Press, 2008. 183 p. 59

CORRÊA, F. de A.; CHAVES, A. A.; LORENA, L. A. N. Hybrid heuristics for the probabilistic maximal covering location-allocation problem. **Operational Research**, v. 7, p. 323–343, 2007. 31

EVERITT, B.; LANDAU, S.; LEESE, M. **Cluster analysis**. London: Wiley, 2001. 10

FACELI, K.; CARVALHO, A. C.; SOUTO, M. C. P. **Validação de algoritmos de agrupamento**. 2005. Disponível em: <ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_254.pdf>. Acesso em: 03 de maio de 2010. 15

FEO, T.; RESENDE, M. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, p. 109133, 1995. 2, 17

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers & Operations Research**, v. 5, p. 553549, 1986. 17, 18

_____. Tabu search and adaptive memory programing: advances, applications and challenges. In: BARR, R. S.; HELGASON., R. V.; KENNINGTON, J. L. (Ed.). **Interfaces in computer ccience and operations research.** Norwell: Kluwer, 1996. p. 1–75. 39

GOLDBERG, D. E. **Genetic algorithms in search: optimization and machine learning**. Berkeley: Addison-Wesley, 1989. 223 p. 2

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. Cluster validity methods: Part i. **SIGMOD Record**, v. 31, n. 2, 2002a. 13, 14, 15

_____. Cluster validity methods: Part ii. **SIGMOD Record**, v. 31, n. 3, 2002b. 15

HANSEN, P.; JAUMARD, B. Cluster analysis and mathematical programming. **Mathematical Programming**, v. 79, p. 191–215, 1997. 6

HANSEN, P.; MLADENOVIć, N. Variable neighborhood search: Principles and applications. **European Journal of Operational Research**, v. 13, n. 3, p. 449 – 467, 2001. 24

HUBERT, L.; ARABIE, P. Comparing partitions. **Journal of Classification**, v. 2, p. 193–218, 1985. 3, 14, 15

JAIN, A.; MURTY, M.; FLYNN, P. Data clustering: A review. **ACM Computering Surveys**, v. 31, n. 3, p. 264–323, 1999. 6, 7, 12, 13

JAIN, A. K.; DUBES, R. C. **Algorithms for clustering data**. New Jersey: Prentice Hall, 1988. 304 p. 6, 7, 8, 9, 11, 13, 15

KAUFMAN, L.; ROUSSEEUW, P. **Finding groups in data: an introduction to cluster analysis**. 9th. ed. New York: Wiley, 1989. 342 p. 17, 53

KIRKPATRICK, S.; GELLAT, D.; VECCHI, M. Optimization by simulated annealing. **Science**, v. 220, p. 671680, 1983. 18

KOCHENBERGER, G.; GLOVER, F.; ALIDAEE, B.; WANG, H. Clustering of microarray data via clique partitioning. **Journal of Combinatorial Optimization**, v. 10, p. 77–92, 2005. 1, 13, 17

LOURENÇO, H. R.; MARTIN, O. C.; STüTZLE, T. Iterated local search. In: **Handbook of Metaheuristics**. New York: Springer: Fred Glover and Gary A. Kochenberger, 2003. p. 320–353. 26, 27

MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In: CAM, L. M. L.; NEYMAN, J. (Ed.). **Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability**. Berkeley: University of California Press, 1967. v. 1, p. 281–297. 2, 12, 17, 53, 55

MEHROTRA, A.; TRICK, M. Cliques and clustering: A combinatorial approach. **Operations Research Letters**, v. 22, p. 1–12, 1998. 13

METZ, J. **Interpretação de clusters gerados por algoritmos de *clustering* hierárquico**. Dissertação de Mestrado — Universidade de São Paulo (USP), São Carlos - SP, 2006. 9, 10, 15

MLADENOVIć, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, v. 24, n. 11, 1997. 2, 23, 63

MURTHY, C.; CHOWDHURY, N. In search of optimal clusters using genetic algorithms. **Pattern Recognition Letters**, v. 17, p. 825–832, 2003. 17, 55

NASCIMENTO, M.; TOLEDO, F.; CARVALHO, A. Investigation of a new grasp-based clustering algorithm applied to biological data. **Computers & Operations Research**, v. 37, n. 8, p. 1381 – 1388, 2010. 1, 2, 17, 45, 46, 49, 53

OLIVEIRA, A. **Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuo e discreto**. Tese (Doutorado em Computação Aplicada) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos - SP, 2004. 31, 39

RAND, W. M. Objective criteria for the evaluation of clustering methods. **Journal of the American Statistical Association**, v. 60, p. 846–850, 1971. 3, 14

SENNE, E. L. F.; LORENA, L. A. N. Abordagens complementares para problemas de p-medianas. **Produção**, Scielo, v. 13, p. 78 – 87, 2003. 35

SILVA, L. R. S. **Aprendizagem participativa em agrupamento nebuloso de dados**. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Estadual de Campinas, Campinas - SP, 2003. 11

STüTZLE, T. Iterated local search for the quadratic assignment problem. **TU Darmstadt**, 1999. 26

XU, R.; WUNSCH, D. Survey of clustering algorithms. **IEEE Transactions on Neural Networks**, v. 16, n. 3, p. 645–678, 2005. 1, 6, 9, 10