Universidade Fererederal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Metaheurística híbrida para o problema de clusterização automática

Gisele Paola Lucioli Silva

JUIZ DE FORA
JULHO, 2017

Metaheurística híbrida para o problema de clusterização automática

Gisele Paola Lucioli Silva

Universidade Ferederal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Prof. Stênio Sã Rosário Furtado Soares

JUIZ DE FORA

JULHO, 2017

Metaheurística híbrida para o problema de clusterização automática

Gisele Paola Lucioli Silva

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEREDERAL DE JUIZ DE FORA , COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Prof. Stênio Sã Rosário Furtado Soares
D. Sc. UFJF

Prof[a]. Luciana Brugiolo Gonçalves
D. Sc. UFJF

Prof. Victor Stroele de Andrade Menezes
D. Sc. UFJF

JUIZ DE FORA

5 DE JULHO, 2017

Aos meus amigos e irmão.

Aos pais, pelo apoio e sustento.

Resumo

A automação de serviços por meio de sistemas e a ampla capacidade de se armazenar dados atualmente gera um volume de informações cada vez maior. Este fato também desperta o interesse em extrair informações valiosas destes dados. Porém, no atual cenário, o que se verifica é um volume de dados que já extrapola a capacidade da mente humana de processar tanta informação, o que motiva o desenvolvimento de técnicas computacionais para extração destas informações e geração de conhecimento. Neste contexto, o Problema de Clusterização Automática - PCA - consiste em encontrar, automaticamente, agrupamentos de dados que se assemelham mutuamente. No presente trabalho, foi desenvolvida uma técnica para o problema da clusterização automática utilizando uma combinação das heurísticas ILS (*Iterated Local Search*) e VND (*Variable Neighborhood Descent*). A comparação dos resultados com algoritmos da literatura mostrou que a técnica proposta encontrou uma solução melhor ou igual em mais de 35% dos casos.

Palavras-chave: Metaheurística, Clusterização, PCA, ILS, VND.

Abstract

The automation of services through systems and the ample capacity to store data generates an increasing volume of information. This also raises the interest in extracting valuable information from these data. However, in the current scenario, what is verified is a volume of data that already exceeds the capacity of the human mind to process so much information, which motivates the development of computational techniques for extracting this information and generating knowledge. In this context, the Automatic Clustering Problem is defined as automatically finding mutually similar groupings of data. In the present work, a technique was developed for the problem of automatic clustering using a combination of ILS (Iterated Local Search) and VND (Variable Neighborhood Descent) heuristics. Comparing the results with algorithms of the literature showed that the proposed technique found a better or equal solution in more than 35% of cases.

Keywords: Metaheuristic, Clustering, PCA, ILS, VND.

Agradecimentos

De todos aqueles que me ajudaram, quero começar agradecendo a minha mãe Elaine e meu padrasto Anilton, pelo apoio e sustento durante a universidade.

Em seguida, ao meu grande amigo Stênio, não somente pela orientação do trabalho, mas por toda motivação, pela amizade e pela credibilidade durante o curso.

A professora Luciana que contribuiu com várias ideias para este trabalho.

Ao Gustavo Seeman, que foi extremamente solícito ao enviar as instâncias utilizadas em seu trabalho.

Ao meu irmão Guilherme, a mamãe de criação Tereza, e os tios Albino, Rossana, Mércia e Alfredo pelos ensinamentos na infância e por me instigarem a curiosidade.

À minha sobrinha Maria Luisa, e os meus afilhados João e Pedro por me despertarem a vontade de ser uma pessoa melhor.

Aos amigos e namorado pelo encorajamento, apoio, compreensão nas minhas ausências e por não me deixarem desistir dos meus sonhos nas horas de dificuldade.

Agradeço aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Aos amigos do Cead e da coordenação, dos meus tempos de bolsista.

"A educação é a arma mais poderosa que você pode usar para mudar o mundo."

Nelson Mandela

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

PC - Problema da Clusterização

PCA - Problema da Clusterização Automática

PAA - Problema de Agupamento Automático

ILS - Iterated Local Search

VND - Variable Neighborhood Descent

1 Introdução

A automação de serviços por meio de sistemas, junto à ampla capacidade de armazenamento de dados atualmente, gera um grande volume de informações disponíveis, despertando o interesse em extração de conhecimento a partir desses dados, cuja aplicação se dá em diversas áreas, como bioinformática, marketing, computação visual e gráfica, computação médica, redes de comunicações, engenharia de transportes, redes de computadores, logística, dentre outras.

A área que explora a extração de conhecimento desses dados é denominada Minerações de Dados. No universo desta área, uma possível abordagem, denotada clusterização, consiste em encontrar K grupos de objetos em uma base de dados de tal forma que elementos de um mesmo grupo apresentem elevado grau de semelhança, enquanto elementos de grupos diferentes apresentem elevado grau de dissimilaridade.

A partir da aplicação de algoritmos de clusterização sobre uma base de dados, é possível analisar a base mais detalhadamente ao se considerar as características de elementos que pertencem a um mesmo grupo. Assim, esta base pode ser representada por diferentes grupos, de forma que cada grupo contém uma parte dos dados originais e pode ser visto como uma base de dados menor, conforme o interesse do especialista da área.

Em diversas aplicações, a natureza dos dados não permite que se saiba de antemão o número K de grupos em que a base de dados deve ser particionada, o que aumenta a dificuldade do problema. Assim, o algoritmo deverá encontrar o melhor particionamento automaticamente, além de dividir os dados nos grupos segundo o grau de semelhança e diferença dos objetos. Os problemas de clusterização em que o número de clusters deve ser determinado pelo algoritmo é denominado como Problema de Clusterização Automática (PCA).

O PCA é um problema de otimização combinatória, e, portanto, pode ser resolvido enumerando todas as soluções possíveis e escolhendo aquela com melhor valor para a função objetivo. Entretanto, esta maneira de se resolver um problema de otimização combinatória é inviável para problemas reais, pois o número de possíveis soluções cresce

exponencialmente em relação ao tamanho da entrada. Portanto, para resolver este problema pode-se utilizar de uma metodologia que explore o espaço de solução de forma a não precisar explorar exaustivamente todo o espaço de soluções, mas somente as regiões mais promissoras, ainda que não se garanta encontrar a solução ótima. As metaheurísticas são algoritmos heurísticos com esta característica.

O presente trabalho propõe o desenvolvimento de uma metaheurística híbrida para solucionar o Problema de Clusterização Automática a partir da combinação de duas metaheurísticas: Busca Local Iterada, ou *Iterated Local Search* (ILS) e Descida em Vizinhança Variável, ou *Variable Neighborhood Descent* (VND), encontrando resultados competitivos com resultados encontrados na literatura tanto na qualidade da solução quanto no tempo.

1.1 Definição do problema

Dada uma base de dados O contendo n objetos e um inteiro k, tal que $1 \leq k \leq n$, particione O em k grupos, denotados clusters, de forma que objetos semelhantes estejam em um mesmo cluster e objetos dissimilares estejam em clusters diferentes.

Em muitas áreas do conhecimento, o especialista não tem, a priori, informação sobre o número k de clusters que melhor caracteriza a base de dados. Neste contexto, o presente trabalho considera que o valor de k deve ser encontrado pelo próprio algoritmo. Tal restrição, acrescida das restrições do PC, definem o Problema de Clusterização Automática - PCA.

Seja uma base de dados representada por um conjunto de objetos $O = \{o_1, o_2, .., o_n\}$, tal que cada objeto o_i é representado por uma tupla de atributos $(o_{i,0}, o_{i,1}, o_{i,2}, \ldots, o_{i,m})$. Para que o processo de agrupamento seja realizado, deve-se considerar uma métrica de similaridade entre objetos. Em geral, esta métrica é estabelecida conforme a natureza dos atributos e o interesse da aplicação.

Assim, o problema consiste em obter $1 \leq k \leq n$ subconjuntos C_i de O tal que:

$$C_i \cap C_j = \emptyset, \quad \forall\, i, j \in \{1, \ldots,\ k\} \tag{1.1}$$

$$\bigcup_{i=1}^{k} C_i = O, \tag{1.2}$$

de tal forma que o somatório das dissimilaridades entre elementos de um mesmo cluster seja minimizado e o somatório das dissimilaridades de elementos de clusters diferentes seja maximizado.

1.2 Justificativa

A grande variedade de áreas em que o problema de clusterização tem aplicação, bem como a sua complexidade computacional, dificultam o estabelecimento de metodologias de propósito geral. O fato do problema de clusterização automática não pertencer à classe de problemas polinomiais sugere que o uso de abordagens heurísticas para o mesmo apresente-se como uma boa estratégia na obtenção de soluções que, embora não se assegure a otimalidade, apresentam boa qualidade em um tempo adequado para aplicações práticas.

1.3 Hipótese

Dado um conjunto O contendo n objetos caracterizados por m atributos, existe um particionamento dos elementos de O em k partições, denotadas clusters de forma a maximizar o somatório das similaridade entre objetos de um mesmo cluster e a minimizar o somatório de similaridade entre objetos de clusters diferentes.

1.4 Objetivo

Este trabalho tem por objetivo o desenvolvimento de uma metaheurística híbrida para o Problema da Clusterização Automática, obtendo soluções de boa qualidade em tempo aceitável quando comparada a algoritmos da literatura.

Este trabalho está organizado da seguinte maneira, o capítulo 2 apresenta a fundamentação teórica das abordagens utilizadas. O capítulo 3 apresenta como foi realizada a implementação da parte prática deste trabalho. Em seguida, o capítulo 4 apresenta os experimentos computacionais. No capítulo 5 é apresentado a conclusão e as propostas

para trabalhos futuros. Este trabalho possui, ainda, três apêndices, sendo que o apêndice A apresenta tabelas de análise da solução inicial e final do algoritmo. O apêndice B apresenta os melhores resultados obtidos pelo algoritmo desenvolvido na parte prática deste trabalho. E por fim, o apêndice C apresenta, graficamente os resultados das instâncias de 2 dimensões utilizadas nos experimentos computacionais.

2 Aspectos Teóricos

Este capítulo é dedicado a apresentação dos principais aspectos teórico do problema de clusterização e de clusterização automática, além de algumas técnicas utilizadas neste trabalho.

2.1 Clusterização

Clusterização é o termo genérico para um processo que une objetos similares de uma base de dados em um mesmo grupo [4]. Cada grupo é chamado de cluster, e, para o Problema da Clusterização (PC), o número de clusters é uma informação previamente conhecida.

Clusterização de dados é o processo de reunir vetores multidimensionais em grupos, tendo em vista obter a maior similaridade dentro de cada grupo [17]. Os objetos de um mesmo cluster devem ser mais parecidos entre si do que com os elementos de outros clusters [7]. Em outras palavras, o objetivo no problema de clusterização é maximizar a homogeneidade dentro de cada cluster e a heterogeneidade entre clusters diferentes [12].

O Problema da Clusterização Automática (PCA) ou Problema de Agrupamento Automático (PAA) é uma generalização do problema descrito acima, quando não é previamente conhecido o número ideal de clusters. Assim, o algoritmo deve encontrar o melhor número de clusters para a instância dada.

O PC e o PCA podem ser aplicados em diversos cenários reais. Em [6] são apresentadas algumas aplicações reais que estão associadas a dados do IBGE (Instituto Brasileiro de Geografia e Estatística). Outras aplicações também podem ser encontradas no contexto escolar [21], em objetos de aprendizagem [5], processamento de imagens, emparelhamento de genes, auxílio no diagnóstico de doenças e mineração de dados [22]. Segundo [31], a clusterização oferece estratégias para uma melhor solução na organização ou análise de um grupo de objetos.

Classificar algoritmos de clusterização acaba tornando-se uma tarefa difícil e polêmica, visto que muitas vezes não podemos dizer que um algoritmo constrói a cluste-

rização utilizando apenas uma das técnicas conhecidas.[28]

Contudo, existem diversos métodos de clusterização como: hierárquicos, particionais, baseados em densidade, baseados em grade, baseados em lógica *fuzzy*, métodos baseados em grafos, métodos baseados em computação evolucionária, a combinação de mais de um método, entre outros.

Os algoritmos de agrupamento baseados em densidade têm como objetivo a determinação de grupos (regiões) de alta densidade de objetos separados por regiões de baixa densidade. [26]

No modelo hierárquico, pode-se ter a construção da solução partindo-se de um único cluster até ter-se cada objeto como um cluster (*top-down*), ou ainda, partindo-se de uma solução em que cada objeto é um cluster e, por um processo de aglutinação, novos clusters são formados até ter-se um único cluster (hierárquico *botom-up*). Na prática, é comum a utilização de abordagens mistas.

Algoritmos com métodos de particionamento, classificam n pontos do conjuntos de entrada em k grupos de tal forma que cada grupo contenha no mínimo um ponto e que cada ponto pertença a um único grupo.[28]

Tanto o Problema de Clusterização quanto a variação automática pertencem a classe NP-Completo. Entretanto, o fato do PCA requerer que se encontre o número ideal de clusters aumenta consideravelmente o tamanho do espaço de solução [2].

Existem três questões fundamentais a serem consideradas na aplicação de análise de agrupamento, a saber: como será medida a similaridade dos dados, quantos grupos formar e como formar o agrupamento [25]. As três próximas seções discutem cada um destes aspectos.

2.2 Medidas de Similaridade

Como apresentado na definição do problema, o agrupamento deve deixar objetos similares no mesmo cluster e objetos dissimilares em clusters distintos, fazendo-se necessário a utilização de alguma métrica para a determinação da dissimilaridade ou similaridade entre dois objetos. Existem diversas métricas, mas o uso das mesmas, em geral, requer que se pré-estabeleça ou se conheça quais os tipos de dados suportados pela aplicação,

bem como qual o método usado no processo de obtenção da solução [1].

Os métodos baseados em densidade não podem ser usados em espaços discretos ou euclidianos. Já os métodos baseados em distância são mais abrangentes, e tornaram-se muito populares na literatura porque podem ser usados com quase todos os tipos de dados, contanto que uma função de distância apropriada para o tipo de dado seja empregada. Segundo [1], o problema de clusterização pode ser reduzido ao problema de encontrar funções de distâncias para um tipo de dado.

A distância euclidiana, apresentada na Equação 2.1, é a métrica mais frequentemente empregada nos casos em que todas as variáveis são quantitativas. Esta métrica é utilizada para calcular medidas específicas, assim como a distância euclidiana simples e a distância euclidiana quadrática ou absoluta [25].

A distância euclidiana simples entre dois objetos x_i e x_j no espaço m-dimensional pode ser obtida através da Equação 2.1.

$$d(x_i, x_j) = \sqrt[2]{\sum_{k=1}^{m}(x_{i,k} - x_{j,k})^2} \tag{2.1}$$

Esta métrica de similaridade funciona bem para um conjunto de dados compacto ou isolado. Um ponto negativo para esta abordagem é a tendência de o atributo de maior escala dominar o de menor escala. Uma solução para este problema é a normalização contínua dos dados. [13].

A normalização de dados é um ajuste na escala de valores dos atributos para um mesmo intervalo, permitindo que duas amostras sejam apropriadamente comparadas. A normalização minimiza os problemas derivados do uso de unidade e dispersões distintas entre as variáveis.

Uma abordagem de normalização é a normalização linear, que leva todos os dados ao intervalo [0,1]. Para tanto, em cada atributo de cada objeto é feito uma transformação de cada valor v em um valor normalizado $v^{'}$ através da Equação 2.2, onde MIN e MAX consistem no menor e maior valor daquele atributo em toda instância, respectivamente.

$$v^{'} = \frac{v - MIN}{MAX - MIN}. \tag{2.2}$$

Já para o caso em que os atributos dos objetos da base são dados categóricos, uma outra abordagem utilizada é a distância de Hamming (DH), pela qual, para cada atributo i, a distância entre dois objetos a e b, inicialmente em zero, é incrementada em uma unidade caso eles sejam diferentes neste atributo. A Equação 2.3 apresenta esta métrica para dois objetos a e b representados por m atributos categóricos.

$$DH(a,b) = \sum_{i=1}^{m} dh(a_t, b_t) \quad onde \quad dh(a_t, b_t) = \begin{cases} 0, \text{ se } a_t = b_t \\ 1, \text{ se } a_t \neq b_t \end{cases} \tag{2.3}$$

2.3 Número de Clusters e Análise do Agrupamento

O problema da clusterização automática requer que se decida quanto ao número ideal de clusters da solução. Portanto, precisa de um método para avaliar a qualidade da solução. Na literatura, pode-se encontrar vários métodos distintos para determinar o número de clusters, como o Calinski and Harabasz's method [18]; o Hartigan's method [11]; o Krzanowski and Lai's method [14]; o Gap method [30]; e o Jump method [29].

O método escolhido para este trabalho foi o índice da silhueta [23], apresentado a seguir.

Índice Silhueta

A análise de cluster através da silhueta foi proposta em [23], com a ideia de analisar graficamente o resultado de uma clusterização e auxiliar na escolha do número "apropriado" de clusters.

Por esta abordagem, para cada objeto i, o valor de silhueta s(i) é calculado baseado na coesão e na dispersão de cada cluster. O agrupamento inteiro é exibido combinando as silhuetas em um único gráfico, permitindo assim uma avaliação da qualidade relativa dos agrupamentos e uma avaliação da configuração dos dados. A largura média da silhueta fornece uma medida de validação do agrupamento, e pode ser usada para se inferir um número adequado de clusters presentes no conjunto de objetos de entrada.

Seja i um objeto pertencente ao cluster C_w, então $a(i)$ é a dissimilaridade do objeto i para cada outro objeto j de C_w. Se C_w possui um único elemento, então $a(i)$ é

fixado como zero, mas se C_w possuir mais de um elemento, deve ser calculado como na Equação 2.4.

$$a(i) = \frac{1}{|C_w|} \sum_{j=1}^{|} C_w | d_{i,j} \forall j \neq i \tag{2.4}$$

A distância do objeto $i \in C_w$ para cada cluster C_v, com $v \neq w$ é dada pela Equação 2.5:

$$d(i, C_v) = \frac{1}{|C_v|} \sum_{i=1}^{|} C_v | d_{i,j} \tag{2.5}$$

A dissimilaridade $b(i)$ do objeto i para cada outro objeto j do seu cluster mais próximo é dada pela Equação 2.6.

$$b(i) = min(d(i, C_v)) \tag{2.6}$$

O valor da silhueta $s(i)$ para o objeto i é dado por:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{2.7}$$

O cálculo do índice da silhueta da solução é a média das somas das silhuetas de todos os objetos. Uma solução S_1 é melhor do que uma solução S_2 se esse índice for maior para S_1 do que para S_2. Portanto, o algoritmo de clusterização deve maximizar este índice como segue a Equação 2.8.

$$MaximizarF(C) = \frac{1}{N} \sum_{i=1}^{N} s(i), \tag{2.8}$$

tal que N equivale ao total de objetos.

Um ponto negativo desta medida, apresentado por [23] é que uma solução que contém um único elemento dentro de um cluster e todos os outros objetos dentro de outro cluster, por exemplo, é classificado como uma boa solução por esta métrica. Enquanto, provavelmente, se o especialista deseja agrupar os dados, ele provavelmente não busca esta solução.

A tabela 2.3 foi apresentada em [8], indicando uma caracterização das estruturas

de clusters encontradas conforme o valor da silhueta.

Valores da Silhueta	Descrição
0,71 - 1,00	Uma estrutura forte foi encontrada.
0,51 - 0,70	Uma estrutura razoável foi encontrada.
0,26 - 0,50	A estrutura é fraca e pode ser superficial. É aconselhável o uso de outros métodos para esses dados.
$\leq 0,25$	Nenhuma estrutura substancial foi encontrada

2.4 Metaheurísticas Utilizadas

Vimos que o PCA é pertencente a classe NP-completo e que metaheurísticas se mostram apropriadas para este tipo de problema. Diante disso, este trabalho modela o PCA como um problema de otimização onde a função objetivo é dada pela equação 2.8. Para isso, foi utilizado uma abordagem composta pela combinação de duas metaheurísticas: *Iterated Local Search* e *Variable Neighborhood Descent* e cada uma delas será apresentada nesta seção.

2.4.1 *Iterated Local Search* (ILS)

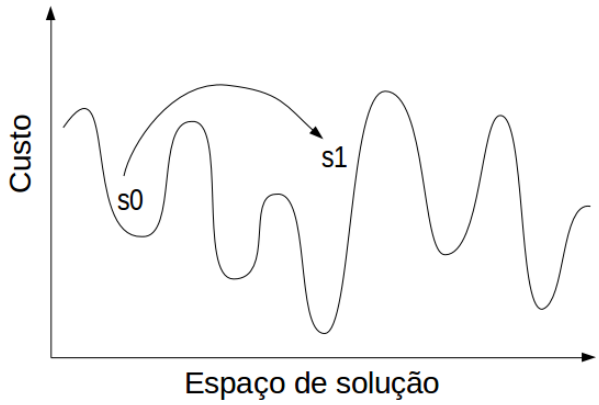
Iterated Local Search (ILS), ou em português Busca Local Iterada, é uma metaheurística com uma proposta de ser simples, tanto conceitualmente quanto na prática e sem perda de generalidade [15].

Uma fragilidade da Busca Local é ser sensível a solução de entrada. Para tratar esta fragilidade, a ideia do ILS é executar a Busca Local a partir de várias soluções diferentes. Assim, novas soluções de partida são geradas através de um método de perturbação na solução ótima local.

O método de perturbação deve ser suficientemente forte para permitir que a busca local explore diferentes soluções, mas também fraca para evitar ciclagem ao longo do processo de busca [15].

Basicamente, o ILS precisa de um procedimento para gerar uma solução de partida. Na sequência, o algoritmo procura um ótimo local, através de um método de busca local, próximo a essa solução. Em seguida, entra em um processo iterativo de refinamento, onde a solução ótima local é perturbada para se explorar outra região do espaço

Figura 2.1: ILS explorando o espaço de soluções.



de solução. Por fim é encontrado o ótimo local dessa nova região, e as soluções são comparadas. O algoritmo sempre escolhe a solução com melhor custo para ir para a próxima iteração. O pseudocódigo do ILS é apresentado em Algoritmo 1.

Algoritmo 1 ILS

```
1: função ILS
2:     s_0 ← SolucaoInicial;
3:     s_0 ← BuscaLocal(s_0));
4:     enquanto critério de parada não é atingido faça
5:         s_1 ← Perturbacao(s_0);
6:         s_1 ← BuscaLocal(s_1);
7:         s_0 ←CritérioDeAceitação(S_0, S_1);
8:     fim enquanto
9:     devolve s_0;
10: fim função
```

A figura 2.1, apresenta como o ILS explora o espaço de soluções. Começa com um solução s_0, encontra um ótimo local. Através de um método de perturbação, uma nova solução $s1$ é encontrada e novamente a busca local pode ser aplicada para encontrar um ótimo local.

2.4.2 *Variable Neighborhood Descent (VND)*

O método de descida em vizinhança variável (*Variable Neighborhood Descent*, VND) é uma variação do método de pesquisa em vizinhança variável (Variable Neighborhood Search, VNS) apresentado em [19]. Trata-se de um procedimento com várias vizinhanças, onde cada uma delas se submete a um procedimento de busca local diferente, permitindo que o algoritmo experimente várias trajetórias de melhoria distintas.

O algoritmo parte de uma solução inicial e a cada iteração ela é submetida a uma vizinhança. Quando a solução corrente for melhor que a de partida, temos um novo ótimo local, então, esta vira a solução de partida do algoritmo que volta para a primeira vizinhança com o novo ótimo local. Entretanto, se a solução corrente não for melhor, o algoritmo prossegue para a próxima vizinhança. Ele se encerra quando atingir a última vizinhança sem melhoria ou chegar a algum critério de parada como tempo máximo de processamento. O pseudocódigo do VND é apresentado em Algoritmo 2.

Algoritmo 2 VND

```
1: função VND(s)
2:     r = numero de vizinhanças;
3:     k ← 1;
4:     enquanto k <r faça
5:         Seja s' um ótimo local segundo o k-ésimo procedimento de refinamento
6:         se f(s') ≥ f(s) então
7:             s' ← s;
8:             k ← 1;
9:         senão
10:            k ← k + 1;
11:        fim se
12:    fim enquanto
13:    devolve s;
14: fim função
```

3 Implementação

Foi desenvolvido um algoritmo baseado em uma metaheurística híbrida para solucionar o problema de clusterização automática. O código está escrito na linguagem C e foi implementado na IDE Codeblocks 16.01.

O Algoritmo 3 apresenta o pseudocódigo dessa metaheurística formada pela combinação das metaheurísticas *Iterated Local Search* e *Variable Neighborhood Descent* apresentadas no capítulo anterior. Nesta combinação, o método de busca local do ILS é substituído pela metaheurística VND.

Algoritmo 3 Metaheurística híbrida ILS + VND

```
1: função ILS-VND
2:     s0 ← GeraSolucaoInicial;
3:     s0 ← VND(s0));
4:     enquanto critério de parada não é atingido faça
5:         s1 ← Perturbacao(s0);
6:         s1 ← VND(s1);
7:         s0 ← MelhorSolucao(s0, s1);
8:     fim enquanto
9:     devolve s0;
10: fim função
```

A implementação é dividida em 4 partes: Pré processamento, que carrega e prepara os dados antes de submeter a metaheurística, Construtivo (representado pela função "GeraSolucaoInicial"), Busca Local (representado pela função "VND") e por último a Perturbação. As próximas seções deste capítulo apresentam a implementação de cada uma destas partes.

3.1 Pré-Processamento

A primeira etapa do algoritmo é responsável pelo carregamento de uma ou mais bases de dados de arquivo(s) para a aplicação e a transformação desses dados. Para tal, é necessário que os arquivos apresentem uma disposição sistemática, em que a primeira linha deve informar o número de registros n seguido do número de atributos m da base,

separados por vírgula.

A segunda linha deve informar para o programa qual o tipo de dado de cada atributo separado por virgula. O programa é capaz de trabalhar com atributos numéricos inteiros ou reais e dados categóricos. Para informar o tipo de cada atributo em questão ao programa, é necessário que a linha seja preenchida com 0 quando o atributo for um atributo inteiro, 1 quando o atributo for real e 2 quando o atributo for categórico. Para todas as linhas em diante, os dados devem conter mais a esquerda o identificador do objeto e em seguida os seus atributos separados por vírgulas. É importante ressaltar que o identificador deve, necessariamente, ser tratado como um valor inteiro. Ele não vai interferir de maneira alguma na clusterização, trata-se apenas de um identificador.

Feito a leitura do arquivo, o algoritmo realiza um pré processamento que visa a normalização dos dados, com o propósito de minimizar problemas oriundos do uso de unidades e dispersões muito heterogêneas entre variáveis. Portanto, assim que os dados são lidos pelo algoritmo, é feito uma normalização segundo a amplitude com a equação 2.2, apresentada anteriormente, deixando todos os atributos com valores entre 0 e 1. A importância desta etapa foi mais detalhada no capítulo 2, que se dá pela medida de similaridade escolhida para o algoritmo: Distância Euclidiana.

A finalidade da clusterização é agrupar os dados sem nenhum conhecimento prévio sobre eles, assim, o resultado do agrupamento está intimamente relacionado com a seleção dos dados que foram submetidos para análise. A inserção de ruídos e *outliers* devem ser evitados o quanto for possível, para o resultado da clusterização não ser influenciado por estas informações. Além da seleção dos registros, a seleção dos atributos também pode influenciar negativamente no resultado [33].

Espera-se que este tratamento seja feito antes de submeter os dados a este algoritmo, pois o único tratamento em que os dados são submetidos é a normalização.

Uma vez que os dados já foram tratados e normalizados, a fase de pré-processamento está concluída e os dados estão prontos para serem submetidos à próxima fase, onde é gerada a solução de partida da metaheurística.

3.2 Algoritmo Construtivo

A estratégia adotada para resolver o problema da clusterização automática requer uma solução de partida, cuja escolha tem influência direta no resultado final do algoritmo. O construtivo é o método responsável por gerar esta solução inicial.

Na implementação deste método são geradas várias soluções viáveis e é selecionada a que apresentar o melhor valor da silhueta (mais próximo de 1). Para gerar esse conjunto de soluções, a base de dados é modelada em um grafo, onde cada vértice representa um objeto e cada aresta representa a dissimilaridade dos objetos que a compõe. Para calcular a dissimilaridade entre dois objetos foi utilizado o cálculo da distância euclidiana (Equação 2.1).

O construtivo inicia um processo iterativo onde cada iteração define um valor máximo de peso de aresta, chamado de limiar. A primeira etapa deste processo é compor o grafo com somente as arestas que possuírem peso menor do que o limiar, seguida da etapa de definição dos clusters e da avaliação da solução atual. Veremos a seguir cada uma das etapas.

Na primeira etapa é construído um grafo onde todas as arestas possuem peso menor do que o limiar. Em seguida, na etapa de definição dos clusters, é realizado uma busca em profundidade para encontrar as componentes conexas do grafo resultante desta remoção. Cada componente encontrada representa um cluster, gerando assim uma nova solução.

A última etapa consiste em avaliar esta solução gerada. Para isso é calculado o índice silhueta. Caso o índice da solução atual seja diferente do anterior, é armazenada a atual em uma lista de soluções viáveis. Ao final do processo, a melhor solução (maior silhueta) é retornada.

No Algoritmo 4, *objetos* é a lista de objetos oriundas da primeira etapa, *min* é a menor distância entre dois objetos, *max* é maior distância entre dois objetos e *incremento* define o passo da iteração. O *incremento* é justamente, o parâmetro do algoritmo que vai ajustar a sensibilidade do limiar L. Quanto maior a precisão deste incremento, maior o número de soluções geradas, em contrapartida também é maior o número de iterações realizadas nesta fase.

Algoritmo 4 Construtivo

```
1: função CONSTRUTIVO(objetos, min, max, incremento)
2:     limiar ← min;
3:     enquanto limiar ≤ max faça;
4:         G1 ← DefineGrafo(limiar, objetos);
5:         C ← AtribuiCluster(C, G1);
6:         silhueta ← CalculaSilhueta(C);
7:         se critério de aceitação da solução então
8:             S ← InsereSolucao(C, silhueta);
9:         fim se
10:        limiar ← (limiar + incremento);
11:    fim enquanto
12:    devolve MelhorSolucao (S);
13: fim função
```

Nos casos em que o incremento tem alta precisão, pode ser que a solução não mude de uma iteração para outra, pois a alteração foi tão pequena que não afetou na clisterização encontrada anteriormente. Para o algoritmo não considerar estas soluções, o critério de aceitação da solução verifica se a silhueta da iteração anterior é diferente da atual, além do refinamento de não considerar soluções com silhuetas negativas. Com isso, o algoritmo só vai considerar como promissoras as soluções distintas e de silhueta positiva.

Após terminar o processo iterativo, o construtivo gerou um *pool* de soluções e precisa decidir qual a melhor para o algoritmo prosseguir para as próximas etapas. Para tomar essa decisão, cada uma das soluções geradas é submetida a um método de acoplamento de clusters de tamanho 1, onde estes clusters são acoplados a outros sorteados aleatoriamente. Após este processo, a silhueta de cada solução é recalculada e a que possuir o maior valor é escolhida como a solução inicial.

Existem duas soluções triviais para a clusterização que devem ser evitadas pelo algoritmo: aquela em que todos os objetos estão em um único cluster e aquela em que cada cluster possui um único objeto. Esse método de acoplamento tem a finalidade de impedir que as soluções convirjam para uma dessas duas soluções triviais. Dessa forma ele evita as soluções em que cada cluster possui um único objeto, através de um laço que verifica o tamanho de todos os clusters e aqueles que apresentarem tamanho igual a 1 serão acoplados a outro sorteado aleatoriamente. Contudo, uma restrição é utilizada neste método: só terá acoplamento se o número de clusters for maior ou igual a 2. Essa restrição foi adicionada para o algoritmo não convergir para 1 único cluster que é a outra

solução trivial que gostaríamos de evitar.

Além de ajudar a identificar bons agrupamentos, o cálculo da silhueta fornece uma medida do quão bem posicionado um objeto está dentro de um cluster. Isto permite que durante esta fase, os objetos mal posicionados, isto é, com $s(i) \leq 0$ sejam identificados. Portanto, as soluções que saem da fase de construção, saem com os objetos mal classificados identificados. Esta informação será utilizada na busca local que é direcionada a realizar trocas desses objetos mal posicionados, na procura de melhorar a clusterização.

Na seção a seguir, é apresentado como estas trocas são feitas. Outra informação que é fornecida no cálculo da silhueta, exatamente na hora de calcular a Equação 2.6, é o cluster mais próximo de cada objeto, com exceção do cluster em que o objeto está incluído. Esta informação também será utilizada adiante, na fase de perturbação.

3.3 Busca Local Direcionada

O terceiro passo do algoritmo é tentar melhorar a solução inicial, gerada na fase de construção. Como dito anteriormente, nesta fase foi calculado o valor da silhueta de cada objeto, e aqueles cuja silhueta possuíam valor negativo, foram marcados como insatisfeitos. Além disso, o cluster em que eles se identificaram mais do que o cluster atual foi guardado junto as suas informações. A busca local é responsável por explorar esta informação e realizar a tentativa de melhorar a solução.

A busca local aqui é denominada como direcionada, pois ela é guiada pela insatisfação dos objetos dentro do cluster atual, trocando-os para o cluster desejado. Quando um objeto está mais próximo de um outro cluster que não seja o qual ele faz parte, então $s(i) < 0$ na equação 2.7. Os objetos que apresentam esse valor negativo são classificados como insatisfeitos. É importante perceber que trocar um objeto de cluster, pode gerar uma insatisfação em outros objetos, e acabar piorando a solução. Para amortecer este problema, o algoritmo utiliza a metaheurística VND, apresentada anteriormente, onde há várias tentativas de melhoria e é escolhida a que apresentar a melhor solução.

Esta metaheurística necessita de uma função de busca local diferente para cada vizinhança, este trabalho propõe a implementação de uma função de busca local que recebe como parâmetro uma porcentagem de objetos insatisfeitos que se deseja trocar. Cada

vizinhança explora esta função com uma porcentagem de troca distinta. A busca local poderá trocar um valor menor ou igual a porcentagem recebida por parâmetro. O pseudocódigo da busca local em uma determinada vizinhança é apresentada no Algoritmo 5.

Algoritmo 5 Busca Local

```
1: função BuscaLocal(solucao, porcentagemTroca)
2:     novaSilhueta ← ∞;
3:     enquanto (solucao.silhueta ≤ novaSilhueta E iteracoes ≤ maxIteacoes) faça
4:         Cluster C ← solucao.clusters;
5:         para cada cluster c faça
6:             numTrocas = CalculaTrocas(Porcentagem, C);
7:             Roleta(C, numTrocas);
8:         fim para
9:         novaSilhueta ← CalculaSilhueta(C);
10:        iteracoes + +;
11:    fim enquanto
12:    se (novaSilhueta ≥ solucao.silhueta) então
13:        solucao.cluster ← C;
14:        solucao.silhueta ← novaSilhueta;
15:    fim se
16:    devolve S
17: fim função
```

Se a busca local apresentar uma melhoria na solução, o algoritmo volta para a busca local da primeira vizinhança agora com a nova solução. Entretanto se não houver melhoria, o algoritmo vai para a próxima vizinhança tentar melhorar a mesma solução. Quando não houver mais objetos insatisfeitos, ou não houver mais vizinhanças para explorar, a busca local retorna a solução obtida.

No caso de o construtivo gerar uma solução com 0 objetos insatisfeitos, a busca local retorna a mesma solução. Pois isso indica que os objetos estão todos bem classificados nesta configuração.

Para trocar os objetos de cluster, o algoritmo se baseia no jogo da roleta, onde cada objeto deve ganhar algum setor da roleta para apostar, e em seguida, um número é sorteado aleatoriamente e o objeto que possui o setor na qual o número sorteado pertença será escolhido para ser deslocado para o cluster de mais afinidade, segundo a silhueta.

Este processo é repetido pelo número de vezes que corresponde a porcentagem de troca passada por parâmetro e é chamado de método da roleta, seu funcionamento é apresentado a seguir.

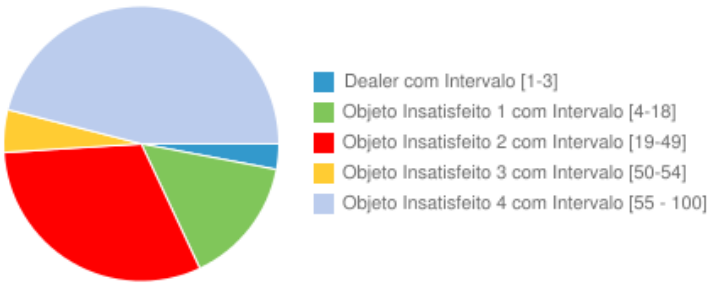
3.3.1 Método da Roleta

Com a porcentagem de troca passada por parâmetro e a quantidade de objetos insatisfeitos salvos na solução, o algoritmo calcula a quantidade de objetos que serão trocados para o cluster desejado. Uma vez definido quantos objetos serão trocados, este método vai determinar quais dos objetos insatisfeitos da solução serão trocados.

O que este método faz é se basear no jogo da roleta, onde jogadores recebem números para apostar e um número é sorteado aleatoriamente. Vence quem apostar no número sorteado.

Trata-se de um processo iterativo com N iterações, onde N é a quantidade de objetos insatisfeitos que serão trocados de cluster e em cada iteração, a roleta terá 100 números que serão divididos cada objeto insatisfeito e um dealer. Quando um número pertencente ao dealer é sorteado, não há troca de cluster para nenhum objeto. Uma vez que um objeto é sorteado ele sai da roleta, e começa a próxima iteração. Caso um número do dealer seja sorteado, nenhuma troca é realizada nesta iteração e o algoritmo segue para a próxima iteração.

Figura 3.1: Roleta dividida para 4 objetos insatisfeitos e um dealer.



Para aumentar a probabilidade dos objetos mais insatisfeitos serem sorteados, a distribuição dos números é dada proporcionalmente a insatisfação de cada objeto, isto é, objetos mais insatisfeitos recebem mais números que objetos menos insatisfeitos. Uma vez que o setor do dealer é definido, o cálculo da insatisfação proporcional de cada objeto i é dado pela equação 3.1. Quando a equação 3.1 for aplicada para todos os objetos, o dealer e cada objeto ganha um setor da roleta referente a quantidade de números que cada objeto vai apostar.

$$InsatisfacaoProp(i,d) = \frac{100-d}{100} \times Insatisfacao(i) \tag{3.1}$$

A Tabela 3.1 retrata uma situação onde uma solução apresenta 4 objetos que vão jogar na roleta, seguido do valor de insatisfação calculado durante o método da silhueta, a insatisfação proporcional representada em uma roleta em que o dealer não possui nenhum número e a insatisfação proporcional em que o dealer possui 3 números. A Figura 3.1 apresenta a roleta do problema representado nesta tabela.

Tabela 3.1: Insatisfação de cada objeto

Objeto	Insatisfação real	InsatisfacaoProp(i,0)	InsatisfacaoProp(i,3)
Objeto Insatisfeito 1	$0,03$	$15,8$	15,55
Objeto Insatisfeito 2	$0,06$	$31,6$	30,65
Objeto Insatisfeito 3	$0,01$	$5,2$	5,04
Objeto Insatisfeito 4	$0,09$	$47,4$	45,98
Total:	$0,19$	100	97

3.4 Perturbação

Como visto anteriormente, a etapa de perturbação faz com que o algoritmo saia de um ótimo local para explorar outra zona do espaço de solução. Para isso ela deve ser suficientemente forte para permitir que a busca local explore diferentes soluções, mas também fraca para evitar ciclagem ao longo do processo de busca.

Para esta etapa temos um laço onde uma solução é perturbada, e em seguida a busca local vem ajustando essa solução para um ótimo local. A função de busca local utilizada foi a mesma aplicada após o construtivo (Seção 3.3). Portanto, nesta seção é apresentado somente o funcionamento do método de perturbação.

O método recebe por parâmetro uma porcentagem de objetos para serem trocados de cluster. Esse parâmetro que vai determinar o quão forte ou o quão fraca será a perturbação. O algoritmo seleciona aleatoriamente os objetos, segundo essa porcentagem, para serem trocados de cluster. Nesta hora, não há critério na escolha do objeto, como é feita na busca local. Contudo, a escolha para o cluster que este objeto será enviado é o mais próximo, que não o cluster ao qual ele já pertence, segundo a Equação 2.6.

Após esta troca, a solução é novamente avaliada pela silhueta, submetida para a busca local, e, por fim, é verificado se houve alguma melhoria. O algoritmo sempre fica com a solução de melhor silhueta. Quando esse laço de perturbação seguido da busca local terminar, temos a solução final do algoritmo.

4 Experimentos Computacionais

Para fins de avaliar os algoritmos desenvolvidos, foram realizados testes computacionais em um conjunto de 82 instâncias obtidas a partir do trabalho apresentado em [26]. Assim como indicado no trabalho de referência, os resultados aqui apresentados são divididos em três grupos de bases de dados (*datasets*): DS1, DS2 e DS3.

O grupo DS1 é composto por nove instâncias conhecidas na literatura. Como pode ser visto na Tabela 4.1, essas instâncias são caracterizadas por um número de atributos que varia de 2 a 13 e número de objetos variando de 75 a 1484 registros. Este grupo de instâncias integra bases apresentadas em diferentes problemas, como em [9], [24], [16], [10], [20] e [32].

A Tabela 4.1 apresenta cada uma dessas instâncias junto ao seu número de atributos (m) ou dimensões e número de registros (n) ou quantidade de objetos.

Tabela 4.1: Instâncias do grupo DS1 [26]

Instância	m	n	Instância	m	n
200DATA	2	200	ruspini	2	75
gauss9	2	900	vowel2	2	528
iris	4	150	wine	13	178
maronna	2	200	yeast	7	1484
spherical_4d3c	3	400			

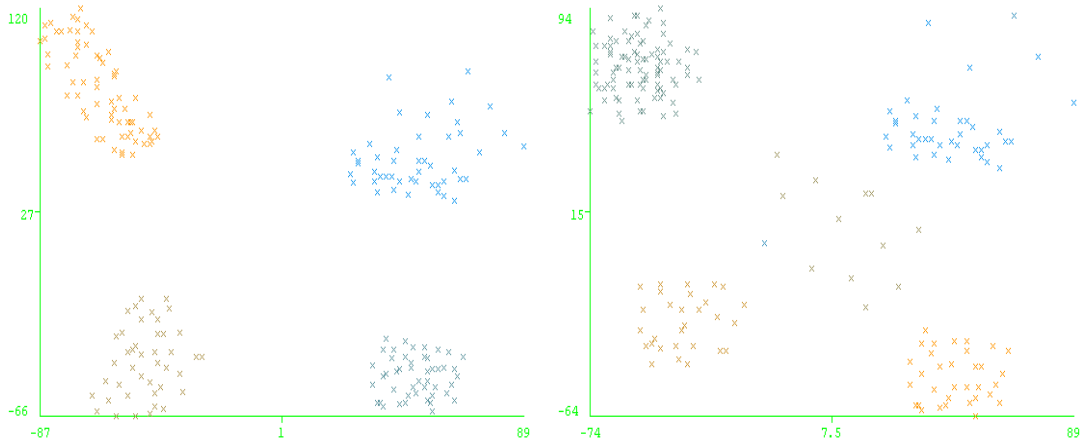
Já o segundo conjunto de testes, denominado DS2, é composto por 51 instâncias, que foram construídas por [3] utilizando a ferramenta **Dots** desenvolvida por [28], onde a dimensão (m) e a quantidade de objetos (n) são apresentadas na Tabela 4.2.

Tabela 4.2: Instância do grupo DS2 [26]

Instância	m	n	Instância	m	n	Instância	m	n	Instância	m	n
100p10c	2	100	500p19c1	2	500	100p2c1	2	100	500p6c1	2	500
100p3c	2	100	500p4c1	2	500	100p3c1	2	100	500p6c1	2	500
100p7c	2	100	600p15c	2	600	100p8c1	2	106	600p3c1	2	600
100p5c1	2	110	700p4c	2	700	100p7c1	2	112	700p15c1	2	703
200p2c1	2	200	800p10c1	2	800	200p3c1	2	200	800p18c1	2	800
200p4c	2	200	800p4c1	2	800	200p4c1	2	200	800p23c	2	806
200p7c1	2	210	900p5c	2	900	200p8c1	2	212	1000p14c	2	900
200p12c1	2	222	1000p5c1	2	1000	300p13c1	2	235	1000p6c	2	1000
300p10c1	2	300	1000p7c1	2	1000	300p2c1	2	300	1300p27c1	2	1005
300p3c	2	300	1100p6c1	2	1100	300p3c1	2	300	1300p17c	2	1300
300p4c1	2	300	1500p6c1	2	1500	300p6c1	2	300	1800p23c	2	1800
400p3c	2	400	1900p24c	2	1901	400p4c1	2	400	2000p11c	2	2000
400p17c1	2	408	2000p26c	2	2000	2000p9c1	2	2000			

Neste conjunto, o nome de cada instância foi definido de acordo com a quantidade de objetos, quantidade de clusters, e se os grupos são bem definidos, coesos e linearmente independentes (denominados, segundo o autor como “comportadas” e “não comportadas”). Na Tabela 4.2, as instâncias terminadas em “c” são consideradas comportadas, enquanto as instâncias terminadas em 1 são denominadas não comportadas, como mostram as Figuras 4.1(a) e 4.1(b), onde $200p4c$ é uma instância comportada e a $200p4c1$ é dita não comportada.

Por último, o conjunto de testes denominado DS3, apresentado na Tabela 4.3, é formado por 22 instâncias que foram construídas e utilizadas por [28] e [27]. A Tabela 4.3 apresenta este conjunto de testes no espaço R^2 e a cardinalidade varia de 30 a 2000 objetos.



(a) 200p4c é uma instância comportada (b) 200p4c1 é uma instância não comportada

Figura 4.1: Instância comportada (a) e não comportada (b)

Tabela 4.3: Instâncias do grupo DS3 [26]

Instâncias	m	n	Instâncias	m	n
30p	2	30	*outliers_ags*	2	80
97p	2	97	3dens	2	128
Outliers	2	150	157p	2	157
convdensity	2	175	181p	2	181
convexo	2	199	2face	2	200
Face	2	296	300p4c	2	300
350p4c	2	350	numbers	2	437
450p4c	2	450	moreshapes	2	489
500p3c	2	500	numbers2	2	540
600p3c	2	600	900p5c	2	900
1000p6c	2	1000	2000p11c	2	2000

Os experimentos computacionais foram realizados em um computador com a seguinte configuração: processador Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz, 8GB de RAM DDR3, Sistema Operacional Linux Mint 17.3 de 64 bits com kernel 3.19.

Para gerar os resultados, foram definidos alguns parâmetros para cada um dos três principais métodos do algoritmo. Primeiramente, no construtivo, existe um loop onde o limiar é incrementado em 0.01 a cada iteração em busca de gerar novas soluções. Em

seguida, no método de busca local foram escolhidas 6 vizinhanças, onde as porcentagens de troca são 10% para a primeira vizinhança, 30% para a segunda, 50% para a terceira, 70% para a quarta, 90% para a quinta e 100% para a sexta. Além das vizinhanças, a busca local necessita de um critério de parada após várias tentativas sucessivas sem melhoria em uma mesma vizinhança. O critério utilizado foi um limite de 10 iterações. No método da roleta, foi considera sempre um setor de 3% dos números para o dealer. Finalmente, para o método de perturbação, a porcentagem de objetos a serem trocados de cluster foi definida como 1% do número n total de objetos. Esses valores foram utilizados após vários testes alterando-se os parâmetros.

Para cada instância, o algoritmo foi executado 30 vezes e em cada uma delas utilizando uma semente de randomização diferente. Ao final de cada execução anotou-se os valores de silhueta, números de cluster gerados e o tempo de execução da solução inicial e da solução final encontrada pelo algoritmo. Com estas informações é possível analisar a solução de partida e a contribuição do restante do algoritmo no resultado final.

As Tabelas B.1, B.2, B.3 e B.4, apresentam os melhores resultados encontrados para cada uma das instâncias dentre as 30 execuções. As Tabelas B.2 e B.3 dividiram o conjunto DS2 em dois subconjuntos somente para facilitar a visualização dos dados, devido ao grande número de instâncias. Observando estas tabelas, podemos perceber que o algoritmo encontrou soluções com silhueta positiva para todas as instâncias e para aquelas dita como “comportadas”, cujo o valor do cluster era esperado, o algoritmo acertou o número de clusters em 100% dos casos.

4.1 Contribuição dos Algoritmos no Índice Silhueta

Para analisar o comportamento do algoritmo, esta seção faz uma comparação da contribuição do método construtivo e do restante do algoritmo no valor do índice silhueta.

As Tabelas A.1, A.2, A.3 e A.4 apresentam uma comparação da solução inicial (S_0) do algoritmo com a solução final (S). Esse estudo permite observar a contribuição da solução de partida no valor da silhueta bem como o restante do algoritmo característico do ILS (Busca local + Perturbação).

Através da tabela A.1 pode-se observar que tanto o construtivo, quanto o restante

do algoritmo são indispensáveis, uma vez que para certas instâncias o construtivo colabora com 100% no valor da silhueta, enquanto para outras instâncias a perturbação e a busca local chegam a aumentar em mais de 100% no valor da silhueta, um exemplo é a instância gauss9.

As soluções finais das instâncias $200DATA$, *iris*, *maronna*, *ruspini*, *yeast* e *spherical_4d3c* são as soluções encontradas no Construtivo. Isso significa que o Construtivo contribuiu com 100% no valor da silhueta para estas instâncias. Através das instâncias *gauss*9, *vowel* e *wine* o pós-construtivo se mostra importante na melhoria da solução, já que para gauss ele é capaz de melhorar a silhueta em 159%, para *wine* em 43% e para *vowel* em 17%.

Para analisar a contribuição no valor total da silhueta, representada na Tabela A.1 pode-se observar a Figuras 4.2 onde são representadas todas as contribuições percentuais de todas as soluções onde ambas partes dos algoritmo contribuíram no valor total da silhueta.

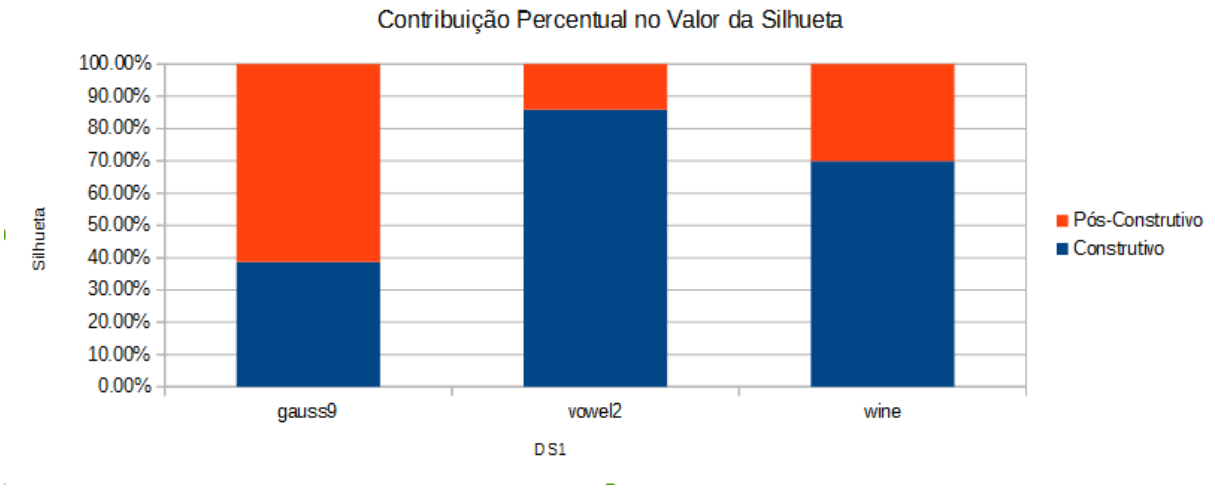


Figura 4.2: Comparação percentual da contribuição do algoritmo construtivo e do restante do algoritmo no valor da silhueta para instâncias do conjunto DS1

No conjunto DS2, as instâncias comportadas são identificadas, e pode-se observar, através das Tabelas A.2 e A.3, que para 100% das instância comportadas a melhor solução foi encontrada já pelo construtivo. E, para as instâncias não comportadas, o restante do algoritmo se mostrou importante contribuindo com algum ganho no valor da silhueta em mais de 50% das instâncias, chegando a contribuir com 169% no índice da silhueta da

instância 800p10c1.

Para poder visualizar a contribuição do método construtivo e do restante do algoritmo no valor da silhueta, apresentados nas Tabelas A.2 e A.3, gerou-se o gráfico representado pela Figura 4.3. Esta representa contribuição percentual do Construtivo e do restante do algoritmo, nos casos em que houveram contribuição das duas partes.

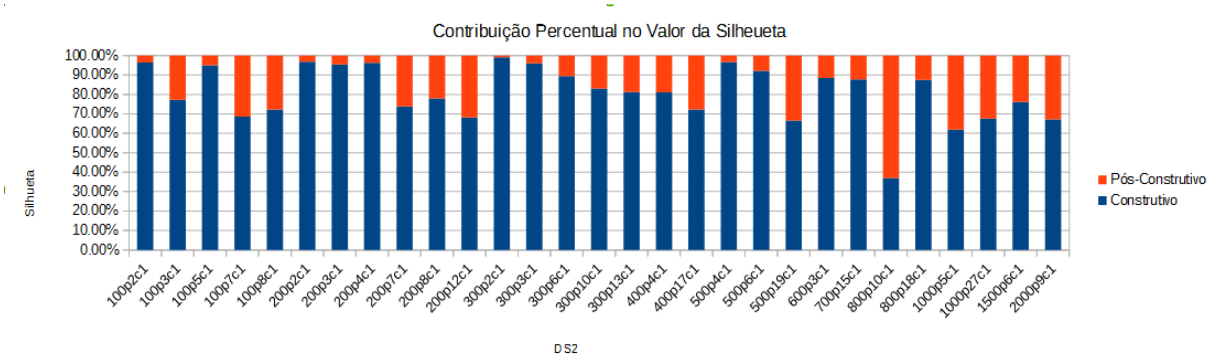


Figura 4.3: Comparação da contribuição do algoritmo construtivo e do restante do algoritmo no valor da silhueta para instâncias do conjunto DS2

Por fim, a Tabela A.4 apresenta a contribuição do construtivo e do restante do algoritmo no valor da silhueta para o conjunto DS3. Esses dados são representados graficamente na Figura 4.4.

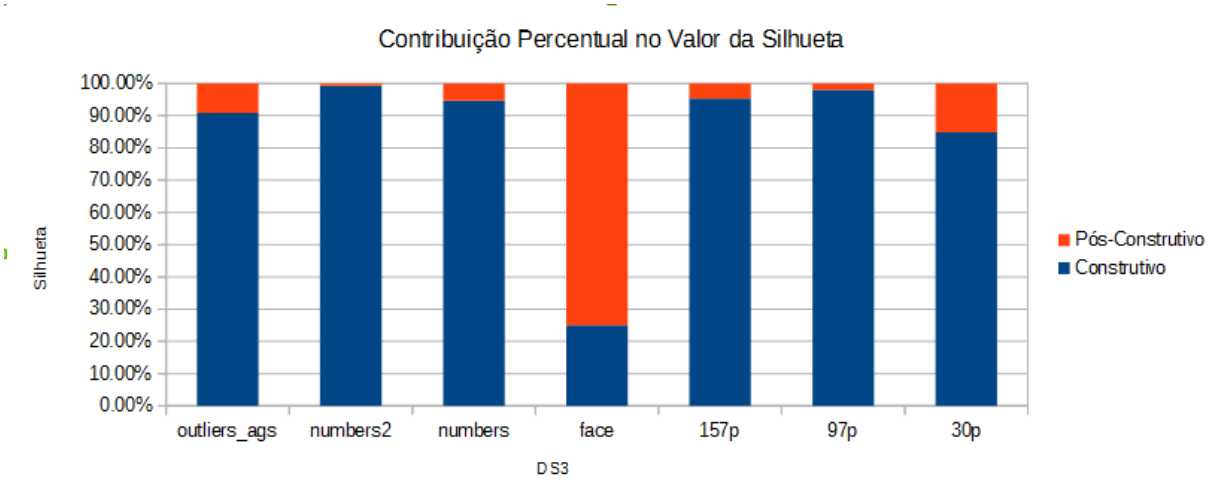


Figura 4.4: Contribuição Percentual do Construtivo para instâncias do conjunto DS3

Para 68% das instâncias de DS3 o algoritmo encontra a melhor solução logo pelo construtivo. Enquanto o restante do algoritmo faz alguma melhoria do restante das instâncias, chegando a melhorar mais de 300% no índice da silhueta para a instância *face*.

Nesta seção foi possível observar que, tanto o Construtivo quanto o restante do algoritmo (Perturbação e Busca Local), apresentaram contribuição significativa no valor do índice da silhueta. O construtivo foi capaz de gerar a solução final para mais de 50% das instâncias. Todavia, para algumas instâncias, apenas com a contribuição da fase de perturbação e da busca local (características do ILS), foi possível melhorar a qualidade da solução obtida na construção, mesmo esta sendo caracterizada por um processo iterativo adaptativo. Assim, graças às características do ILS, houve um aumento de até 300% na qualidade da solução.

4.2 Comparação e Análise dos resultados

Em [26] foram desenvolvidos alguns métodos heurísticos para o problema da clusterização automática. Nesta seção é apresentada uma comparação dos resultados obtidos pelo algoritmo híbrido proposto, denominado ILS-VND, com o algoritmo AECBL1, proposto por [4], e os algoritmos ILS-FJGP, ILS-DBSCAN e Heurística Híbrida (HH) para o problema da clusterização automática desenvolvidos e apresentados por [26].

A Tabela 4.4 apresenta a comparação da silhueta de cada instância do conjunto DS1 para cada um desses algoritmos. As colunas AECBL1-K e ILS-VND-k indicam o número de clusters obtidos pelos respectivos algoritmos. Como em [26], o autor não apresenta o número de clusters obtidos, esta informação somente é apresentada para dois algoritmos. Os valores destacados em negrito indicam os melhores resultados.

Tabela 4.4: Comparação dos resultados dos algoritmos para as instâncias do grupo DS1

Instâncias	AECBL1-K	AECBL1	ILS-FJGP	ILS-DBSCAN	HH	ILS-VND-k	ILS -VND
200DATA	3	**0.823**	0.455	0.762	**0.823**	3	0.822
gauss9	9	**0.482**	0.289	0.106	**0.482**	3	0.383
iris	2	**0.687**	0.643	0.102	**0.687**	2	0.630
maronna	4	**0.575**	0.197	0.218	**0.575**	2	0.539
ruspini	4	0.738	0.522	**0.746**	0.738	4	0.740
spherical_4d3c	4	**0.689**	0.223	0.132	**0.689**	4	**0.689**
vowel2	74	**0.451**	0.242	0.417	**0.451**	2	0.405
wine	2	**0.660**	0.539	0.641	**0.660**	2	0.385
yeast	2	0.628	-0.055	0.380	0.628	2	**0.657**

Para o conjunto de instâncias DS1, conforme a Tabela 4.4, o ILS-VND apresentou um resultado com índice silhueta maior para a instância *yest*.

O ILS-VND encontrou soluções com índice de silhueta maior ou igual do que os algoritmos da literatura para 22% das instâncias. E apresentou resultado melhor do que pelo menos um algoritmo para 88% das instâncias de DS1.

Em 55% das instâncias o algoritmo proposto encontrou soluções melhores do que o ILS-FJGP. E para 66% das instâncias de DS1, ele apresentou resultados melhores do que o ILS-DBSCAN.

Tabela 4.5: Comparação dos resultados dos algoritmos para as instâncias do grupo DS2 - parte 1

Instâncias	AECBL1-K	AECBL1	ILS-FJGP	ILS-DBSCAN	Heurística Híbrida	ILS-VND - k	ILS -VND
100p2c1	2	**0.743**	0.551	**0.743**	**0.743**	2	0.673
100p3c	3	0.786	0.502	**0.792**	0.786	3	0.784
100p3c1	3	0.579	0.391	0.559	0.579	3	**0.589**
100p5c1	8	**0.700**	0.655	0.654	**0.700**	9	0.692
100p7c	7	0.834	**0.838**	**0.838**	0.834	7	0.804
100p7c1	7	0.491	0.423	0.470	0.491	12	**0.509**
100p8c1	7	0.528	0.479	0.442	0.528	12	**0.529**
100p10c	10	0.834	**0.838**	**0.838**	0.834	10	0.835
200p12c1	13	0.577	0.344	0.441	0.577	13	**0.591**
200p2c1	2	0.764	0.561	0.758	0.764	2	**0.765**
200p3c1	3	0.681	0.440	0.672	0.681	3	**0.684**
200p4c	4	0.773	0.441	**0.776**	0.773	4	0.773
200p4c1	4	**0.745**	0.499	0.737	**0.745**	6	0.705
200p7c1	14	0.579	0.381	0.487	0.579	11	**0.585**
200p8c1	9	0.576	0.365	0.505	0.576	13	**0.577**
300p10c1	10	**0.609**	0.337	0.589	**0.609**	18	0.572
300p13c1	13	0.566	0.331	0.456	0.566	14	**0.592**
300p2c1	10	0.776	0.621	0.777	0.776	2	**0.778**
300p3c	3	0.766	0.431	**0.772**	0.766	3	0.767
300p3c1	3	0.676	0.435	**0.679**	0.676	3	**0.679**
300p4c1	2	0.607	0.478	**0.614**	0.607	2	0.569
300p6c1	8	**0.662**	0.470	0.633	**0.662**	11	0.631
400p17c1	2	**0.513**	0.278	0.427	**0.513**	30	0.464
400p3c	3	0.799	0.486	0.808	0.799	3	**0.818**
400p4c1	4	**0.602**	0.264	0.427	**0.602**	2	0.465
500p19c1	20	**0.483**	0.265	0.382	**0.483**	43	0.460
500p3c	3	0.825	0.604	0.830	0.825	3	**0.842**

Conforme as Tabelas 4.5 e 4.6, o algoritmo proposto (ILS-VND) apresentou agrupamentos melhores, ou pelo menos de mesmo custo, de 33% das instâncias.

Apresentou o pior resultado para 2% das instâncias.

Para 96% das instâncias apresentou resultados melhores do que o ILS-FJGP e em 48%, melhor do que o ILS-DBSCAN.

Tabela 4.6: Comparação dos resultados dos algoritmos para as instâncias do grupo DS1 -parte 2

Instâncias	AECBL1-K	AECBL1	ILS-FJGP	ILS-DBSCAN	HH	ILS-VND - k	ILS-VND
500p4c1	5	**0.661**	0.383	**0.661**	**0.661**	4	0.627
500p6c1	6	**0.630**	0.210	0.610	**0.630**	8	0.583
600p15c	15	0.781	0.262	**0.782**	0.781	15	**0.782**
600p3c1	3	**0.721**	0.393	0.710	**0.721**	14	0.548
700p15c1	15	**0.680**	0.322	0.574	**0.680**	26	0.656
700p4c	4	0.797	0.332	**0.804**	0.797	4	0.789
800p10c1	2	**0.468**	0.275	0.409	**0.468**	84	0.391
800p18c1	19	**0.692**	0.208	0.598	**0.692**	29	0.657
800p23c	23	0.787	0.301	**0.794**	0.787	23	0.788
800p4c1	4	**0.702**	0.279	0.652	**0.702**	18	0.446
900p12c	12	0.841	0.289	**0.842**	0.841	12	**0.842**
900p5c	5	0.716	0.302	**0.722**	0.716	5	0.698
1000p14c	14	0.831	0.231	**0.832**	0.831	14	0.828
1000p27c1	28	**0.523**	0.225	0.326	**0.523**	82	0.432
1000p5c1	5	**0.639**	0.202	0.628	**0.639**	6	0.599
1000p6c	6	0.736	0.142	0.742	0.736	6	**0.751**
1100p6c1	6	**0.671**	0.168	0.649	**0.671**	26	0.445
1300p17c	17	0.823	0.142	**0.824**	0.823	17	0.822
1500p6c1	6	**0.644**	0.050	0.618	**0.644**	32	0.377
1800p22c	22	0.798	0.145	**0.810**	0.802	22	0.801
1900p24c	24	**0.799**	0.288	**0.799**	**0.799**	24	0.797
2000p11c	11	0.713	-0.086	0.716	0.713	11	**0.724**
2000p26c	26	0.779	0.268	**0.808**	0.799	26	0.800
2000p9c1	9	**0.624**	-0.026	0.169	**0.624**	60	0.378

Por fim, a Tabela 4.7, apresenta a comparação da silhueta de cada instância do conjunto DS3 para cada um dos algoritmos. O ILS-VND encontrou um índice de silhueta melhor ou igual do que os outros algoritmos em 45% das instâncias.

Para 90% das instâncias, o algoritmo proposto encontrou melhores resultados que o ILS-FJGP. Para 40% das instâncias, o algoritmo proposto encontrou melhores resultados

que o ILS-DBSCAN.

Para 14% das instâncias o algoritmo obteve o pior resultado de todos os algoritmos comparados.

Tabela 4.7: Comparação dos resultados dos algoritmos para as instâncias do grupo DS3

Instâncias	AECBL1-K	AECBL1	ILS-FJGP	ILS-DBSCAN	HH	ILS-VND - k	ILS -VND
1000p6c	6	0.736	0.206	0.742	0.736	6	**0.751**
157p	4	**0.666**	0.338	0.664	**0.666**	4	0.662
181p	6	0.737	0.599	**0.744**	0.737	6	0.737
2000p11c	11	0.713	0.202	0.716	0.713	11	**0.724**
2face	2	0.667	0.548	**0.674**	0.667	2	0.464
300p4c	4	0.750	0.251	0.750	0.750	4	**0.764**
30p	9	0.724	0.564	0.791	0.787	11	**0.800**
350p5c	5	0.759	0.250	**0.768**	0.759	5	0.730
3dens	2	0.762	0.764	**0.764**	0.762	2	0.698
450p4c	4	**0.766**	0.427	0.772	**0.766**	4	**0.766**
500p3c	3	0.825	0.604	0.830	0.825	3	**0.842**
600p3c	3	0.751	0.331	**0.752**	0.751	3	**0.752**
900p5c	5	0.716	0.302	**0.722**	0.716	5	0.699
97p	3	0.711	0.419	0.711	0.711	3	**0.715**
convdensity	3	0.854	0.577	**0.858**	0.854	3	0.845
convexo	3	0.668	0.618	**0.676**	0.668	3	0.656
face	16	0.527	0.203	0.306	0.526	18	**0.528**
moreshapes	6	0.732	0.434	**0.734**	0.732	6	0.729
numbers	10	**0.581**	0.258	0.561	**0.581**	9	0.504
numbers2	10	**0.600**	0.267	**0.600**	**0.600**	11	0.548
outliers	2	0.785	**0.786**	**0.786**	0.785	2	0.769
outliers_ags	7	0.748	0.742	0.742	0.748	2	**0.805**

Não foi feita nenhuma análise do comportamento assintótico do algoritmo proposto, ainda que possa-se afirmar que o mesmo tem comportamento polinomial no tamanho da instância e na dimensão das mesmas. Para mostrar que o desempenho do algoritmo tem aplicação viável, as Tabelas 4.8, 4.9 e 4.10 apresentam os tempos de processamento (em segundos) obtido por [26] em um computador com a configuração: processador i7

de 3.0 GHz; com 4GB de RAM; Sistema Operacional Linux distribuição Ubuntu 9.10 e kernel 2.6.18. Enquanto os resultados do ILS-VND foram obtidos por outro computador com a configuração inferior, conforme apresentado no início deste capítulo.

Tabela 4.8: Comparação de tempos DS1

Instâncias	AECBL1	HH	ILS-VND	Instâncias	AECBL1	HH	ILS-VND
$200DATA$	54.19	54.21	**0.17**	$gauss9$	228.87	228.89	**55.97**
$iris$	17.90	17.92	**0.13**	$maronna$	178.69	178.70	**0.11**
$ruspini$	22.20	0.17	**0.02**	$spherical\_4d3c$	43.18	**0.69**	0.81
$vowel2$	1821.61	1821.63	**18.44**	$wine$	35.76	35.77	**4.97**
$yeast$	372.59	372.61	**158.11**				

Para as instâncias pertencentes ao grupo DS1, os resultados obtidos pelo algoritmo proposto foram gerados em menos tempo que os algoritmos da literatura para 88% das instâncias. Isto significa que apenas uma instância demorou mais tempo para encontrar o resultado.

Tabela 4.9: Comparação de tempos DS2

Instâncias	AECBL1	HH	ILS-VND	Instâncias	AECBL1	HH	ILS-VND
100p10c	17.71	0.05	**0.03**	100p2c1	17.93	17.94	**0.05**
100p3c	17.79	**0.03**	**0.03**	100p3c1	58.65	58.67	**0.44**
100p5c1	113.17	113.19	**0.22**	100p7c	13.30	**0.03**	**0.03**
100p7c1	107.64	107.65	**0.29**	100p8c1	39.62	39.64	**0.50**
200p12c1	145.30	145.31	**1.00**	200p2c1	40.30	40.32	**0.12**
200p3c1	69.72	93.59	**0.25**	200p4c	36.99	**0.03**	0.13
200p4c1	53.63	6.48	**0.28**	200p7c1	95.49	95.50	**0.90**
200p8c1	461.31	461.33	**0.89**	300p10c1	1252.57	722.72	**2.34**
300p13c1	209.06	209.08	**1.42**	300p2c1	104.38	189.56	**3.27**
300p3c	43.10	**0.04**	0.38	300p3c1	221.70	3.96	**0.47**
300p4c1	38.91	38.93	**4.15**	300p6c1	347.78	19.62	**3.87**
400p17c1	95.18	95.20	**7.02**	400p3c	51.80	**0.05**	0.87
400p4c1	85.94	85.96	**0.94**	500p19c1	332.52	332.54	**8.10**
500p3c	60.53	**0.06**	2.04	500p4c1	575.66	575.67	**9.89**
500p6c1	290.66	290.67	**8.79**	600p15c	291.67	**0.19**	1.82
600p3c1	115.72	115.74	**25.93**	700p15c1	901.18	901.19	**33.78**
700p4c	94.64	**0.09**	3.50	800p10c1	136.96	136.97	**37.05**
800p18c1	1490.96	1490.98	**25.31**	800p23c	224.64	**0.45**	4.49
800p4c1	136.26	136.28	**65.61**	900p12c	428.56	13.21	**6.69**
900p5c	125.45	**0.14**	6.65	1000p14c	247.62	25.66	**8.93**
1000p27c1	1398.75	1398.77	**128.82**	1000p5c1	232.16	232.18	**64.32**
1000p6c	188.67	**0.16**	9.32	1100p6c1	467.11	467.13	**106.17**
1300p17c	234.17	**2.39**	15.71	1500p6c1	320.22	320.24	**136.19**
1800p22c	1658.86	**0.52**	45.60	1900p24c	281.13	**0.78**	50.61
2000p11c	1342.98	**0.55**	66.51	2000p26c	2507.53	**0.64**	57.10
2000p9c1	393.65	393.66	**322.23**				

Para as instâncias pertencentes ao grupo DS2, segundo a Tabela 4.9, o algoritmo

prosposto chegou ao resultado utilizando menos tempo do que os algoritmos da literatura para 72% das instâncias.

Tabela 4.10: Comparação de tempos DS3

Instâncias	AECBL1	HH	ILS-VND	Instâncias	AECBL1	HH	ILS-VND
1000p6c	414.03	**0.17**	9.20	157p	52.81	52.83	**0.12**
181p	177.86	**0.07**	0.10	2000p11c	1826.71	**0.55**	66.68
2face	58.51	**0.03**	0.13	300p4c	105.77	**0.04**	0.37
30p	16.70	6.57	**0.07**	350p5c	38.96	**0.05**	0.55
3dens	36.74	**0.03**	0.07	450p4c	79.10	**0.06**	1.32
500p3c	50.63	**0.06**	2.07	600p3c	69.33	**0.08**	2.52
900p5c	953.11	**0.14**	6.72	97p	24.90	24.91	**0.06**
convdensity	24.93	0.82	**0.12**	convexo	28.97	**0.03**	0.20
face	472.43	472.44	**5.36**	moreshapes	112.62	**1.01**	1.34
numbers	521.06	521.07	**14.12**	numbers2	2047.18	**0.27**	23.59
outliers	28.08	0.40	**0.21**	*outliers_ags*	12.46	12.48	**0.09**

Para instâncias do conjunto DS3, o algoritmo apresentou, soluções com tempos menores do que da literatura para 36% das instâncias, conforme os dados apresentados pela Tabela 4.10.

Tanto na tabela 4.8, 4.9, 4.10, o algoritmo se mostra competitivo, umas vez que não apresenta nenhum tempo pior que do que todos os algoritmos comparados.

Portanto, é possível perceber que o algoritmo apresenta tempos de execução viáveis e competitivos com o da literatura.

5 Conclusões e Trabalhos Futuros

Este trabalho apresentou um algoritmo para resolver o Problema de Clusterização Automática baseando-se na combinação de duas metaheurísticas: ILS e VND. No algoritmo proposto, ILS-VND, a etapa de busca local do ILS é substituída pela metaheurística VND.

Para avaliar o algoritmo proposto, 82 instâncias com dados numéricos, número m de atributos variando de 2 a 13 e número n de objetos variando de 30 a 2000, foram utilizadas, obtidas de [26]. Os resultados do ILS-VND foram comparados com os resultados dos algoritmos heurísticos AECBL1 [3], ILS-FJGP,ILS-DBSCAN e HH apresentados em [26] .

Para o conjunto de testes realizados, o método construtivo se mostrou eficiente para encontrar o número de clusters de instâncias comportadas, onde o método acertou o número de cluster em 100% desses casos. O método construtivo se mostrou eficiente, contribuindo com 100% no valor da silhueta final para mais de 50% das instâncias. A fase pós-construtivo (busca local e perturbação) se mostrou muito eficiente para determinadas bases, apresentando uma melhoria em relação a solução inicial de mais de 300% na instância $face$, por exemplo.

Não foi feita uma análise quanto à calibragem dos parâmetros utilizados para gerar os resultados, e um estudo sobre esta calibragem pode impactar em uma melhoria nos resultados apresentados. Contudo, com a configuração de parâmetros utilizada, o ILS-VND já apresentou um valor de silhueta competitivo e até maiores do que as abordagens da literatura em várias instâncias do conjunto de testes.

No primeiro conjunto de instâncias, o ILS-VND encontrou uma clusterização melhor do que os algoritmos comparados somente para a instância *yeast*, onde o algoritmo apresentou silhueta no valor de 0.657, enquanto a maior entre os outros algoritmos comparados foi 0.628.

No conjunto DS2, o algoritmo apresentou 33% soluções com o índices de silhueta maiores ou iguais do que dos algoritmos da literatura. Já no conjunto DS3, em 45% das instâncias, o algoritmo apresentou um valor para silhueta mais alto ou igual do que os

outros algoritmos.

O algoritmo proposto foi executado em um computador com uma configuração inferior aos testes realizados por [26], entretanto 64% das instâncias obtiveram seus resultados em menos tempo do que os algoritmos da literatura. Os tempos de execução do algoritmo se mostraram competitivos quanto aos da literatura. Demonstrando que os resultados são gerados em tempos viáveis.

Uma sugestão para trabalho futuro é uma alteração no método de acoplamento, onde clusters compostos por um único elemento são acoplados em outros sorteados aleatoriamente. O fato de ser randômico pode piorar muito a solução, portanto, pode ser utilizado um critério como o utilizado na perturbação, ou um centroide, por exemplo.

Além disso, um método que poderia ajudar o algoritmo seria o de desacoplamento, onde fosse possível, desassociar elementos ou sub-clusters de um cluster, permitindo que o algoritmo se arrependesse de juntar alguns objetos. O algoritmo pode terminar a clusterização com objetos insatisfeitos na solução. Este método poderia, assim como na busca local, utilizar os objetos insatisfeitos para desacoplar de outro cluster.

O critério de parada do laço da perturbação do ILS foi estabelecido como sendo um número fixo de iterações. Este critério poderia ser substituído por tempo, ou ainda, um estudo deverá ser feito no sentido de se avaliar o número de iterações sem melhora. Assim, o algoritmo poderia buscar mais soluções para determinadas instâncias em que o processamento foi rápido e não precisaria ficar tanto tempo em outras instâncias.

O ILS-VND foi implementado para trabalhar com dados numéricos e categóricos, contudo o estudo foi feito somente para dados numéricos. Como trabalho futuro, sugere-se testar o algoritmo com instâncias em cujos atributos constam dados categóricos.

O método construtivo encontrou bons resultados, em um tempo de execução rápido, portanto, um GRASP, pode ser promissor, utilizando as soluções geradas por este método.

A Tabelas de Contribuição dos Algoritmos no Índice Silhueta

Este apêndice contém as tabelas de comparação dos resultados obtido pelo Construtivo e o resultado final do algoritmo. Estas tabelas foram geradas para permitir uma análise do ganho que as características do ILS, com os métodos de Busca Local e Perturbação, forneceram para o valor do índice silhueta.

Tabela A.1: Comparação da solução inicial (S_0) e final(S) dos resultados gerados para instâncias de DS1

Instâncias	K_0	K	$Silhueta(S_0)$	$Silhueta(S)$	Ganho Pós-construtivo
200DATA	3	3	0.822	0.822	0
gauss9	3	3	0.148	0.383	**159%**
iris	2	2	0.630	0.630	0
maronna	2	2	0.539	0.539	0
ruspini	4	4	0.740	0.740	0
spherical_4d3c	4	4	0.689	0.689	0
vowel2	2	2	0.348	0.405	**17%**
wine	2	2	0.269	0.385	**43%**
yeast	2	2	0.657	0.657	0

Tabela A.2: Comparação da solução inicial (S_0) e final(S) dos resultados gerados para instâncias de DS2 - parte 1

Instâncias	K_0	K	$Silhueta(S_0)$	$Silhueta(S)$	Ganho Pós-construtivo
100p2c1	2	2	0.649	0.673	**4%**
100p3c	3	3	0.784	0.784	0
100p3c1	4	3	0.455	0.589	**29%**
100p5c1	10	9	0.657	0.692	**5%**
100p7c	7	7	0.804	0.804	0
100p7c1	13	12	0.349	0.509	**46%**
100p8c1	14	12	0.382	0.529	**39%**
100p10c	10	10	0.835	0.835	0
200p2c1	2	2	0.741	0.765	**3%**
200p3c1	3	3	0.653	0.684	**5%**
200p4c	4	4	0.773	0.773	0
200p4c1	6	6	0.679	0.705	**4%**
200p7c1	11	11	0.432	0.585	**35%**
200p8c1	13	13	0.450	0.577	**28%**
200p12c1	13	13	0.403	0.591	**47%**
300p2c1	2	2	0.771	0.778	**1%**
300p3c	3	3	0.767	0.767	0
300p3c1	3	3	0.652	0.679	**4%**
300p4c1	2	2	0.569	0.569	0
300p6c1	11	11	0.564	0.631	**12%**
300p10c1	18	18	0.475	0.572	**20%**
300p13c1	14	14	0.481	0.592	**23%**
400p3c	3	3	0.818	0.818	0
400p4c1	2	2	0.377	0.465	**23%**
400p17c1	30	30	0.335	0.464	**39%**
500p3c	3	3	0.842	0.842	0

Tabela A.3: Comparação da solução inicial (S_0) e final(S) dos resultados gerados para instâncias de DS2- parte 2

Instâncias	K_0	K	$Silhueta(S_0)$	$Silhueta(S)$	Ganho Pós-construtivo
500p4c1	4	4	0.606	0.627	**3%**
500p6c1	10	8	0.537	0.583	**9%**
500p19c1	43	43	0.306	0.460	**50%**
600p3c1	20	14	0.485	0.548	**13%**
600p15c	15	15	0.782	0.782	0
700p4c	4	4	0.789	0.789	0
700p15c1	32	26	0.576	0.656	**14%**
800p4c1	18	18	0.446	0.446	0
800p10c1	86	84	0.145	0.391	**169%**
800p18c1	38	29	0.575	0.657	**14%**
800p23c	23	23	0.788	0.788	0
900p5c	5	5	0.695	0.698	0
900p12c	12	12	0.842	0.842	0
1000p5c1	6	6	0.371	0.599	**62%**
1000p6c	6	6	0.751	0.751	0
1000p14c	14	14	0.828	0.828	0
1000p27c1	83	82	0.292	0.432	**48%**
1100p6c1	26	26	0.445	0.445	0
1300p17c	17	17	0.822	0.822	0
1500p6c1	34	32	0.287	0.377	**31%**
1800p22c	22	22	0.801	0.801	0
1900p24c	24	24	0.797	0.797	0
2000p9c1	62	60	0.254	0.378	**49%**
2000p11c	11	11	0.724	0.724	0
2000p26c	26	26	0.800	0.800	0

Tabela A.4: Comparação da solução inicial (S_0) e final(S) dos resultados gerados para instâncias de DS3

Instâncias	K_0	K	$Silhueta(S_0)$	$Silhueta(S)$	Ganho Pós-construtivo
$outliers\_ags$	2	2	0.731	0.805	10%
outliers	2	2	0.769	0.769	0
numbers2	11	11	0.544	0.548	1%
numbers	9	9	0.477	0.504	6%
moreshapes	6	6	0.729	0.729	0
face	18	18	0.132	0.528	301%
convexo	3	3	0.656	0.656	0
convdensity	3	3	0.845	0.845	0
2000p11c	11	11	0.724	0.724	0
1000p6c	6	6	0.751	0.751	0
900p5c	5	5	0.699	0.699	0
600p3c	3	3	0.752	0.752	0
500p3c	3	3	0.842	0.842	0
450p4c	4	4	0.766	0.766	0
350p5c	5	5	0.730	0.730	0
300p4c	4	4	0.764	0.764	0
181p	6	6	0.737	0.737	0
157p	4	4	0.631	0.662	5%
97p	3	3	0.701	0.715	2%
30p	11	11	0.679	0.800	18%
3dens	2	2	0.698	0.698	0
2face	2	2	0.464	0.464	0

B Melhores resultados, dentre as 30 sementes, encontrados pelo algoritmo

Tabela B.1: Melhores Resultados para instâncias de DS1

Instância	Silhueta	K	Tempo(s)
200DATA	0.822	3	0.166
gauss9	0.383	3	15.707
iris	0.630	2	0.123
maronna	0.539	4	0.109
ruspini	0.740	4	0.019
spherical_4d3c	0.689	4	0.763
vowel2	0.405	2	20.964
wine	0.385	2	8.869
yeast	0.657	2	146.016

Tabela B.2: Melhores Resultados para instâncias de DS2-parte 1

Instância	Silhueta	K	Tempo(s)
100p2c1	0.673	2	0.032
100p3c	0.784	3	0.032
100p3c1	0.589	3	0.797
100p5c1	0.692	9	0.856
100p7c	0.804	7	0.030
100p7c1	0.509	12	0.285
100p8c1	0.529	12	0.153
100p10c	0.835	10	0.031
200p2c1	0.765	2	0.118
200p3c1	0.684	3	0.225
200p4c	0.773	4	0.128
200p4c1	0.705	6	0.469
200p7c1	0.585	11	0.239
200p8c1	0.577	13	0.782
200p12c1	0.591	13	0.789
300p2c1	0.778	2	2.558
300p3c	0.767	3	0.368
300p3c1	0.679	3	0.381
300p4c1	0.569	2	4.063
300p6c1	0.631	11	2.064
300p10c1	0.572	18	0.742
300p13c1	0.592	14	0.829
400p3c	0.818	3	0.838
400p4c1	0.465	2	0.915
400p17c1	0.464	30	4.910
500p3c	0.842	3	1.829

Tabela B.3: Melhores Resultados para instâncias de DS2-parte 2

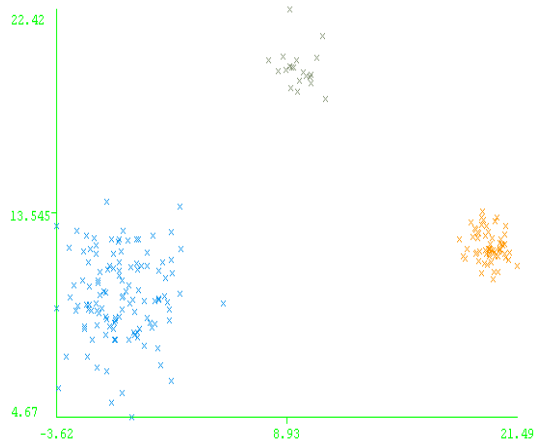
Instância	S	K	T
500p4c1	0.627	4	2.072
500p6c1	0.583	8	2.958
500p19c1	0.460	43	4.914
600p3c1	0.548	14	17.585
600p15c	0.782	15	1.739
700p4c	0.789	4	3.398
700p15c1	0.656	26	24.354
800p4c1	0.446	18	61.553
800p10c1	0.391	84	14.640
800p18c1	0.657	29	23.511
800p23c	0.788	23	4.364
900p5c	0.698	5	6.318
900p12c	0.842	12	6.387
1000p5c1	0.599	6	24.474
1000p6c	0.751	6	8.844
1000p14c	0.828	14	9.022
1000p27c1	0.432	82	173.143
1100p6c1	0.445	26	115.972
1300p17c	0.822	17	16.129
1500p6c1	0.377	32	110.831
1800p22c	0.801	22	45.332
1900p24c	0.797	24	50.698
2000p9c1	0.378	60	350.826
2000p11c	0.724	11	61.505
2000p26c	0.800	26	56.537

Tabela B.4: Melhores Resultados para Instâncias de DS3

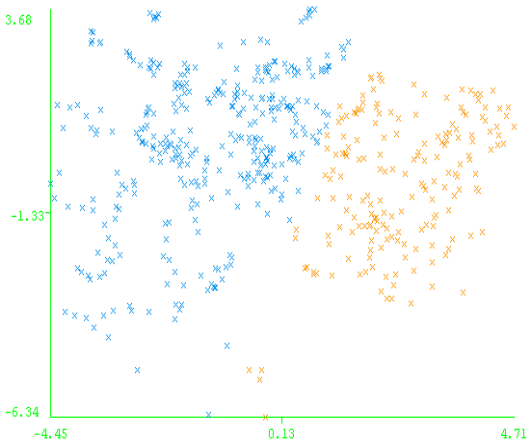
Instância	S	K	T
2face	0.464	2	0.119
3dens	0.698	2	0.063
30p	0.800	11	0.033
97p	0.715	3	0.029
157p	0.662	4	0.072
181p	0.737	6	0.100
300p4c	0.764	4	0.244
350p5c	0.730	5	0.533
450p4c	0.766	4	0.270
500p3c	0.842	3	2.023
600p3c	0.752	3	2.406
900p5c	0.699	5	6.426
1000p6c	0.751	6	8.219
2000p11c	0.724	11	61.189
convdensity	0.845	3	0.116
convexo	0.656	3	0.190
face	0.528	18	2.595
moreshapes	0.729	6	1.271
numbers	0.504	9	9.911
numbers2	0.548	11	23.072
outliers	0.769	2	0.157
outliers_ags	0.805	2	0.041

C Apresentação Gráfica dos Resultados

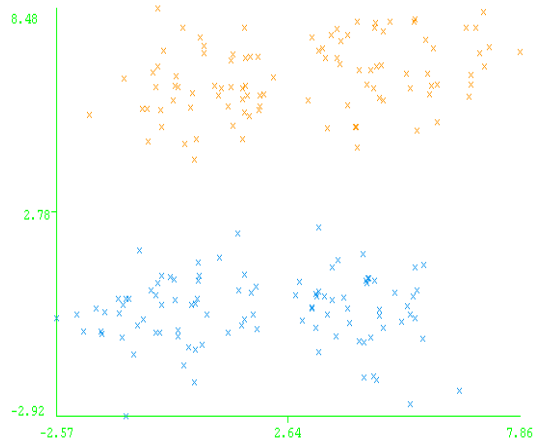
Para representar, visualmente, os resultados do algoritmo implementado, este apêndice apresenta os gráficos das instâncias de duas dimensões do conjunto DS1, DS2 (com menos de 300 pontos) e DS3.



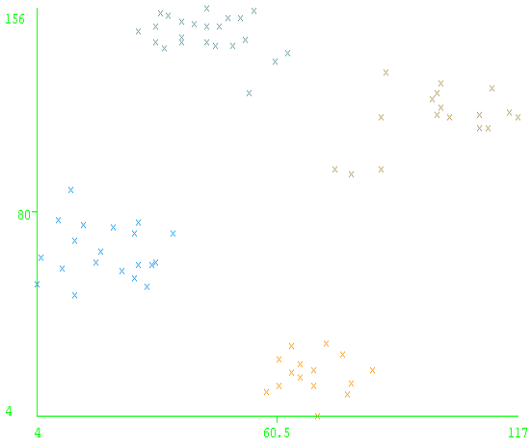
(C.1) 200DATA



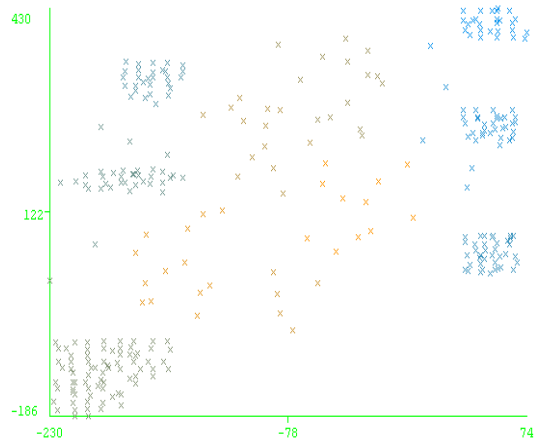
(C.2) vowel2



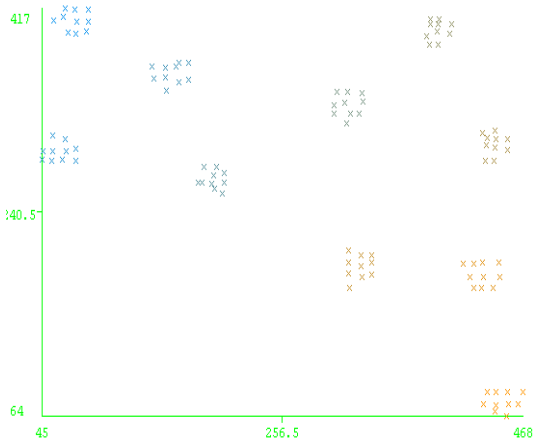
(C.3) maronna



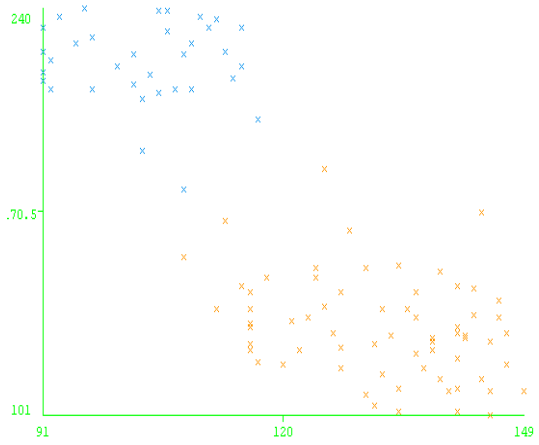
(C.4) ruspini



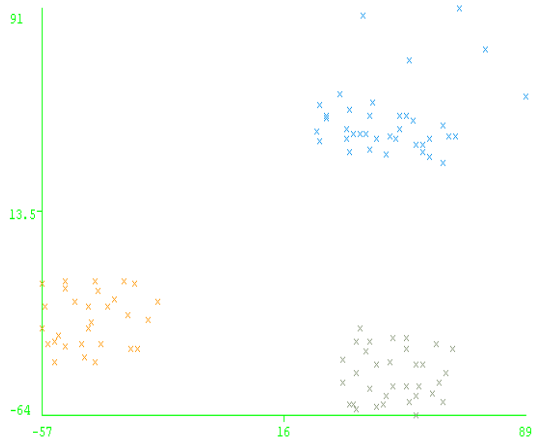
(C.5) 300p6c1



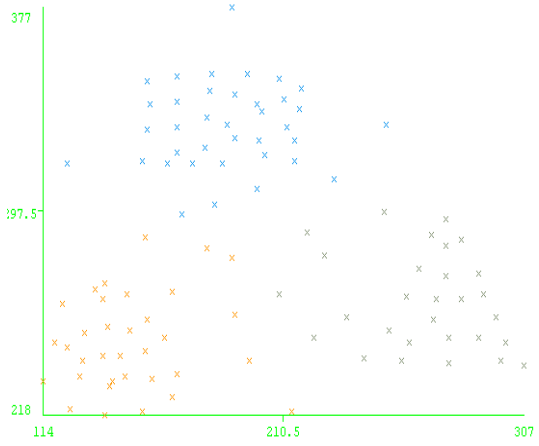
(C.6) 100p10c



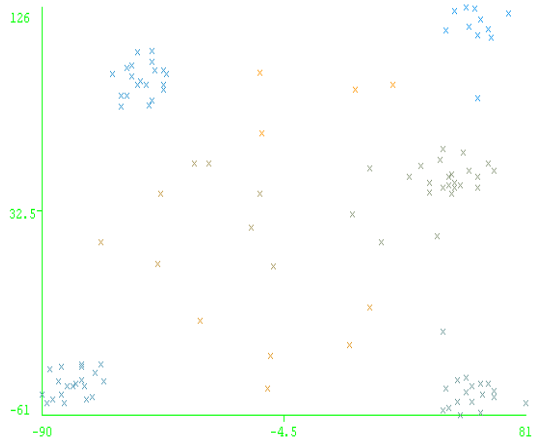
(C.7) 100p2c1



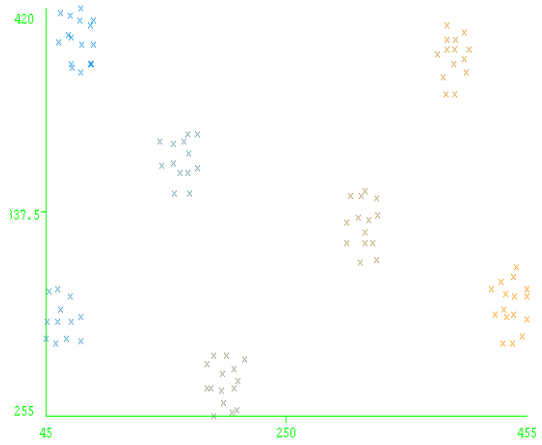
(C.8) 100p3c



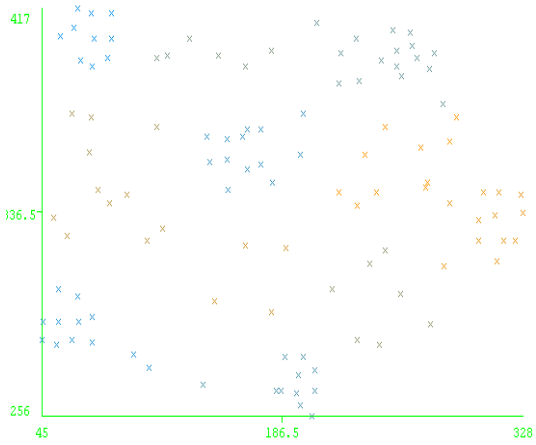
(C.9) 100p3c1



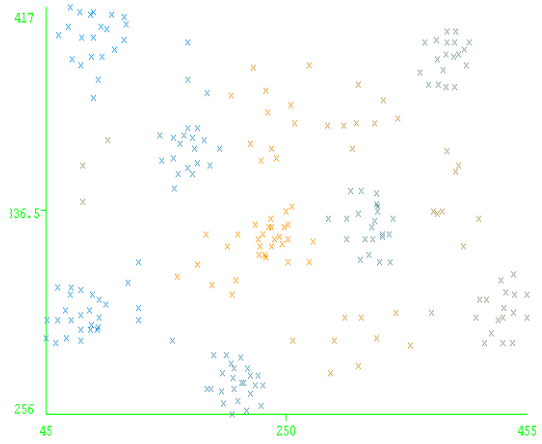
(C.10) 100p5c1



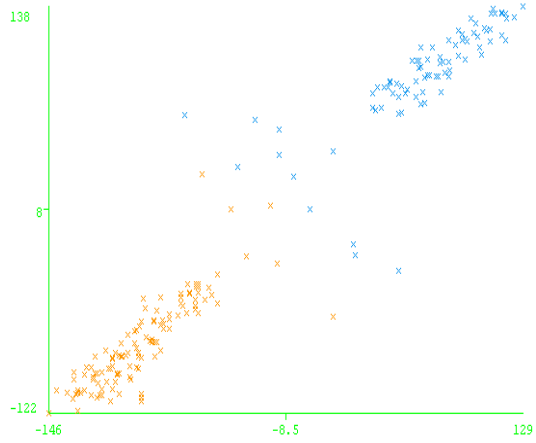
(C.11) 100p7c



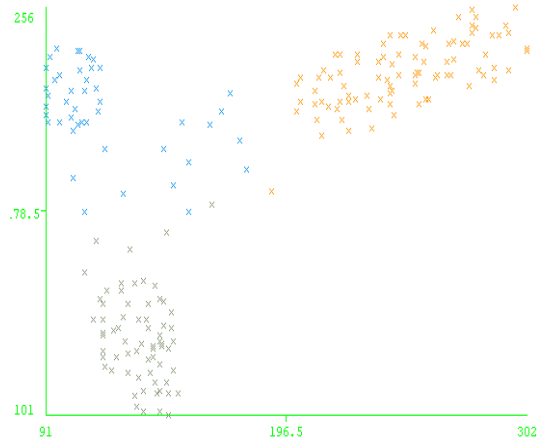
(C.12) 100p7c1



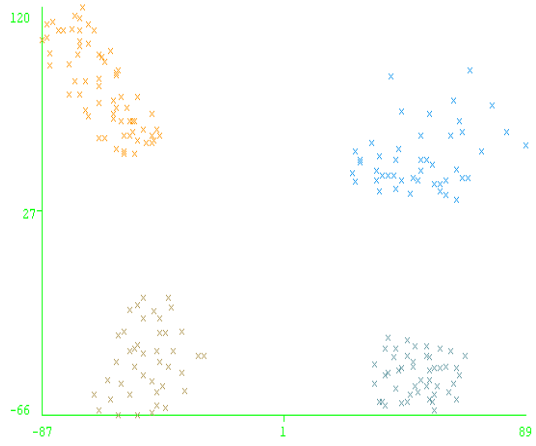
(C.13) 200p12c1



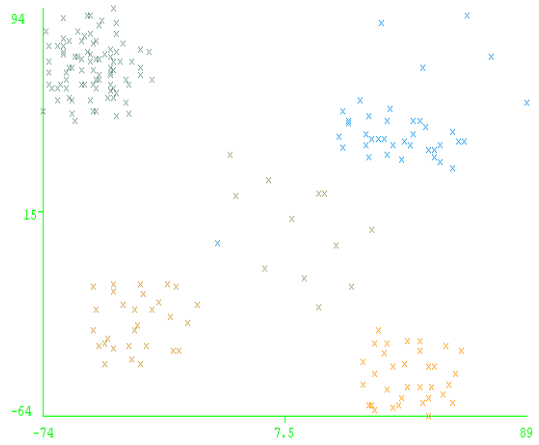
(C.14) 200p2c1



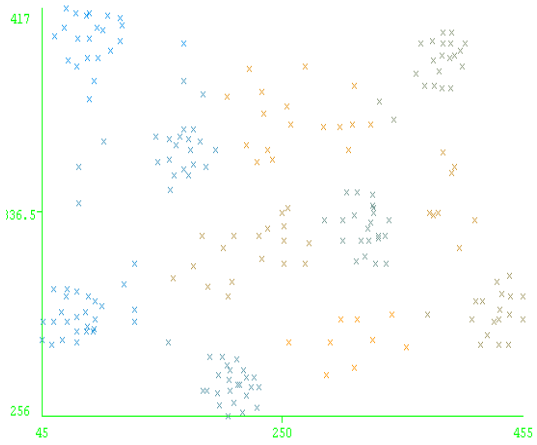
(C.15) 200p3c1



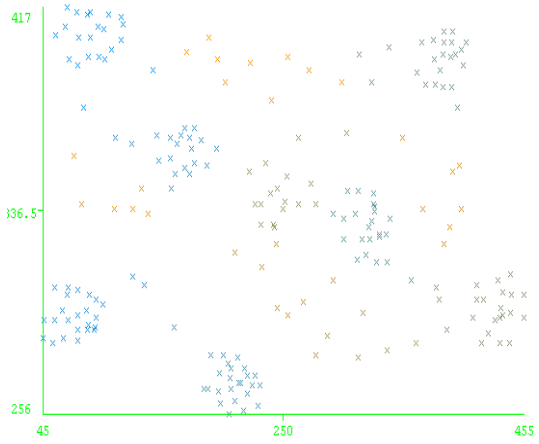
(C.16) 200p4c



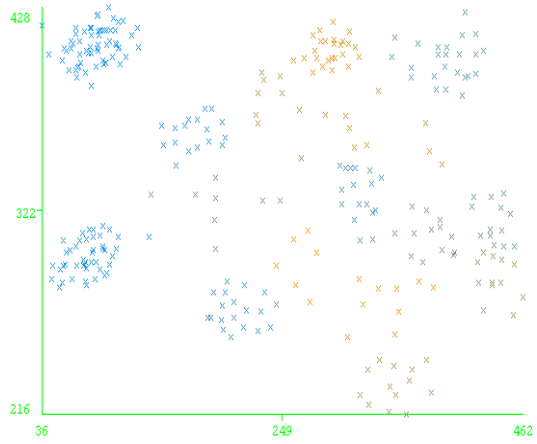
(C.17) 200p4c1



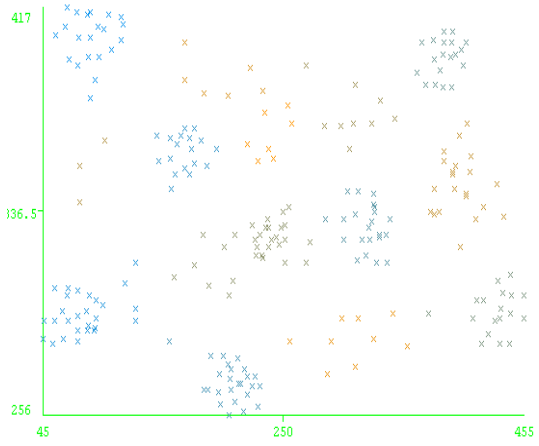
(C.18) 200p7c1



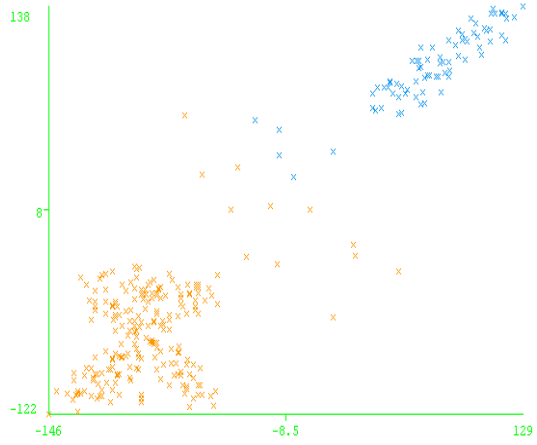
(C.19) 200p8c1



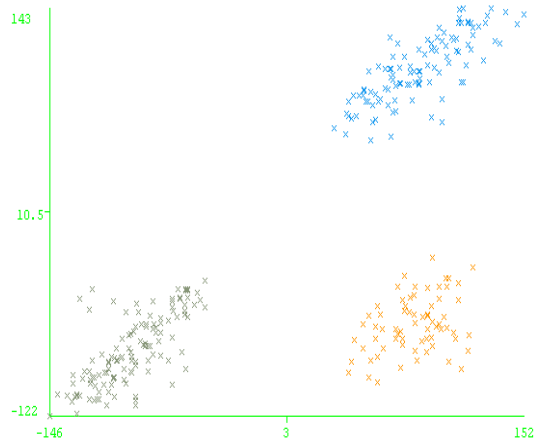
(C.20) 300p10c1



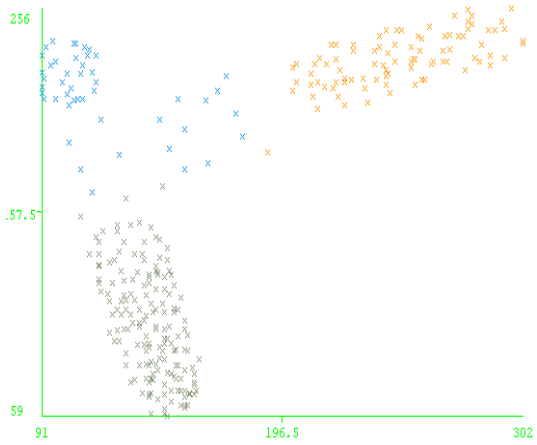
(C.21) 300p13c1



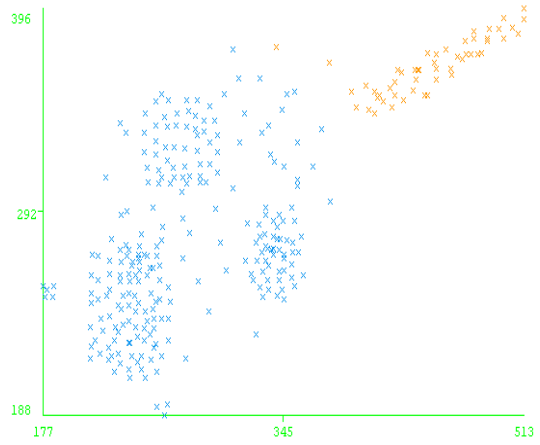
(C.22) 300p2c1



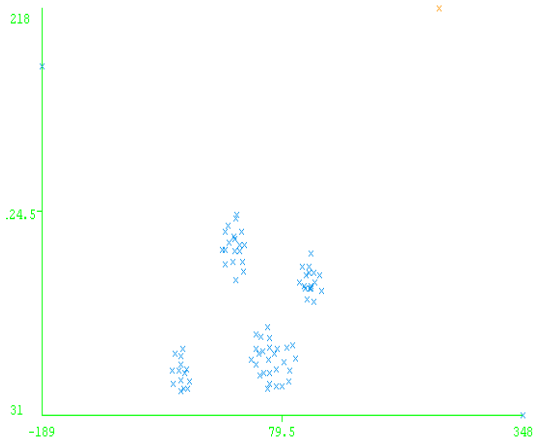
(C.23) 300p3c



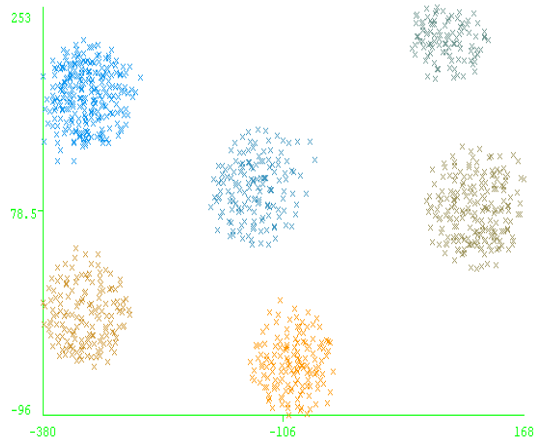
(C.24) 300p3c1



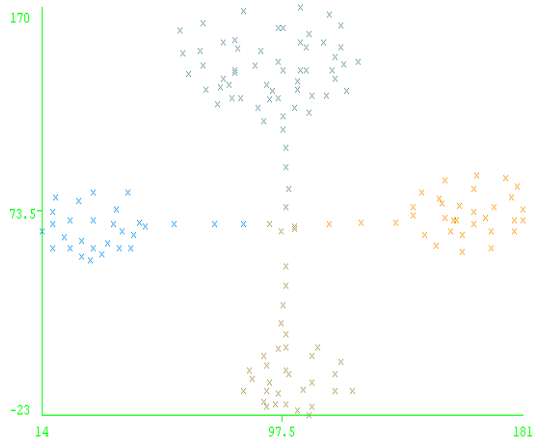
(C.25) 300p4c1



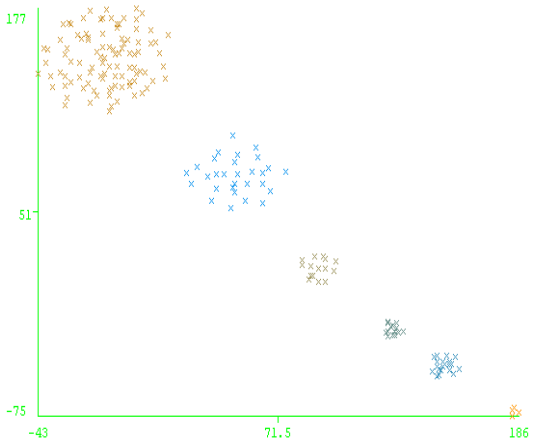
(C.26) *outliers_ags*



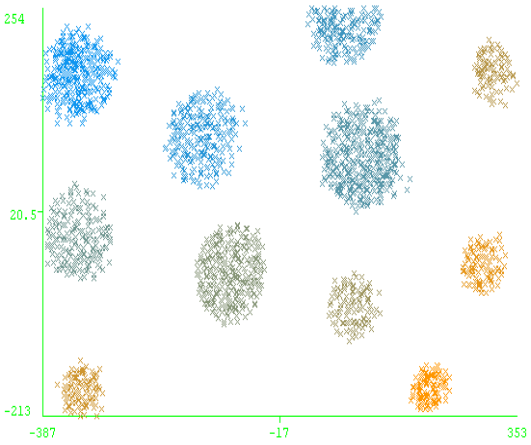
(C.27) 1000p6c



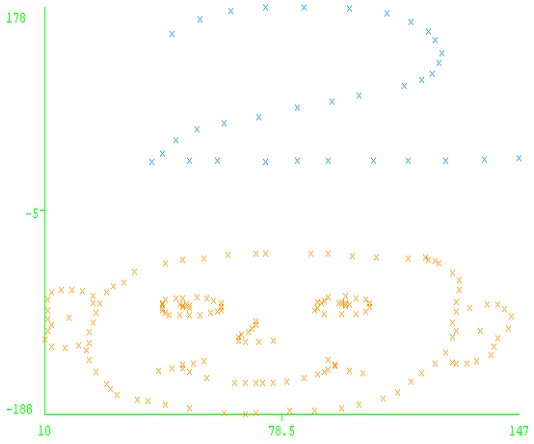
(C.28) 157p



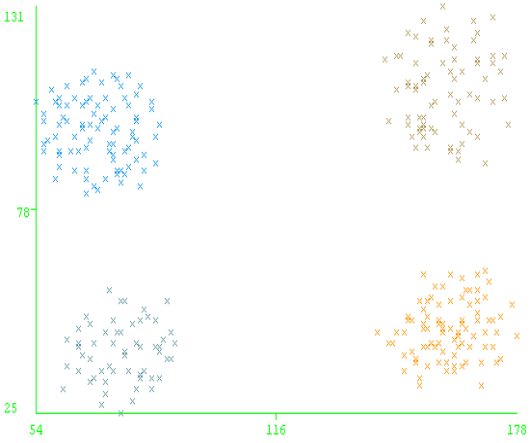
(C.29) 181p



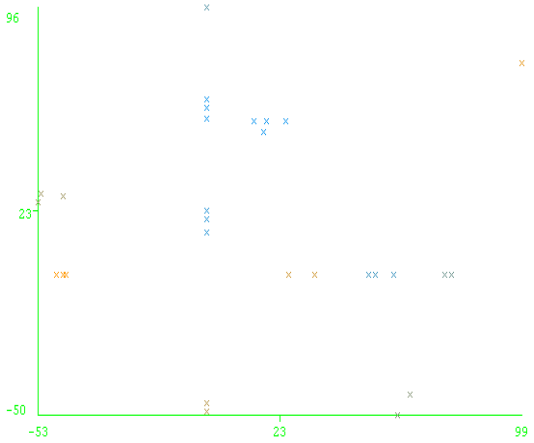
(C.30) 2000p11c



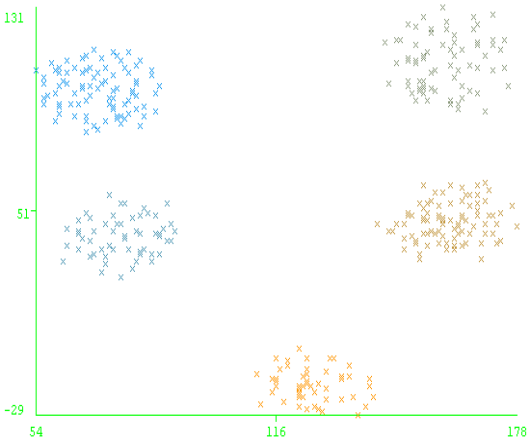
(C.31) 2face



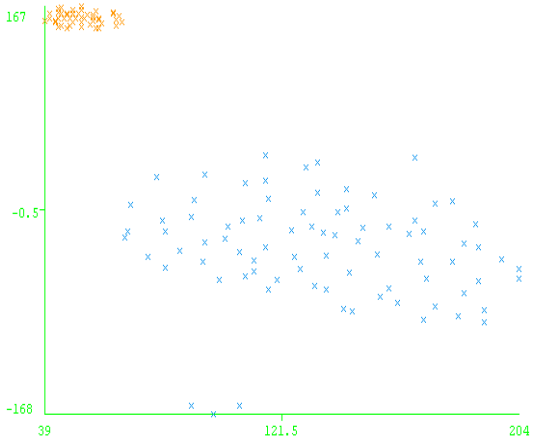
(C.32) 300p4c



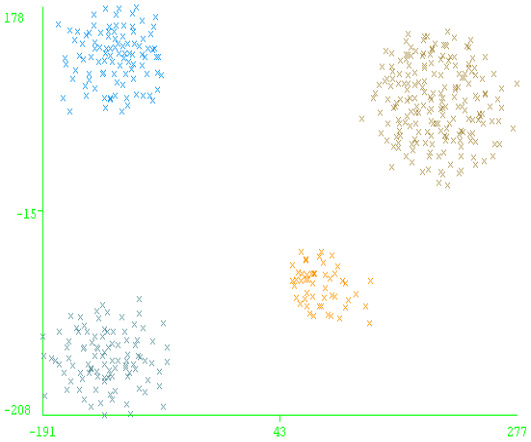
(C.33) 30p



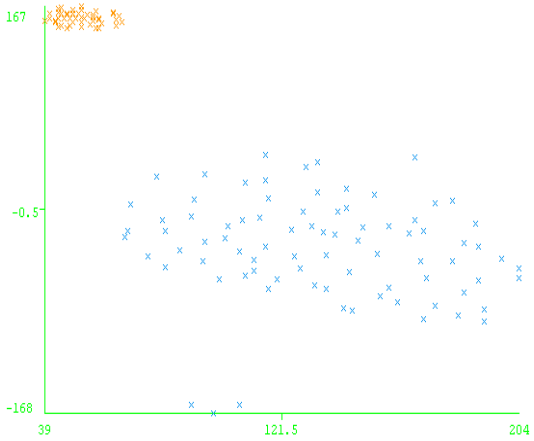
(C.34) 350p5c



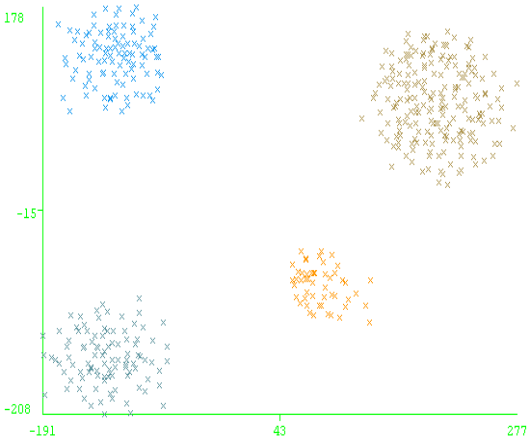
(C.35) 30p



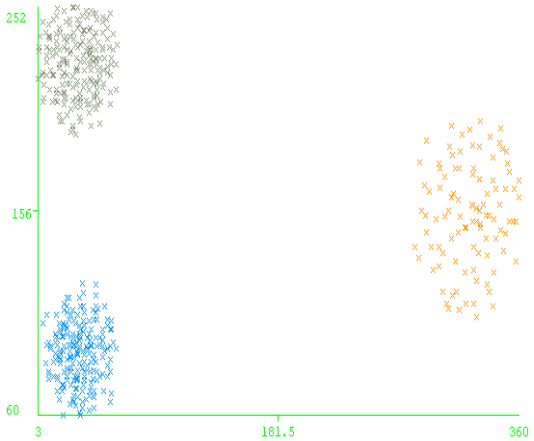
(C.36) 350p5c



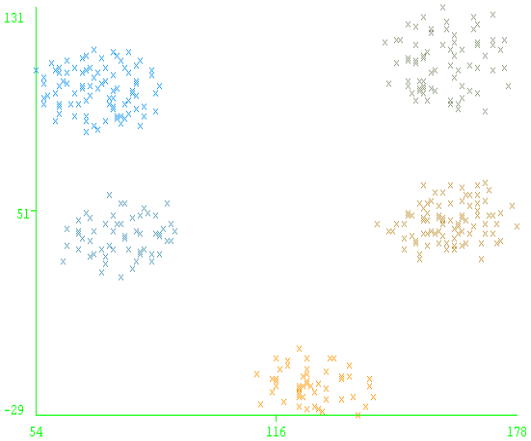
(C.37) 3dens



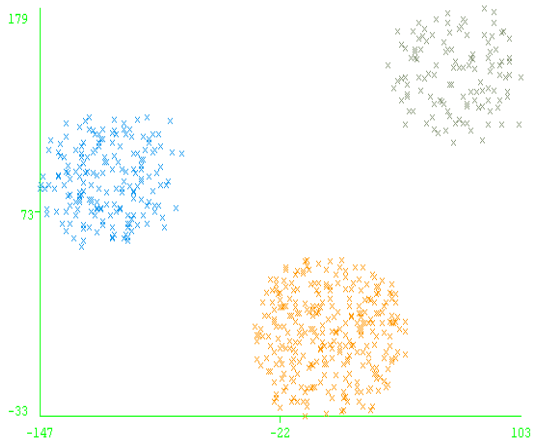
(C.38) 450p4c



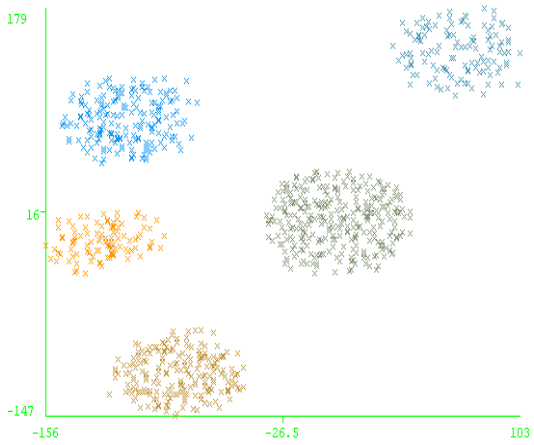
(C.39) 30p



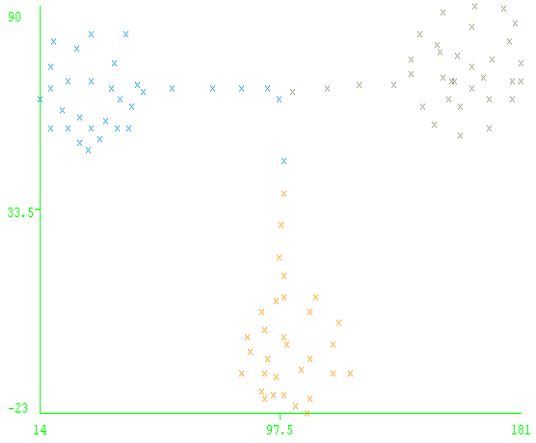
(C.40) 350p5c



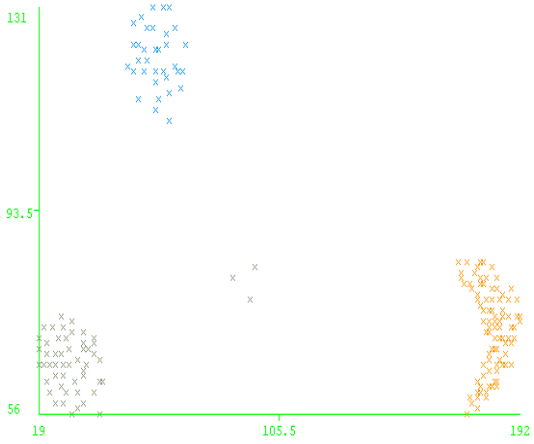
(C.41) 600p3c



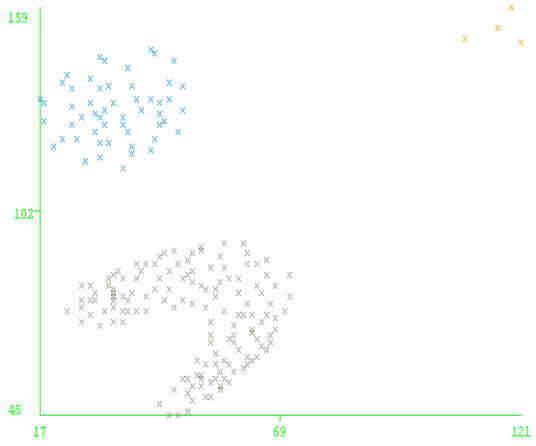
(C.42) 900p5c



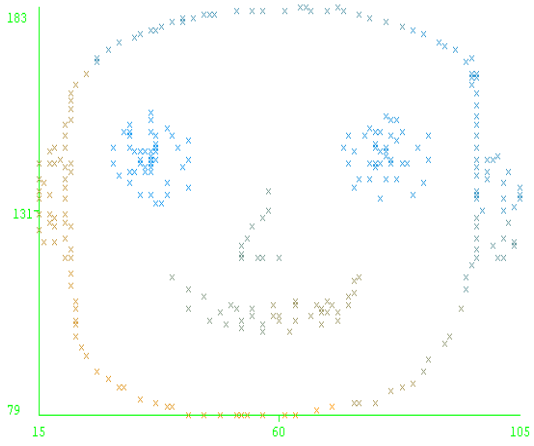
(C.43) 97p



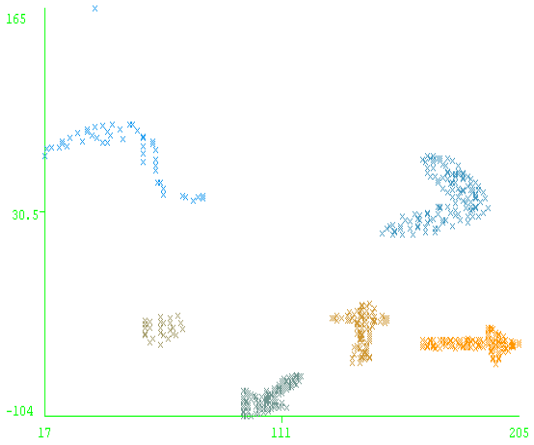
(C.44) convdensity



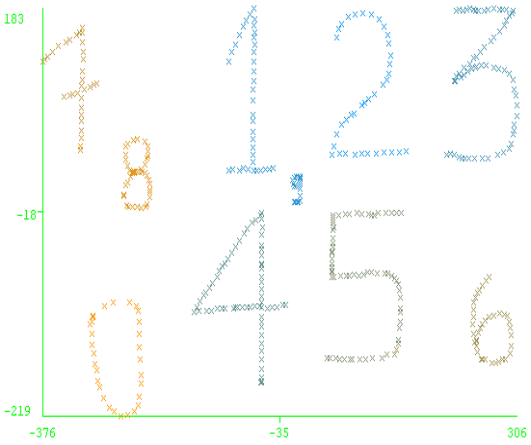
(C.45) convexo



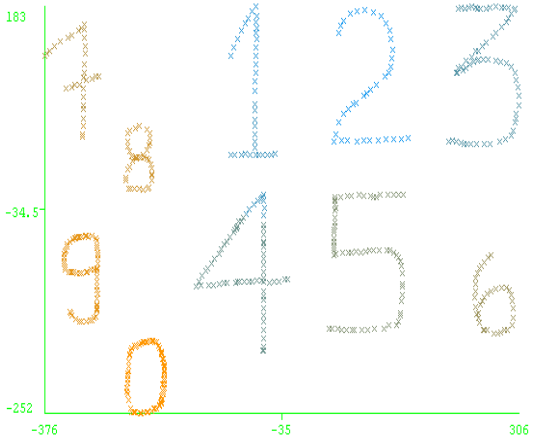
(C.46) face



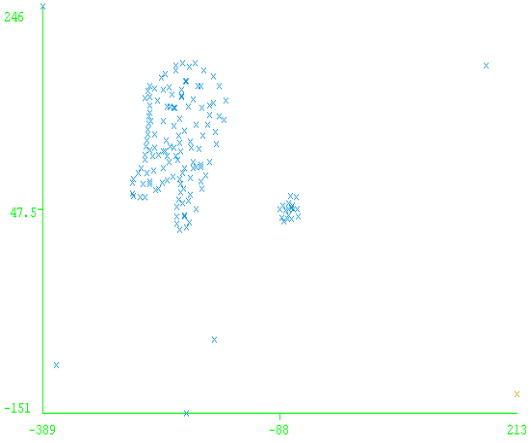
(C.47) moreshapes



(C.48) numbers



(C.49) numbers2



(C.50) outliers

Bibliografia

[1] Charu C Aggarwal e Chandan K Reddy. *Data clustering: algorithms and applications*. CRC Press, 2013.

[2] Peter Brucker. "On the complexity of clustering problems". Em: *Optimization and olperations research*. Springer, 1978, pp. 45–54.

[3] Marcelo D Cruz. "O problema de clusterização automática". Em: (2010).

[4] Marcelo Dib Cruz e Luiz Satoru Ochi. "Um Algoritmo Evolutivo com Memória Adaptativa para o Problema de Clusterização Automática". Em: *Learning and Nonlinear Models* 8.4 (2011), pp. 227–239.

[5] PATRIC FERREIRA da Silva. "Uso de Rede de Kohonen para a Clusterizacao de Objetos de Aprendizagem". Tese de doutorado. Masters. dissertation, Dept. Elect. Eng., UPM, Sao Paulo, 2007.

[6] José André de Moura Brito e Flávio Marcelo Tavares Montenegro. "Aplicações de Técnicas de Pesquisa Operacional em Problemas de Agrupamento do IBGE". Em: *População, espaço e sustentabilidade: contribuições para o desenvolvimento do Brasil, pp.15-33* (2015).

[7] Lando Mendonça di Carlantonio. "Novas metodologias para clusterizaçao de dados". Tese de doutorado. UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2001.

[8] Marcos Neves do Vale. "Agrupamentos de dados: Avaliação de Métodos e Desenvolvimento de Aplicativo para Análise de Grupos". Tese de doutorado. PUC-Rio, 2005.

[9] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". Em: *Annals of human genetics* 7.2 (1936), pp. 179–188.

[10] Jerome Friedman, Trevor Hastie e Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001.

[11] John A Hartigan e JA Hartigan. *Clustering algorithms*. Vol. 209. Wiley New York, 1975.

[12] Eduardo R Hruschka et al. "A genetic algorithm for cluster analysis". Em: *Intelligent Data Analysis* 7.1 (2003), pp. 15–25.

[13] Anil K Jain, M Narasimha Murty e Patrick J Flynn. "Data clustering: a review". Em: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323.

[14] Wojtek J Krzanowski e YT Lai. "A criterion for determining the number of groups in a data set using sum-of-squares clustering". Em: *Biometrics* (1988), pp. 23–34.

[15] Helena R Lourenço et al. "Iterated Local Search". Em: *arXiv preprint math.OC/0102188* (2001).

[16] Ricardo Maronna e Pablo M Jacovkis. "Multivariate clustering procedures with variable metrics". Em: *Biometrics* (1974), pp. 499–505.

[17] DW Van der Merwe e Andries Petrus Engelbrecht. "Data clustering using particle swarm optimization". Em: *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*. Vol. 1. IEEE. 2003, pp. 215–220.

[18] Glenn W Milligan e Martha C Cooper. "An examination of procedures for determining the number of clusters in a data set". Em: *Psychometrika* 50.2 (1985), pp. 159–179.

[19] Nenad Mladenović e Pierre Hansen. "Variable neighborhood search". Em: *Computers & operations research* 24.11 (1997), pp. 1097–1100.

[20] Murilo Coelho Naldi. "Técnicas de combinação para agrupamento centralizado e distribuído de dados". Tese de doutorado. Universidade de São Paulo, 2011.

[21] Edson P Pimentel, Vilma F França e Nizam Omar. "A identificação de grupos de aprendizes no ensino presencial utilizando técnicas de clusterização". Em: *XIV Simpósio Brasileiro de Informática na Educação* (2003).

[22] IK Ravichandra Rao. "Data mining and clustering techniques". Em: *DRTC Workshop on Semantic Web*. Vol. 8. 2003.

[23] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". Em: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[24] Enrique H Ruspini. "Numerical methods for fuzzy clustering". Em: *Information Sciences* 2.3 (1970), pp. 319–350.

[25] Enio Júnior Seidel et al. "Comparação entre o método Ward e o método K-médias no agrupamento de produtores de leite". Em: *Ciência e Natura* 30.1 (2008), p. 7.

[26] Gustavo Silva Semaan. "Algoritmos para o Problema de Agrupamento Automático". Tese de doutorado. Tese de Doutorado, Instituto de Computação, Universidade Federal Fluminense, 2013.

[27] SSRF Soares e Luiz Satoru Ochi. "Um Algoritmo Evolutivo com Reconexão de Caminhos para o Problema de Clusterização Automática". Em: *XII Latin Ibero American Congress on Operations Research*. 2004, pp. 7–13.

[28] ASRF Soares. "Metaheurísticas para o problema de clusterização automática". Em: (2004).

[29] Catherine A Sugar e Gareth M James. "Finding the number of clusters in a dataset: An information-theoretic approach". Em: *Journal of the American Statistical Association* 98.463 (2003), pp. 750–763.

[30] Robert Tibshirani, Guenther Walther e Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic". Em: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[31] Alexandre Tomasi e Luciana Lee. "COMPARAÇÃO DE METAHEURÍSTICAS PARA O PROBLEMA DE CLUSTERIZAÇÃO AUTOMÁTICA". Em: *Anais da V Escola Regional de Informática de Mato Grosso-2014* (2013), p. 64.

[32] Xiaogang Wang, Weiliang Qiu e Ruben H Zamar. "CLUES: A non-parametric clustering method based on local shrinking". Em: *Computational Statistics & Data Analysis* 52.1 (2007), pp. 286–298.

[33] Mingjin Yan. "Methods of determining the number of clusters in a data set and a new clustering criterion". Tese de doutorado. Virginia Polytechnic Institute e State University, 2005.