ANÁLISE DE CLUSTERIZAÇÃO: ESTUDO COMPARATIVO DE MECANISMOS DE AGRUPAMENTO USANDO O ALGORITMO DE COLÔNIA DE FORMIGAS

Trabalho de Conclusão de Curso

Engenharia da Computação

Ruben Neri de Araújo
Orientador: Prof. Dr. Mêuser Jorge Silva Valença

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

RUBEN NERI DE ARAÚJO

ANÁLISE DE CLUSTERIZAÇÃO: ESTUDO COMPARATIVO DE MECANISMOS DE AGRUPAMENTO USANDO O ALGORITMO DE COLÔNIA DE FORMIGAS

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Novembro de 2012.

Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação
Projeto de Final de Curso

ESCOLA POLITÉCNICA
DE PERNAMBUCO

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 12 de 12 de 2012, às 9:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente Ruben Neri de Araújo, orientado pelo professor Mêuser Jorge Silva Valença, sob título Análise de Clusterização: Estudo Comparativo de Mecanismos de Agrupamento Usando o Algoritmo de Colônia de Formigas, a banca composta pelos professores:

Sérgio Murilo Maciel Fernandes

Mêuser Jorge Silva Valença

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 10 (DEZ.)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 03 dias para entrega da versão final da monografia a contar da data deste documento.

SÉRGIO MURILO MACIEL FERNANDES

MÊUSER JORGE SILVA VALENÇA

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Agradecimentos

Em primeiro lugar, agradeço a Deus por permitir a conquista desta etapa acadêmica em minha vida.

Agradeço aos meus pais e familiares, por estarem sempre presentes ao longo desta trajetória.

Agradecimentos especiais também ao Prof. Dr. Mêuser, meu orientador, por mostrar o caminho do aprendizado necessário para elaboração deste projeto.

Enfim, a todos que de alguma forma contribuíram com este momento.

Resumo

A tarefa de agrupamento de dados é uma das atividades mais comuns na natureza. O comportamento coletivo de insetos, em especial, tem motivado o surgimento de diversos algoritmos de clusterização, beneficiando muitas áreas, entre elas, a computação. O Algoritmo de Agrupamento Simples por Colônia de Formigas (SACA – *Standard Ant Clustering Algorithm*) da espécie *Pheidolle pallidula* tem sido alvo de alguns estudos. O objetivo deste projeto é de estudar o SACA bem como algumas variações do mesmo no intuito de promover melhorias na formação final dos agrupamentos. Para isto, este trabalho realiza através de experimentos e testes estatísticos (t de *Student* e *Wilcoxon*) um estudo comparativo do algoritmo SACA em sua forma original, do SACA com melhoria no parâmetro α e do SACA com medição de insucessos em um agente que deixa o objeto carregado em uma nova posição. Neste caso, foram utilizadas duas diferentes bases de dados e o parâmetro de avaliação dos algoritmos foi a taxa de acerto dos mesmos. Os experimentos foram executados por meio do uso do software Matlab. Os resultados obtidos mostraram que o SACA com medição de insucessos se mostrou melhor em alguns experimentos e pior em outros. Contudo, apesar de os resultados mostrarem que mais estudos nesta área são necessários, este tipo de parametrização no algoritmo SACA mostrou ser uma possível alternativa de agrupamento de dados com redução de custos computacionais e operacionais indesejáveis.

Abstract

The task of data clustering is one of the most common activities in nature. The collective behavior of insects, in particular, has motivated the emergence of various clustering algorithms, benefiting many areas, among them the computation. The Standard Ant Clustering Algorithm (SACA) based on ant colony *Pheidolle pallidula* the species has been the object of some studies. The objective of this project is to study the SACA as well as some variations of the same in order to promote improvements in the final formation of clusters. Therefore, this task is accomplished through experiments and statistical tests (Student t test and Wilcoxon) a comparative study of the algorithm SACA in its original form, the SACA with improved parameter α, and SACA with measurement failures in an agent leaving the loaded object in a new position. In this case, it used two different databases and parameter evaluation of the algorithms was the hit rate them. The experiments were performed by using the Matlab software. The results showed that the SACA with measurement of failures was better in some experiments and worse in others. However, although the results show that more studies in this area are needed, this type of parameterization in the algorithm SACA shown to be a possible alternative data clustering with reduction computational and operational costs undesirable.

Sumário

Capítulo 1 Introdução **1**

1.1 Motivação e Problema 1

1.2 Objetivos 3

1.2.1 Objetivo Geral 3

1.2.2 Objetivos Específicos 3

1.3 Estrutura da Monografia 3

Capítulo 2 Revisão Bibliográfica **4**

2.1 Processo de Agrupamento 4

2.1.1 Objeto e Atributo 5

2.1.2 Agentes 5

2.1.3 Distância e Similaridade 5

2.1.4 Matriz de Dissimilaridade 6

2.2 Métodos de Agrupamento 6

2.2.1 Fundamentados em Particionamento 7

2.2.2 Fundamentados em Hierarquia 7

2.2.3 Fundamentados em Densidade 7

2.2.4 Fundamentados em Estrutura de Grade 8

2.2.5 Fundamentados em Modelos 8

2.3 Algoritmo de Agrupamento Simples por Colônia de Formigas 8

2.3.1 Princípios Básicos do SACA 11

2.3.2 Funcionamento do SACA 12

2.3.3 Pseudocódigo do SACA 13

2.3.4 Funções e Parâmetros do SACA 14

2.4 Principais Modificações Propostas e Parametrizações para o Algoritmo de Agrupamento Simples por Colônia de Formigas 17

2.4.1 Modificação da Função Densidade 17

2.4.2 Memória dos Agentes 18

2.4.3 Raio de Percepção Crescente 18

2.4.4 Separação Espacial 18

2.4.5 Parâmetro α 19

2.4.6 Tamanho da Grade 20

2.4.7 Quantidade de Iterações 20

Capítulo 3 Metodologia 21

3.1 Bases de Dados 21

3.2 Implementação 23

3.2.1 Descrição da Implementação 23

3.2.2 Cenários de Execução 25

3.3 Planejamento do Experimento 26

3.3.1 Variáveis e Escalas 27

3.3.2 Hipóteses Nulas e Alternativas 28

3.3.3 Instrumento 29

3.3.4 Metodologia de Análise 29

Capítulo 4 Testes e Resultados **31**

4.1 Experimentos na Base de Dados Iris 31

4.1.1 Experimento I 31

4.1.2 Experimento II 33

4.1.3 Experimento III 35

4.2 Experimentos na Base de Dados *Wine* 37

4.2.1 Experimento IV 37

4.2.2 Experimento V 39

4.2.3 Experimento VI 40

4.3 Aplicação dos Testes e Análise dos Resultados 42

4.3.1 Testes Estatísticos na Base de Dados Iris 42

4.3.2 Testes Estatísticos na Base de Dados *Wine* 43

4.3.3 Análise dos Resultados 44

Capítulo 5 Conclusões e Trabalhos Futuros **46**

Bibliografia **48**

Apêndice A Resultados dos Experimentos na Base de Dados Iris **50**

Apêndice B Resultados dos Experimentos na Base de Dados *Wine* **53**

Apêndice C **55**

Código do Matlab para Agrupamento da Base de Dados Iris através do SACA Original **55**

Índice de Figuras

Figura 1. Formação de cemitérios de formigas da espécie *Pheidolle pallidula* (HARTMANN [8])....9

Figura 2. Ninhos de formigas da espécie *Leptothorax unifasciatus* (HARTMANN [8]) 10

Figura 3. Pseudocódigo do SACA (HANDL [7])....13

Figura 4. Probabilidade de pegar um objeto em função de $f(i)$ (LAURO [9])....15

Figura 5. Probabilidade de deixar um objeto em função de $f(i)$ (LAURO [9])....16

Figura 6. Representação da vizinhança de um agente para $\sigma = 3$ (HARTMANN [8]) 16

Figura 7. Fluxograma do SACA (LAURO [9]) 24

Figura 8. Clusterização do Experimento I....32

Figura 9. *Boxplot* da taxa de acerto do Experimento I....33

Figura 10. Clusterização do Experimento II....34

Figura 11. *Boxplot* da taxa de acerto do Experimento II....34

Figura 12. Clusterização do Experimento III....35

Figura 13. *Boxplot* da taxa de acerto do Experimento III....36

Figura 14. Clusterização do Experimento IV 38

Figura 15. *Boxplot* da taxa de acerto do Experimento IV....38

Figura 16. Clusterização do Experimento V 39

Figura 17. Clusterização do Experimento VI 40

Figura 18. *Boxplot* da taxa de acerto do Experimento VI.....41

Índice de Tabelas

Tabela 1. Amostra normalizada da base de dados Iris 22

Tabela 2. Amostra normalizada da base de dados *Wine* 23

Tabela 3. Parâmetros do Experimento I 31

Tabela 4. Parâmetros do Experimento II 33

Tabela 5. Parâmetros do Experimento III 36

Tabela 6. Parâmetros do Experimento IV 37

Tabela 7. Parâmetros do Experimento V 39

Tabela 8. Parâmetros do Experimento VI 41

Tabela 9. Resumo dos Testes Estatísticos na Base de Dados Iris 43

Tabela 10. Resumo dos Testes Estatísticos na Base de Dados *Wine* 44

Tabela 11. Taxa de acerto nos experimentos da base de dados Iris 50

Tabela 12. Taxa de acerto nos experimentos da base de dados *Wine* 53

Tabela de Símbolos e Siglas

SACA – Standard Ant Clustering Algorithm (Algoritmo de Agrupamento Simples por Colônia de Formigas)

UCI – University of California, Irvine (Universidade da Califórnia, Irvine)

EMQ – Erro Médio Quadrático

Capítulo 1

Introdução

Este capítulo visa descrever o problema e a motivação para o desenvolvimento deste trabalho. Expõe seus principais objetivos e ao término, mostra o conteúdo dos capítulos seguintes.

1.1 Motivação e Problema

O processo de agrupamento é uma das atividades mais antigas da história da humanidade. Separar objetos com características semelhantes em grupos distintos muitas vezes acontece de forma tão natural e intuitiva que a constatação deste fenômeno por vezes não é percebida.

Na natureza, verifica-se também a existência deste mecanismo de agrupamento. Os insetos, em especial, manifestam através de suas vidas em coletividade, procedimentos comportamentais onde é visível a existência da clusterização[1]. Sabe-se que a complexidade na vida coletiva é resultante do processo de interação dos hábitos simples dos indivíduos que formam a colônia, conforme DORIGO e GAMBARDELLA [4]. Com colônia de formigas, em particular, muitos estudos têm sido desenvolvidos na computação a partir da observação de como as formigas se comportam em suas rotinas diárias.

Através dessas observações, LUMER e FAIETA [11] apresentaram o Algoritmo de Agrupamento Simples por Colônia de Formigas (SACA). A ideia deste algoritmo se espelha no fato que as formigas são capazes de realizar rotinas complexas de agrupamento através da execução simples de tarefas individuais. MONMARCHÉ [12] detalha a vantagem em se usar o SACA para agrupamento pelo

[1] Clusterização: A clusterização corresponde à técnica de se agrupar dados que representam objetos físicos ou abstratos em grupos de objetos com características semelhantes.

fato de o mesmo não necessitar conhecer a classificação inicial das entidades a serem agrupadas em virtude dele ser do tipo não supervisionado[2].

Algumas aplicações têm sido relatadas na literatura ressaltando o uso deste algoritmo, entre elas, a área de robótica tem se destacado. BECKERS, HOLLAND e DENEUBOURG [1] descrevem simulações computacionais feitas do comportamento de clusterização de itens mortos numa colônia de formigas. CASTRO [2], por sua vez, ressalta a importância em se usar no contexto de robótica coletiva, robôs simples e baratos com alta tolerância a falhas no lugar de um único robô com alta capacidade de processamento. Por outro lado, WATTHAYU [15] relata o uso do SACA em um estudo de caso envolvendo a cidade de Bangkok na Tailândia. O objetivo era descobrir rotas otimizadas de percurso entre diversas origens e destinos dentro desta região.

Notáveis contribuições já existem para melhoria deste algoritmo. Entretanto, dentre os principais problemas ora citados, destaca-se a questão da convergência na formação dos agrupamentos. Melhoras sutis foram feitas como a proposta por VIZINE et al. [14], entretanto, nenhuma delas suficientemente eficaz ou em concordância com a principal virtude do SACA que é a simplicidade. Em alguns casos, a causa desta não convergência é atribuída ao critério de parada escolhido na execução do algoritmo.

Em virtude do exposto, estudos para melhoria da convergência do SACA e de suas variantes são importantes visto que promovem o uso eficiente do algoritmo, dirimindo custos computacionais e operacionais indesejáveis.

[2] Algoritmo não supervisionado: Tipo de algoritmo de aprendizagem onde não existe a necessidade de se conhecer previamente a classe de saída dos objetos a serem classificados.

1.2 Objetivos

1.2.1 Objetivo Geral

Comparar diferentes cenários de parametrização do algoritmo baseado em colônia de formigas da espécie *Pheidolle pallidula*.

1.2.2 Objetivos Específicos

- Estudar o mecanismo de agrupamento de dados com o uso da metáfora de colônia de formigas;
- Comparar a taxa de acerto do SACA com variantes do mesmo através de validação estatística;
- Testar diferentes critérios de parada com o objetivo de se avaliar a convergência dos agrupamentos formados.

1.3 Estrutura da Monografia

O Capítulo 2 contempla a fundamentação teórica necessária para o entendimento do presente projeto. No Capítulo 3, está descrito qual metodologia foi empregada nos experimentos. Por sua vez, o Capítulo 4 descreve a aplicação de testes estatísticos e os resultados que foram obtidos. Finalmente, o Capítulo 5 mostra a parte de conclusões e trabalhos futuros.

Capítulo 2
Revisão Bibliográfica

2.1 Processo de Agrupamento

O processo de se agrupar diferentes entidades ou registros em grupos com características similares é conhecido como agrupamento ou clusterização em mineração de dados. Os grupos formados, também conhecidos como *clusters*[3], são o resultado do procedimento classificativo aplicado na base de dados representativa das entidades a serem agrupadas. Geralmente, as entidades agrupadas em um mesmo *cluster* se concentram em uma região densa e possuem características particulares bem próximas entre si.

Técnicas de agrupamento têm sido largamente utilizadas em diferentes campos de estudo tais como reconhecimento de padrões, análise de dados e processamento de imagens. Segundo LAURO [9], aplicações de clusterização no mundo empresarial tem sido úteis no mapeamento do comportamento de clientes sob determinado aspecto de interesse, como por exemplo, o perfil de adimplência dos mesmos. Na Biologia, agrupamentos aplicados a diferentes espécies de plantas têm auxiliado em estudos para a produção de novas vacinas contra doenças. Classificação de documentos utilizando clusterização tem proporcionado novos mecanismos de busca de informação.

Outra importante área onde processos de agrupamento têm sido cada vez mais utilizados é na robótica visto que algumas aplicações neste contexto podem ser executadas através de ações simples e repetitivas (CASTRO [2]). Fazer uso de robôs coletores de lixo de modo que os mesmos sejam capazes de agrupá-lo em

[3] Cluster: Grupo de entidades que é formado resultante do processo de agrupamento.

diversos grupos próprios (metal, vidro, plástico, papel) é apenas uma das muitas possíveis aplicações neste campo de estudo.

Em mineração de dados nem sempre é possível utilizar o processo de agrupamento de forma satisfatória. A natureza da base de dados envolvida em muito influencia neste desempenho. A seguir, será descrito os principais conceitos envolvidos em matéria de clusterização de dados (além dos que já foram vistos) que serão úteis no entendimento desta monografia. Mais informações sobre o tema podem ser obtidas também em HAN e KAMBER [6].

2.1.1 Objeto e Atributo

É comum usar-se na comunidade científica o termo registro, objeto, entidade ou item para se referir ao ente que deverá sofrer o processo de agrupamento. O atributo, por sua vez, corresponde justamente à característica de cada um destes objetos, podendo assumir valores numéricos ou nominais. Neste trabalho, serão utilizados os termos objetos e atributos. Uma base de dados composta por n objetos e t atributos pode ser representada pelo conjunto $\{x_1, x_2, ..., x_n\}$ onde cada x_i corresponde a um vetor $(x_{i1}, x_{i2}, ..., x_{it})$ com seus respectivos x_{ij} representando os atributos de cada objeto.

2.1.2 Agentes

O termo agente serve para designar a entidade que é responsável por realizar propriamente o processo de agrupamento. É sua função mover os objetos durante a clusterização além de ter que fazer a tomada de decisão em ter que deixar o objeto ou levá-lo para uma nova posição. HANDL [7] sugere que o número de agentes seja 10, independentemente da base de dados.

2.1.3 Distância e Similaridade

Em agrupamento de dados, distância e medida de similaridade são importantes conceitos. A similaridade mostra o quão semelhante determinado objeto é de outro, e, através dessa caracterização, é possível quantificar o grau de proximidade entre os mesmos, aqui representado pela distância. A equação 2.1 descreve a métrica que

foi utilizada no presente projeto. Nela, a função $d(i, j)$ corresponde justamente à distância Euclidiana[4] e é calculada a partir dos valores dos atributos dos objetos envolvidos no cálculo. Os argumentos i e j são estes objetos. Os vetores $\{x_{i1}, x_{i2},..., x_{it}\}$ e $\{x_{j1}, x_{j2},..., x_{jt}\}$ representam, respectivamente, os atributos de i e j. Por definição, $d(i, j) \in [0,1]$.

$$d(i, j) = \sqrt{\left|x_{i1} - x_{j1}\right|^2 + \left|x_{i2} - x_{j2}\right|^2 + ... + \left|x_{it} - x_{jt}\right|^2} \quad (2.1)$$

2.1.4 Matriz de Dissimilaridade

A matriz de dissimilaridade representa o grau de dissimilaridade entre os objetos envolvidos no processo de agrupamento. Em outras palavras, cada elemento desta matriz corresponde à distância $d(i, j)$ dos objetos i e j envolvidos no cálculo. Portanto, como consequência, tem-se que para todos os objetos envolvidos no agrupamento $d(i, j) = d(j, i)$ e $d(i, i) = 0$. A distância $d(i, j)$ é um número não negativo que se aproxima de zero quando os objetos são muito similares e cresce à medida que os mesmos apresentam uma maior diferença.

2.2 Métodos de Agrupamento

Atualmente, muitos são os métodos disponíveis para agrupamento de dados. A escolha da melhor técnica à ser empregada está ligada diretamente com a natureza dos dados à serem trabalhados e com a finalidade de aplicação da mesma. De acordo com HAN e KAMBER [6], os métodos de agrupamento são classificados em:

[4] Distância Euclidiana: Métrica utilizada em mineração de dados para quantificar o grau de proximidade entre diferentes objetos através do cálculo da raiz quadrada das diferenças dos quadrados de cada atributo destes objetos envolvidos.

2.2.1 Fundamentados em Particionamento

A ideia de um método baseado em particionamento consiste em dividir-se os n objetos do agrupamento em k partições, também conhecidas como *clusters*, onde os elementos de cada partição possuem características semelhantes. Como consequência do agrupamento, tem-se que $k \leq n$, cada objeto deve fazer parte de apenas uma partição e cada partição deve conter pelo menos um objeto.

A execução deste tipo de técnica de agrupamento ocorre inicialmente com a geração das k partições de acordo com algum parâmetro pré-estabelecido. No algoritmo *k-means* (HAN e KAMBER [6]), por exemplo, os grupos são formados a partir do cálculo da distância em relação aos atributos de cada objeto. Cada objeto então fará parte do *cluster* pelo qual esse valor de distância seja o menor possível comparativamente com os demais *clusters*. Outro algoritmo bem conhecido dentro do contexto de particionamento é o *k-medoid* (HAN e KAMBER [6]).

2.2.2 Fundamentados em Hierarquia

Os métodos de agrupamento baseados em hierarquia possuem dois possíveis tipos de abordagem. Na abordagem *bottow-up*[5], inicialmente cada objeto é dito ser um grupo distinto e então, esses grupos vão se combinando à medida que se analisa um nível de hierarquia superior ao atual. O processo segue até que um único grupo seja formado no mais alto nível da hierarquia ou uma determinada condição de parada do algoritmo seja satisfeita. Por sua vez, a abordagem *top-down*[6] atua no sentido inverso ao descrito anteriormente.

2.2.3 Fundamentados em Densidade

Este tipo de técnica valoriza muito a vizinhança do objeto que está sendo agrupado. A principal diferença deste método para os fundamentados em particionamento é que neste, além do cálculo da distância entre os objetos, é levado em consideração

[5] Bottow-up: Tipo de abordagem em que se inicia do mais baixo nível de uma hierarquia até atingir o mais alto nível.

[6] Top-down: Oposta a Bottow-up. Parte-se do mais alto nível da hierarquia até o mais baixo.

também a vizinhança dos mesmos. Os objetos tendem a se agrupar em regiões cuja vizinhança se torna mais densa e, por outro lado, se desagregar de regiões menos densas.

2.2.4 Fundamentados em Estrutura de Grade

Os métodos baseados em estrutura de grade são representados através de um plano (grade) composto por um número específico de células e os objetos do agrupamento. As células correspondem as possíveis posições pelas quais os objetos ao longo do processo de clusterização podem ocupar. Segundo LAURO [9], este tipo de abordagem tem como principal ganho o tempo de processamento.

2.2.5 Fundamentados em Modelos

Métodos de agrupamento baseados em modelos fazem uso de modelos matemáticos para promover o agrupamento de dados. O objetivo é a busca otimizada de geração dos *clusters*. Para tanto, funções de distribuição de probabilidade são utilizadas nestes tipos de técnicas.

2.3 Algoritmo de Agrupamento Simples por Colônia de Formigas

A coletividade dos insetos representada pelas suas atividades diárias manifesta-se na natureza de diferentes formas. O comportamento social de trabalho em grupo destes seres vivos varia desde a coleta de alimentos e construção de ninhos até a formação de cemitérios e controle térmico da colônia. O perfeito equilíbrio entre os entes envolvidos neste contexto de agrupamento, cada qual executando a função que lhe é devida, é condição fundamental para a consequente sobrevivência dos mesmos.

Diversos tipos de colônias de formigas têm despertado interesse de estudo no meio científico. HARTMANN [8] observou o comportamento de algumas espécies, dentre elas, a *Pheidolle pallidula*. A Figura 1 descreve o comportamento de formação de cemitérios desta espécie de formiga. À medida que pequenos grupos de formigas

mortas vão se formando carregadas pelas formigas operárias, outras formigas mortas vão sendo trazidas e deixadas nestes agrupamentos já existentes, aumentando-os de tamanho no decorrer do tempo.

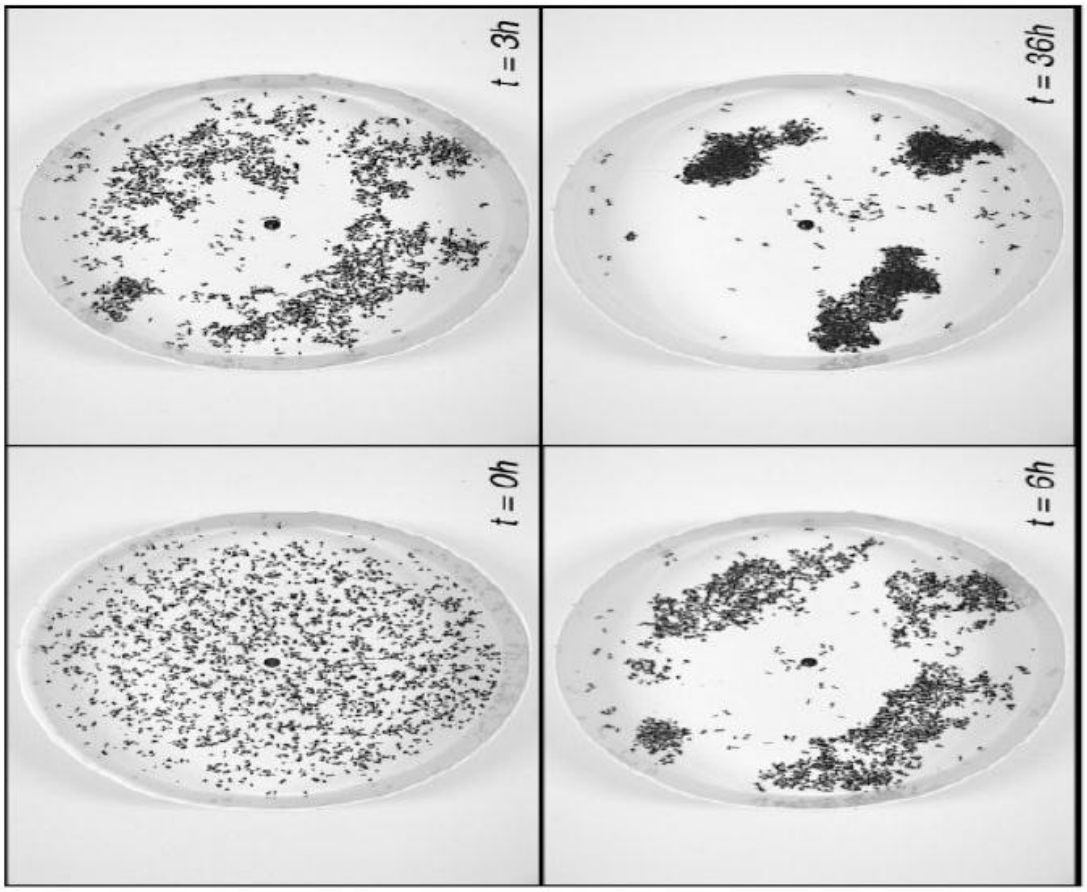

Figura 1. Formação de cemitérios de formigas da espécie *Pheidolle pallidula* (HARTMANN [8])

Formigas da espécie *Leptothorax unifasciatus* foram observadas por FRANKS e SENDOVA-FRANKS [5]. Como consequência deste estudo, verificou-se que estas formigas constroem seus ninhos em formato de anéis concêntricos. Sabe-se que os ovos e as pequenas larvas são agrupados no centro enquanto que as de tamanhos maiores vão sendo colocadas mais longe do mesmo, proporcionalmente ao tamanho da larva, ou seja, quanto maior, mais distante do centro, conforme se pode ver na Figura 2.

A forma como estes insetos interagem entre si em suas rotinas diárias têm despertado muito interesse em mineração de dados. Em especial, estudos de

simulação computacional cada vez mais buscam fazer uso de técnicas e métodos descobertos neste campo de pesquisa. De acordo com LAURO [9], pelo menos dois motivos existem para esse crescente interesse. Primeiramente, simular computacionalmente estes mecanismos de agrupamento promove um melhor entendimento da própria natureza. Além do mais, características como robustez e confiabilidade devido à redundância envolvida na forma como estes animais realizam atividades coletivas têm sido importantes fatores para a construção de diversos algoritmos, entre eles o SACA.

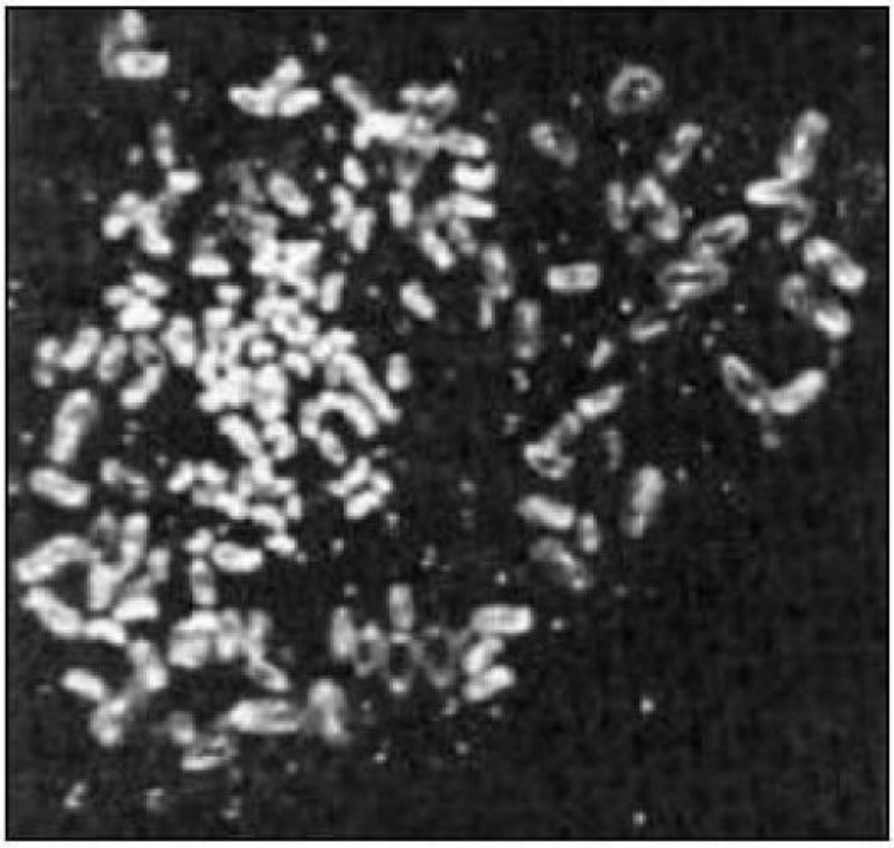

Figura 2. Ninhos de formigas da espécie *Leptothorax unifasciatus* (HARTMANN [8])

Observações em como formigas da espécie *Pheidolle pallidula* agrupavam outras formigas mortas da mesma colônia inspiraram DENEUBOURG et al. [3] a desenvolver um modelo de agrupamento baseado em grupo de agentes homogêneos. A essência deste modelo é de se ter formigas (agentes) que se movem aleatoriamente de modo que, ao encontrar um objeto, fará uma escolha de deixá-lo ou de pegá-lo e se mover para uma nova posição. Esta escolha é

fundamentada através de uma função probabilística de pegar ou de deixar o objeto. A probabilidade de deixar o objeto aumenta se a formiga estiver numa região onde a vizinhança de onde ela está tiver outros objetos semelhantes, por outro lado, a probabilidade diminui se a vizinhança contiver poucos objetos semelhantes. De maneira inversa, a probabilidade de pegar o objeto funciona. Em outras palavras, quanto mais densa a região, maior será a probabilidade de deixar e menor será a de pegar.

Alguns experimentos foram realizados inspirados neste modelo. BECKERS, HOLLAND e DENEUBOURG [1], por exemplo, realizaram testes de agrupamento utilizando robôs simples e sem memória, que simulavam o comportamento dos agentes.

Uma generalização do modelo proposto por DENEUBOURG et al. [3] foi desenvolvido por LUMER e FAIETA [11]. A principal diferença nesta proposta está em se utilizar uma função contínua para se avaliar o grau de similaridade entre os objetos do agrupamento.

2.3.1 Princípios Básicos do SACA

O algoritmo SACA foi apresentado à comunidade científica por LUMER e FAIETA [11]. Ele consiste basicamente em se ter agentes (formigas) movendo-se aleatoriamente em um plano (grade) através de um processo recursivo promovendo então o agrupamento dos objetos.

Inicialmente, os objetos são espalhados aleatoriamente na grade. Os agentes responsáveis pelo agrupamento também são posicionados de forma randômica. No decorrer do processo, cada agente irá mover-se de forma aleatória da célula atual para uma nova célula, desde que a mesma não esteja ocupada por algum objeto. Como princípio básico do SACA, cada célula deve conter apenas um único objeto e cada objeto deve estar em uma única célula. Deste modo, a grade terá algumas células que estarão vazias e outras que estarão preenchidas.

Durante a execução do algoritmo, cada agente irá mover-se pela grade em busca de objetos que estejam livres, ou seja, que não estejam sendo carregados por

outros agentes. Ao chegar em uma nova posição, o agente deverá pegar ou não aquele objeto mediante um processo de tomada de decisão de acordo com uma função probabilística (probabilidade de pegar). De igual maneira, um agente que esteja carregando um objeto para uma nova célula deverá deixar ou não o mesmo e sair à procura de outro (probabilidade de deixar). Portanto, o agrupamento proposto pelo SACA consiste basicamente de avaliação probabilística.

Como consequência da execução desta clusterização, os grupos formados tendem a formar regiões densas na grade. Os objetos que possuem características similares, portanto, idealmente formam o seu correspondente *cluster* distinguindo aquele grupo dos demais agrupamentos formados.

2.3.2 Funcionamento do SACA

A execução do algoritmo SACA está segmentada em duas partes:

- Fase inicial
  - Os objetos, que são representados pelos índices dos dados, são espalhados aleatoriamente na grade bidimensional. Cada objeto ocupa a posição de uma célula.
  - Cada agente, de forma randômica, seleciona um objeto.
  - O agente assume a posição na grade correspondente ao objeto que pegou.
- Fase de agrupamento
  - Um agente é selecionado de forma aleatória.
  - O agente selecionado move-se para uma nova célula que esteja vazia na grade.
  - O agente decide de acordo com uma função de probabilidade (probabilidade de deixar) se deixará o objeto na nova posição. Em caso positivo, o objeto é deixado e o agente deverá, de forma

aleatória, escolher outro objeto dentre aqueles que não estão sendo carregados por nenhum outro agente. A escolha de pegar este novo objeto é feita mediante uma avaliação de acordo também com uma função de probabilidade (probabilidade de pegar). Em caso negativo, ou seja, a decisão seja de não pegar o objeto, o agente deverá escolher outro objeto e fazer o processo de avaliação novamente. A execução age de forma recursiva até o agente pegar outro objeto.

- Retorna-se para o primeiro passo da Fase de agrupamento e repete-se a execução de acordo com um número de iterações previamente estabelecido.

2.3.3 Pseudocódigo do SACA

Em 2003, HANDL [7] mostrou o pseudocódigo para o SACA conforme ilustrado na Figura 3.

Algorithm 1 basic-ant

```
1:   begin
2:   INITIALISATION PHASE
3:   Randomly scatter data items on the toroidal grid
4:   for each j in 1 to #agents do
5:           i := randon_select (remaining items)
6:           pick_up (agent (j),i)
7:           g := randon_select (remaining_empty_grid_locations)
8:           place_agent (agent (j), g)
9:   end for
10:  MAIN LOOP
11:  for each it_ctr in 1 to #iterations do
12:          j := randon_select(all_agents)
13:          step (agent (j), stepsize)
14:          i := carried_item (agent(j))
15:          drop: = drop_item? (f*(i))
16:          if drop = TRUE then
17:                  while pick = FALSE do
18:                          i := randon_select(free_data_items)
19:                          pick: = pick_item? (f*(i))
20:                  end while
21:          end if
22:  end for
23:  end
```

Figura 3. Pseudocódigo do SACA (HANDL [7])

2.3.4 Funções e Parâmetros do SACA

Como já dito anteriormente, a decisão de deixar ou de pegar um objeto em determinada célula obedece a funções probabilísticas. As probabilidades podem ser calculadas através das funções de densidade descritas por DENEUBOURG et al. [3]. A equação 2.2 mostra a probabilidade de pegar enquanto que a equação 2.3 retrata a probabilidade de deixar.

$$p_p(i) = \left(\frac{k_p}{k_p + f(i)} \right)^2 \tag{2.2}$$

$$p_d(i) = \left(\frac{f(i)}{k_d + f(i)} \right)^2 \tag{2.3}$$

Os parâmetros k_p e k_d demonstram a influência que a função densidade[7], $f(i)$, exerce no cálculo da probabilidade de um agente respectivamente, pegar ou deixar um objeto em determinada posição. A literatura cita normalmente estes valores como sendo, respectivamente, 0.1 e 0.3. Através da Figura 4, vê-se o comportamento da probabilidade de pegar um objeto em função de $f(i)$. É fácil identificar na imagem que à medida que a densidade cresce de valor, a probabilidade diminui, ou seja, é mais difícil pegar o respectivo objeto. Por outro lado, na Figura 5, que mostra a probabilidade de deixar um objeto em função de $f(i)$, o comportamento da curva atua de forma inversa ao descrito na Figura 4. Neste caso, para maiores valores da função densidade, a probabilidade de deixar o objeto aumenta.

A função densidade, $f(i)$, foi definida por LUMER e FAIETA [11] como se vê na equação 2.4:

[7] Função densidade: É responsável por quantificar o nível de proximidade de um determinado objeto em relação aos demais objetos que estão na sua vizinhança.

$$f(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j \left(1 - \frac{d(i,j)}{\alpha}\right) \leftarrow f(i) > 0 \\ 0 \leftarrow c.c. \end{cases} \quad (2.4)$$

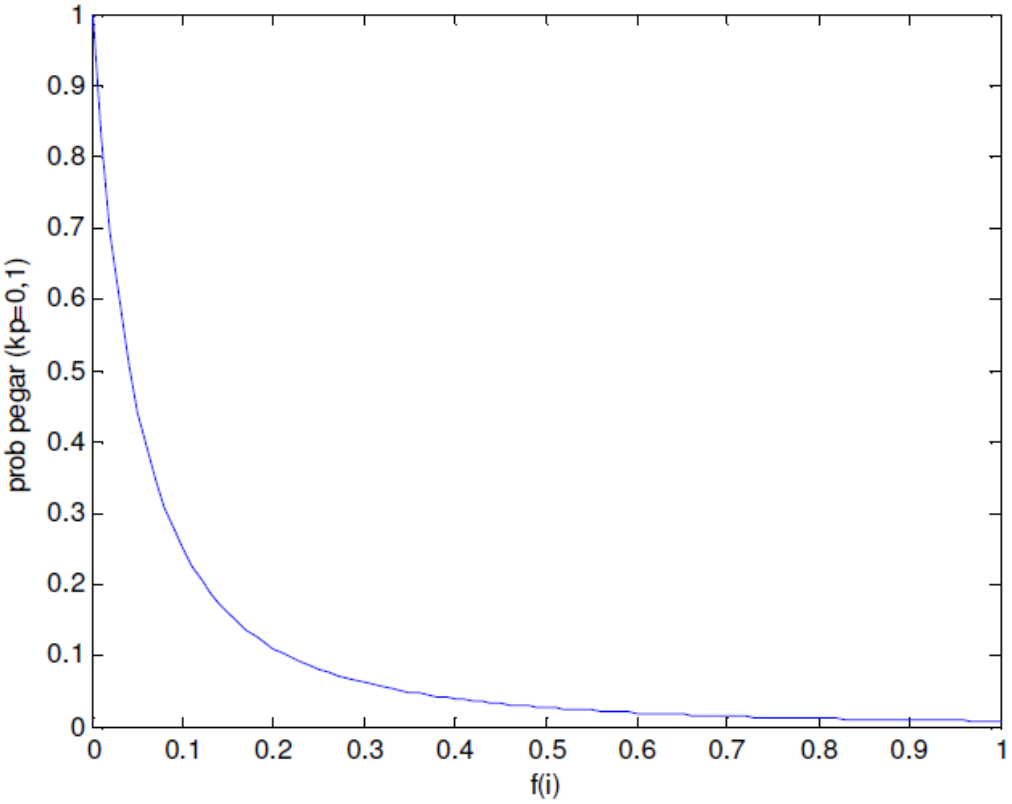

Figura 4. Probabilidade de pegar um objeto em função de $f(i)$ (LAURO [9])

O termo $d(i,j)$ representa a medida de dissimilaridade entre dois objetos. A forma de cálculo desta medida é descrita pela equação 2.1. O parâmetro α é um escalar que reflete a dependência entre os dados ($\alpha \in [0,1]$). Um valor muito baixo do mesmo pode dificultar a formação de agrupamentos enquanto que valores excessivamente altos podem unir grupos que são distintos.

O item σ^2 corresponde ao tamanho da vizinhança da célula cujo agente encontra-se. Frequentemente, $\sigma^2 \in [9,49]$. A Figura 6 ilustra esta vizinhança.

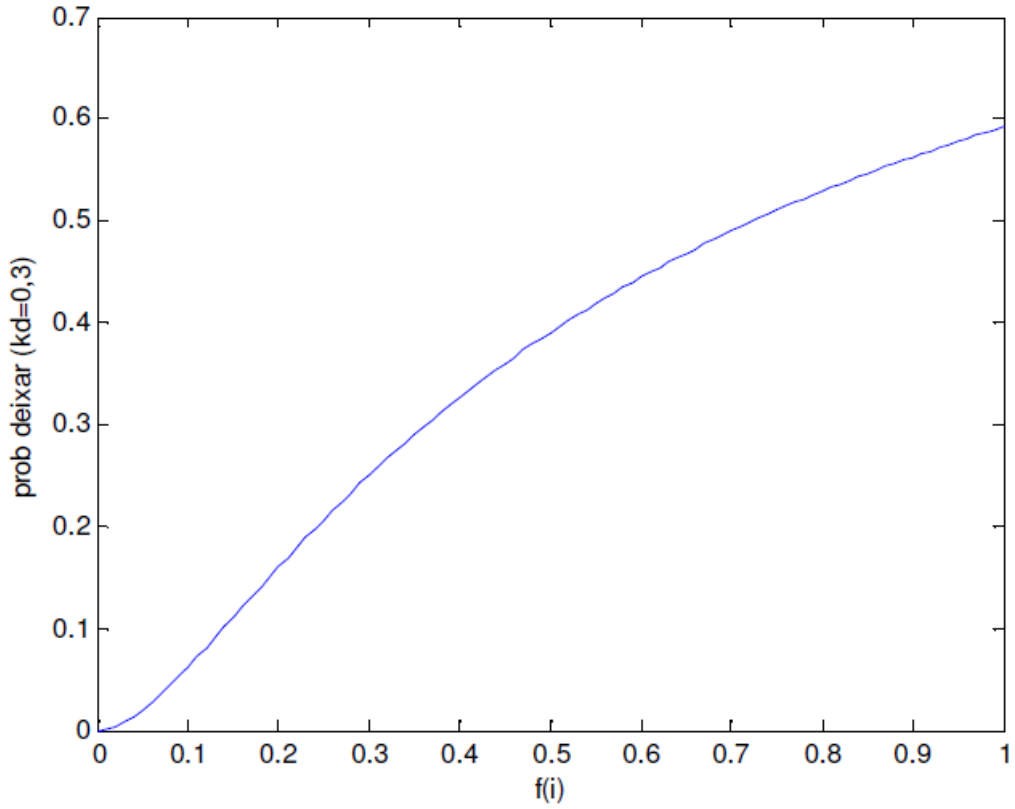

Figura 5. Probabilidade de deixar um objeto em função de $f(i)$ (LAURO [9])

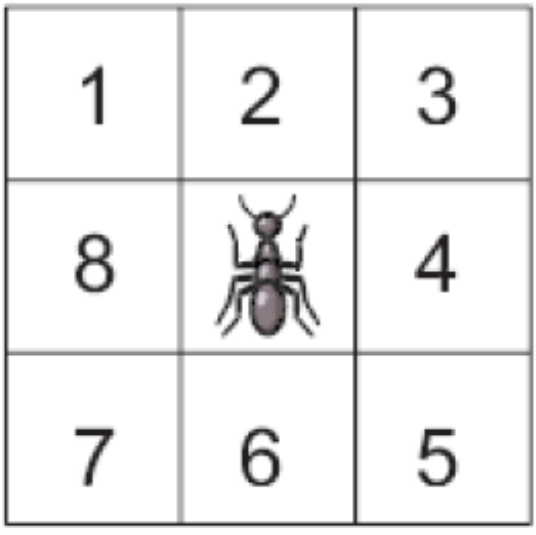

Figura 6. Representação da vizinhança de um agente para $\sigma = 3$ (HARTMANN [8])

2.4 Principais Modificações Propostas e Parametrizações para o Algoritmo de Agrupamento Simples por Colônia de Formigas

Desde que a versão inicial do SACA foi apresentado para o meio científico muitas alterações tem sido propostas. MONMARCHÉ [12] faz uso do algoritmo *k-means* após o agrupamento simples para corrigir os erros de agrupamento e agrupar os objetos que ficaram sem classificação. HARTMANN [8] mostrou o uso de uma rede neural artificial para decidir o comportamento do agente, além de um algoritmo genético para decidir os pesos desta rede. Por sua vez, VIZINE et al. [14] propôs um algoritmo adaptativo que é mais robusto em termos de geração do número de *clusters*.

Em relação ao algoritmo original, HANDL [7] propôs algumas melhorias visando resolver dois principais problemas. Primeiramente, a versão inicial do SACA não gera os agrupamentos propriamente e sim uma distribuição espacial dos objetos na grade. Em segundo lugar, a dificuldade que se tinha em ajustar os parâmetros do algoritmo ao se modificar a natureza dos dados envolvidos. Diante disto, tem-se a seguir as principais propostas de alterações.

2.4.1 Modificação da Função Densidade

A função densidade $f(i)$ é substituída por uma função $f^*(i)$, descrita na equação 2.5:

$$f^*(i) = \begin{cases} \dfrac{1}{\sigma^2}\sum_j \left(1 - \dfrac{d(i,j)}{\alpha}\right) \leftarrow f^*(i) > 0 \wedge \forall j \left(1 - \dfrac{d(i,j)}{\alpha}\right) \\ 0 \leftarrow c.c. \end{cases} \tag{2.5}$$

A principal diferença entre $f^*(i)$ e $f(i)$ é a condição $\forall j\left(1-\frac{d(i,j)}{\alpha}\right)$ na primeira parte da função. Esta restrição a mais impõe penalização para objetos que são dissimilares.

2.4.2 Memória dos Agentes

A ideia de cada agente possuir uma memória para armazenar as informações das posições em que os objetos anteriores tinham sido deixados já havia sido abordada por LUMER e FAIETA [11]. Quando um agente pega um novo objeto, a posição do objeto anterior com menor dissimilaridade é utilizado como referência para orientar o movimento randômico para a nova posição.

2.4.3 Raio de Percepção Crescente

Uma informação que é muito importante para determinar a qualidade do agrupamento formado é o tamanho da vizinhança percebido pelos agentes. Idealmente, quanto maior for esta vizinhança, melhor será então o agrupamento formado. Entretanto, ao se usar esta opção, além do elevado custo computacional associado, tem-se também uma dificuldade na formação dos *clusters* na parte inicial do processo de agrupamento.

A proposta de melhoria consiste então em se utilizar um tamanho de vizinhança variável ao longo das iterações. Portanto, nas primeiras execuções deve-se fazer uso de um valor baixo ($\sigma = 3$) para se evitar a dificuldade de formação de grupos iniciais além de prevenir a criação de grupos indesejáveis e, na etapa complementar, deve-se elevar o valor ($\sigma = 11$) do mesmo.

2.4.4 Separação Espacial

Um dos problemas detectados no algoritmo SACA diz respeito à formação de grupos muitos próximos uns dos outros. Isto pode acontecer devido ao fato que, uma vez os agrupamentos iniciais formados, os grupos tendem a se movimentar pouco na grade espacial.

Para contornar este tipo de situação, HANDL [7] sugeriu a substituição do parâmetro $\frac{1}{\sigma^2}$ da função densidade $f(i)$ por $\frac{1}{N_{OCC}}$, nas primeiras iterações do agrupamento, onde N_{OCC} é o número real de células ocupadas na vizinhança. Esta modificação implica no uso da similaridade, e não densidade, na formação dos grupos. Após determinada iteração pré-estabelecida, a função $f(i)$ volta a sua forma original.

2.4.5 Parâmetro α

O grau de dissimilaridade entre os objetos é um fator importante para a escolha do valor de α. A medição de sucessos e insucessos obtidos pelos agentes ao tentar deixar os objetos em alguma posição pode servir como métrica para determinar o valor deste parâmetro. A ideia desta melhoria no SACA é descrita da seguinte forma:

- O valor de α inicial é gerado de forma aleatória ($\alpha \in [0,1]$);
- O parâmetro tem seu valor atualizado a cada $N_{efetivos}$ iterações;
- Durante estas $N_{efetivos}$ iterações, são registrados vários insucessos de deixar o objeto acontecer (N_{falhas});
- A razão de falhas é dada por $r_{falhas} = \frac{N_{falhas}}{N_{efetivos}}$, onde o valor $N_{efetivos}$ é 100. Este valor foi adotado de modo a permitir a comparação de r_{falhas} com o valor percentual descrito na condição da equação 2.6;
- A regra de atualização de α é dada pela equação 2.6.

$$\alpha = \begin{cases} \alpha + 0.01 \leftarrow r_{falhas} > 0.99 \\ \alpha - 0.01 \leftarrow r_{falhas} \leq 0.99 \end{cases} \quad (2.6)$$

2.4.6 Tamanho da Grade

Para um conjunto de n objetos, a dimensão espacial da grade quadrada onde estes objetos e os agentes responsáveis pelo agrupamento deverão mover-se livremente é determinada pela relação $\sqrt{10n} \times \sqrt{10n}$.

2.4.7 Quantidade de Iterações

O quantitativo de iterações cresce proporcionalmente ao número de objetos da base de dados. Neste caso, a relação proposta é descrita na equação 2.7.

$$N_{iteracoes} = 2000 \times n \tag{2.7}$$

Outras modificações no algoritmo SACA original podem ser obtidas em HANDL [7].

Capítulo 3

Metodologia

Este capítulo descreve como o processo de desenvolvimento deste projeto está estruturado. A seção 3.1 é responsável por mostrar as bases de dados que foram utilizadas nos experimentos, enquanto a seção 3.2 demonstra a parte de implementação do algoritmo SACA e algumas variações. Por fim, a última seção apresenta o planejamento dos experimentos.

3.1 Bases de Dados

Como, entre outros, o objetivo deste trabalho era classificar os agrupamentos formados, então, foi necessário à escolha de bases de dados que pudessem permitir este tipo de cenário. Neste caso, as bases de dados utilizadas como referências foram a base de dados Iris e a base de dados *Wine*, ambas obtidas na UCI (*University of California, Irvine*) *Machine Learning Repository* (UCI [13]).

Devido à métrica que foi utilizada (distância Euclidiana) na geração dos *clusters* no processo de agrupamento, foi preciso que as bases escolhidas possuíssem seus atributos do tipo numérico, o que já acontece com as bases citadas. Outro ponto importante a ser considerado é que os dados (atributos) das instâncias (objetos) envolvidos em cada uma destas bases de dados precisaram ser normalizados para o correto cálculo da Matriz de Dissimilaridade. O processo de normalização foi construído dividindo-se o valor em questão pelo maior valor obtido na base de dados para aquele mesmo atributo.

Iris corresponde a uma classe de plantas com flor que na base de dados selecionada, divide-se em três possíveis classificações (*Setosa, Versicolour, Virginica*). Esta base foi disponibilizada na UCI por R.A. Fisher em 1988 e tem sido amplamente utilizada pela comunidade científica. Uma característica importante que se extrai deste repositório é que, ao se organizar os agrupamentos, uma das classes

(*Setosa*) distingue-se bem das demais. Os atributos na base correspondem a características da flor deste tipo de planta, sendo os seguintes:

- Comprimento da sépala;
- Largura da sépala;
- Comprimento da pétala;
- Largura da pétala.

Estes atributos estão em unidades de centímetros (cm). O número de objetos (itens) a serem agrupados é de 150. A Tabela 1 contém uma amostra (com dados normalizados) desta base de dados.

Tabela 1. Amostra normalizada da base de dados Iris

Comprimento da sépala (cm)	**Largura da sépala (cm)**	**Comprimento da pétala (cm)**	**Largura da pétala (cm)**	**Classe**
0.64	0.79	0.20	0.08	Setosa
0.82	0.63	0.66	0.60	Versicolour
0.82	0.72	0.73	0.80	Virginica

A outra base de dados selecionada para uso nos experimentos deste projeto foi a base *Wine*, disponibilizada na UCI por Riccardo Leardi em 1991. Este repositório contempla informações de 178 instâncias com 13 atributos cada uma, relativas a características de três diferentes tipos de vinhos cultivados em uma mesma região na Itália. Os atributos envolvem informações técnicas e específicas de características químicas e visuais de cada tipo de vinho, motivo pelo qual não foram detalhados o nome destes campos. A Tabela 2 contém uma amostra (com dados normalizados) desta base de dados, onde Ai para i=1,2,...,13 corresponde a cada um dos atributos e C é a classificação final do objeto.

Tabela 2. Amostra normalizada da base de dados *Wine*

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	C
0.95	0.29	0.75	0.52	1	0.72	0.60	0.42	0.63	0.43	0.60	0.98	1	1
0.82	0.27	0.68	0.68	1	0.28	0.20	0.56	0.40	0.23	0.52	0.45	1	2
0.90	0.79	0.88	0.83	1	0.51	0.18	0.40	0.31	0.65	0.39	0.48	0	3

3.2 Implementação

O pseudocódigo do SACA, proposto por HANDL [7] e descrito na seção 2.3.3, bem como algumas variações do mesmo foram implementados no software Matlab 7.12.0.635. A Figura 7 mostra o fluxograma do algoritmo SACA. A seguir, serão vistos detalhes da implementação.

3.2.1 Descrição da Implementação

A implementação da representação espacial da grade onde aconteceria o processo de agrupamento aconteceu através de uma matriz no Matlab. Os índices de posição desta matriz representam justamente as células responsáveis por armazenar os objetos durante a clusterização. Através desta correspondência, é possível obter a posição exata que determinado objeto se encontra ao longo da execução do código. Uma posição nula nesta matriz indica a ausência de objeto naquele lugar.

Durante a execução, os objetos podem mover-se livremente pelas células da matriz desde que a nova posição esteja vazia. Cada célula pode ter no máximo um objeto. A representação dos objetos no código acontece de forma numérica e depende da base de dados envolvida no processo.

Uma segunda matriz foi criada com as mesmas dimensões da matriz inicial que representa a grade. A ideia é de se utilizar esta outra matriz para ter a

informação sobre a posição que cada um dos agentes se encontra na grade. Através disto, é possível identificar qual agente está segurando determinado objeto.

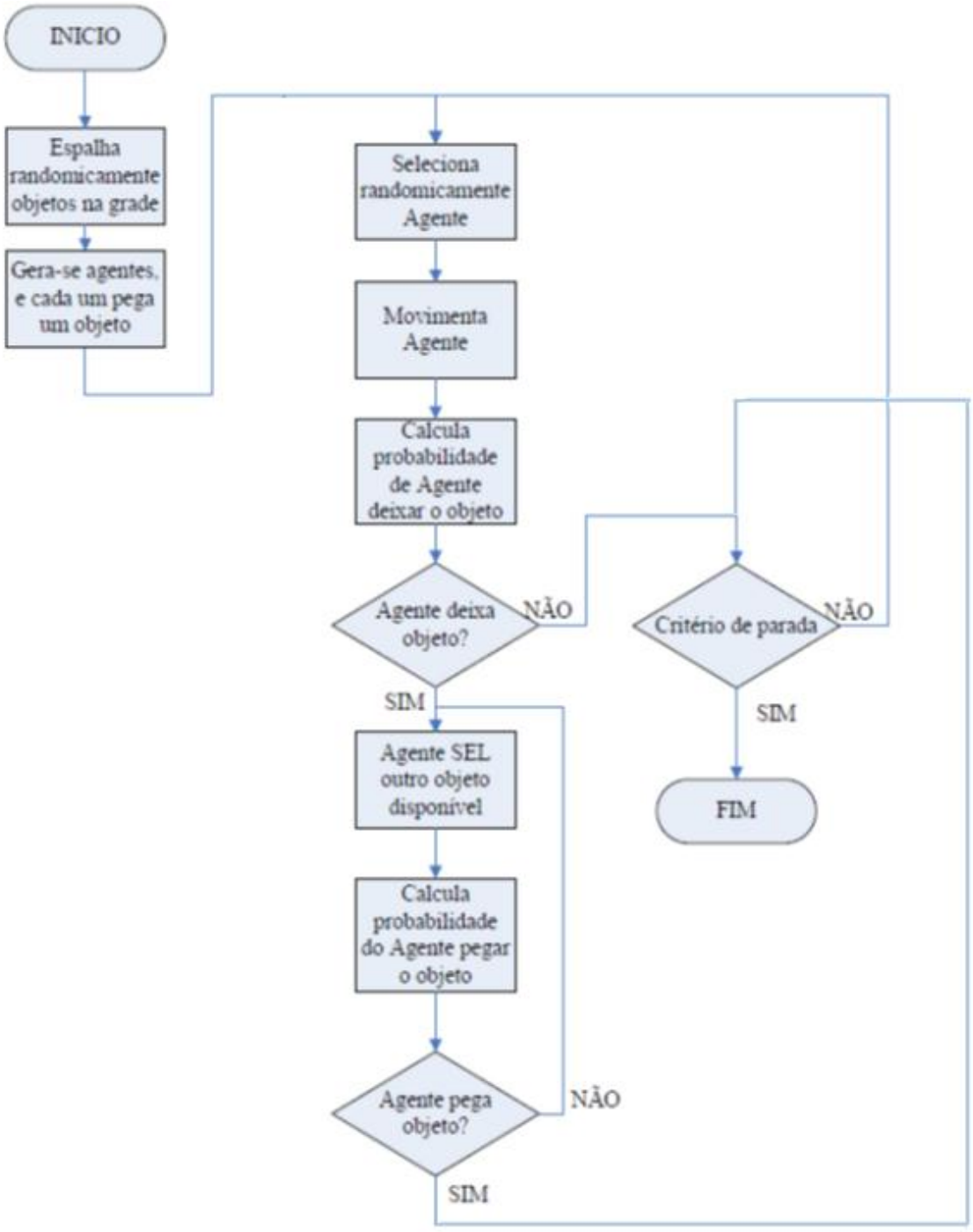

Figura 7. Fluxograma do SACA (LAURO [9])

À medida que o agrupamento é feito, os objetos tendem a juntar-se formando os *clusters* na grade. Para facilitar a identificação visual dos grupos distintos, os

objetos são representados através de cores e símbolos diferentes, cada qual de acordo com a sua respectiva classificação.

3.2.2 Cenários de Execução

Como foi visto na seção 1.2, este projeto teve por objetivo promover um estudo comparativo do algoritmo SACA e variações do mesmo, com a finalidade de melhoria na eficiência do processo de agrupamento. Na seção 2.4, pode-se ver algumas propostas de aperfeiçoamento do SACA em sua forma básica. No presente trabalho, além de se fazer uso da versão simples do SACA conforme a seção 2.3.3, foi construída também a solução com alteração no parâmetro α (ver seção 2.4.5) e uma nova configuração também foi proposta. As três parametrizações foram aplicadas nas duas bases de dados vistas na seção 3.1, conforme a seguir:

- SACA na versão original;
- SACA com melhoria no parâmetro α. Este parâmetro é um escalar que quantifica o quão denso os agrupamentos formados devem ser. Segundo experimentos relatados por LAURO [9], este tipo de melhoria apresentou bons resultados;
- SACA com medição de insucessos. Os dois primeiros cenários de implementação utilizam como critério de parada do algoritmo um número de iterações pré-estabelecido. Esta 3ª parametrização utiliza um critério de parada diferente: avalia a quantidade de insucessos que determinado agente teve ao sair em busca de um novo objeto que esteja livre. Conforme se pode ver no fluxograma representado pela Figura 7, quando a decisão de um agente de deixar o objeto na nova posição é positiva, o mesmo deverá sair à procura de um novo objeto que esteja livre. Ao encontrar, a decisão de pegar ou não o mesmo obedece a uma função probabilística (ver seção 2.3.2). Caso a decisão seja negativa, ou seja, o agente não pegue o objeto, ele sairá em busca de outro e então uma nova avaliação é feita. O processo repete-se até que o agente consiga pegar um objeto. A cada insucesso seguido do agente, o valor é contabilizado. Se esse valor exceder certo

limiar, a execução do código é finalizada, independentemente do número de iterações. Este tipo de cenário evita que a rotina fique presa a uma recursão infinita diante de possíveis insucessos seguidos por parte de um agente em pegar um novo objeto.

No decorrer deste projeto, outros melhorias no SACA como a alteração na função densidade descrita na seção 2.4.1 e a separação espacial vista na seção 2.4.4 foram desenvolvidas inicialmente, entretanto, não foram utilizadas como objeto de estudo comparativo do presente trabalho.

3.3 Planejamento do Experimento

Neste projeto, os experimentos envolvidos consistiram em comparar-se a taxa de acerto do algoritmo SACA em diferentes parametrizações relatadas na seção 3.2.2, fazendo-se uso das bases de dados descritas na seção 3.1, através da aplicação de testes estatísticos. A taxa de acerto considerada no presente trabalho é avaliada a partir da geração dos *clusters* e é representada pela equação 3.1.

$$taxadeacerto = \frac{N_{acertos}}{N_{objetos}} \tag{3.1}$$

O termo $N_{acertos}$ corresponde ao número de objetos que foram agrupados corretamente, enquanto que $N_{objetos}$ corresponde ao total de objetos da base de dados. Neste trabalho, o acerto é considerado se o objeto que está sendo avaliado foi agrupado no seu respectivo *cluster*. É considerado fazer parte de um determinado *cluster* objetos que estejam no máximo a distância de uma célula do mesmo, ou seja, objetos que sejam vizinhos ao agrupamento formado. Os objetos que não pertencem a nenhum agrupamento ou fazem parte de um grupo que não são os seus de origem são contabilizados como erro de classificação.

A partir dos objetivos deste projeto, assumiu-se a seguinte hipótese:

- Hipótese 1 (H1): O uso do algoritmo SACA com medição de insucessos do agente produz agrupamentos com melhor taxa de

acerto em relação ao SACA em sua forma básica e ao SACA com melhoria no parâmetro α.

3.3.1 Variáveis e Escalas

- Fator 1 (F1): algoritmo de agrupamento dos objetos na grade.
    - Alternativa 1 (F1-A1): agrupamento construído a partir do SACA na versão original (seção 2.3).
    - Alternativa 2 (F1-A2): agrupamento construído a partir do SACA com melhoria no parâmetro α (seção 2.4.5).
    - Alternativa 3 (F1-A3): agrupamento construído a partir do SACA com medição de insucessos (seção 3.2.2).
    - Os valores desta variável são medidos de acordo com uma escala nominal.
        - Valores possíveis: SACA em sua forma básica, SACA com melhoria no parâmetro α e SACA com medição de insucessos.
- Parâmetro 1 (P1): base de dados considerada.
    - Os valores desta variável são medidos de acordo com uma escala nominal.
        - Valores possíveis: Iris e *Wine*.
- Variável de Resposta 1 (R1): taxa de acerto dos agrupamentos formados.
    - Os valores desta variável são medidos de acordo com uma escala razão.
        - Valores possíveis: taxa de acerto dos agrupamentos formados, definido como a razão entre o número de

exemplos classificados corretamente e o número total de exemplos.

3.3.2 Hipóteses Nulas e Alternativas

A partir da definição das variáveis na seção 3.3.1, pode-se formalizar a hipótese H1 nas seguintes hipóteses nulas e alternativas:

- $H_{0\text{-}1,1}$: $\mu_{R1,F1-A3} - \mu_{R1,F1-A1} \leq 0$
    - A hipótese nula tem o seguinte significado: o valor médio de R1, quando for feito uso da alternativa F1-A3, será menor ou similar ao valor médio de R1, quando for feito uso da alternativa F1-A1. Em outras palavras, a diferença entre as duas médias será menor ou igual à zero.
- $H_{0\text{-}1,2}$: $\mu_{R1,F1-A3} - \mu_{R1,F1-A2} \leq 0$
    - A hipótese nula tem o seguinte significado: o valor médio de R1, quando for feito uso da alternativa F1-A3, será menor ou similar ao valor médio de R1, quando for feito uso da alternativa F1-A2. Em outras palavras, a diferença entre as duas médias será menor ou igual à zero.
- $H_{1\text{-}1,1}$: $\mu_{R1,F1-A3} - \mu_{R1,F1-A1} > 0$
    - A hipótese alternativa tem o seguinte significado: o valor médio de R1, quando for feito uso da alternativa F1-A3, será maior que o valor médio de R1, quando for feito uso da alternativa F1-A1. Em outras palavras, a diferença entre as duas médias será maior que zero.
- $H_{1\text{-}1,2}$: $\mu_{R1,F1-A3} - \mu_{R1,F1-A2} > 0$
    - A hipótese alternativa tem o seguinte significado: o valor médio de R1, quando for feito uso da alternativa F1-A3, será maior que o valor médio

de R1, quando for feito uso da alternativa F1-A2. Em outras palavras, a diferença entre as duas médias será maior que zero.

3.3.3 Instrumento

As unidades experimentais (bases de dados) utilizadas nos experimentos foram armazenadas em arquivos de texto de modo a serem posteriormente utilizadas no Matlab durante a fase de agrupamento. Os dados resultantes da execução foram armazenados em arquivos do Microsoft Excel 2010. Os parâmetros correspondentes de cada execução também foram salvos nestes arquivos. O algoritmo SACA e algumas variações foram implementados no software Matlab 7.12.0.635. Por fim, o software R na versão 2.15.2 foi utilizado na parte de testes estatísticos.

3.3.4 Metodologia de Análise

Os experimentos foram conduzidos através de dois grupos de projetos experimentais, um para cada base de dados. As alternativas descritas na seção 3.3.1 foram aplicadas nas duas bases (Iris e *Wine*), sendo 33 vezes em cada uma delas. A cada execução, a informação da taxa de acerto (equação 3.1) foi armazenada.

Antes da aplicação dos testes de hipóteses, foi necessária a aplicação da técnica de *boxplot* através do Matlab para identificar a existência de *outliers*[8] nos dados coletados. A presença de *outliers* pode comprometer a qualidade dos testes estatísticos, motivo pelo qual os mesmos devem ser retirados da base. Maiores informações sobre esta técnica podem ser obtidas em [10].

Após a filtragem descrita anteriormente, os dados ficaram disponíveis para aplicação dos testes estatísticos de hipóteses. No presente projeto, as escolhas foram dos testes t de *Student*, que tem como premissa ser aplicado em dados que representam uma distribuição normal, e o teste de *Wilcoxon*, que pode ser utilizado em dados que não necessariamente obedecem a uma distribuição normal. Neste

[8] Outliers: Medições que representam valores não esperados em uma métrica.

caso, visto que mais de 30 coletas de dados foram realizadas em cada experimento, pode-se supor a normalidade na distribuição gerada. Caso isto não seja suficiente, o teste de *Wilcoxon* pôde ser utilizado nestes experimentos. O nível de significância considerado foi de 5%. A aplicação destes testes aconteceu no software R. LEVINE, BERENSON e STEPHAN [10] explicam o funcionamento dos testes t de *Student* e *Wilcoxon*.

Capítulo 4

Testes e Resultados

O objetivo deste capítulo é relatar a aplicação da metodologia descrita no capítulo anterior e analisar os resultados obtidos bem como proceder com a execução dos testes estatísticos. As seções 4.1 e 4.2 mostram os dados coletados resultantes dos experimentos nas bases Iris e *Wine*, respectivamente, enquanto que a seção 4.3 descreve a aplicação dos testes estatísticos com análise dos resultados.

4.1 Experimentos na Base de Dados Iris

A primeira parte de execução dos experimentos deste projeto aconteceu na base de dados Iris, descrita anteriormente na seção 3.1. Os três cenários que representam os experimentos foram elencados na seção 3.2.2 e serão detalhados a seguir. Os dados coletados nestes experimentos estão disponíveis no Apêndice A.

4.1.1 Experimento I

Neste primeiro experimento, o algoritmo SACA foi utilizado em sua forma original (seção 2.3.3). A título de informação, o código construído no Matlab para este caso está disponível no Apêndice C. A Tabela 3 mostra a parametrização do SACA que foi utilizada durante as 33 execuções do código. Estes valores foram escolhidos seguindo a orientação tanto de outros trabalhos ora citados neste projeto, como de execuções exaustivas nesta base de dados, a fim de se obter o melhor cenário de agrupamento possível, visto que, como já dito anteriormente, a parametrização do algoritmo SACA depende muito da natureza da base de dados envolvida.

Tabela 3. Parâmetros do Experimento I

$N_{objetos}$	$N_{iteracoes}$	$N_{agentes}$	N_{grupos}	**dimensão da grade**	k_p	k_d	α	σ
150	300000	10	3	39	0.1	0.25	0.4	3

Em cada iteração, após a formação dos agrupamentos é possível visualizar que os objetos da classe *Setosa* formaram um *cluster* diferenciando-se das classes *Virginica* e *Versicolour*, o que já era esperado para esta base de dados. A Figura 8 ilustra um dos exemplos obtidos.

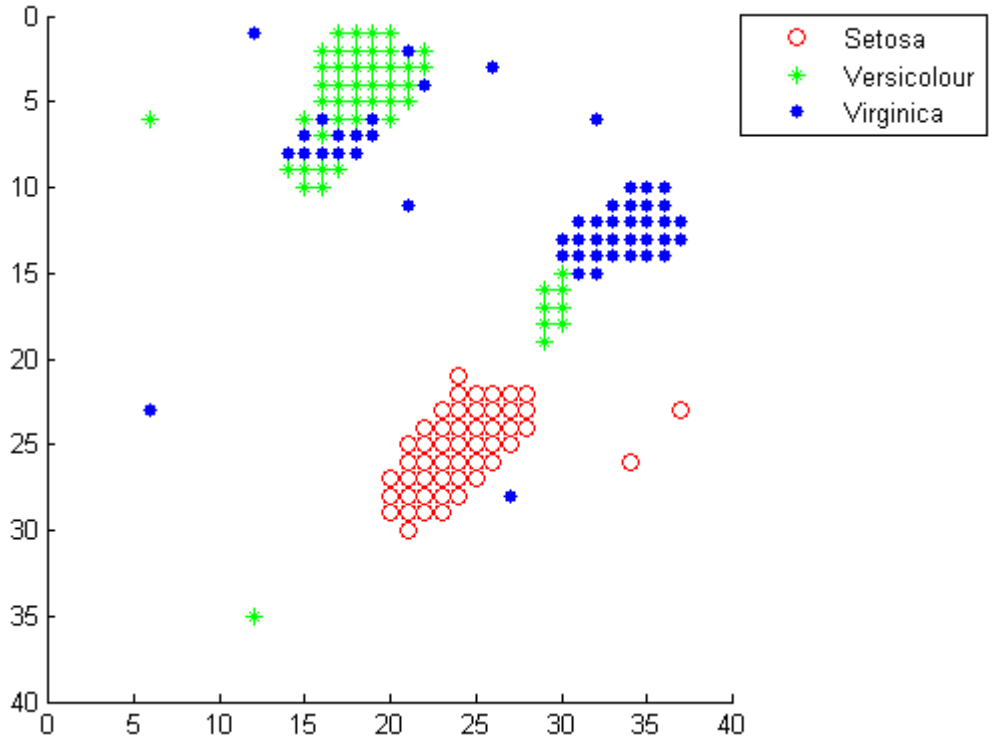

Figura 8. Clusterização do Experimento I

Após o cálculo da taxa de acerto (disponível no Apêndice A), pôde-se aplicar a técnica de *boxplot*. O gráfico gerado está representado na Figura 9. Pode-se visualizar uma cruz vermelha na parte superior da imagem indicando a presença de um *outlier* entre os dados disponíveis. Observa-se também que o valor da mediana representada pela linha vermelha coincide com a posição do 1º quartil, representado na figura pela linha inferior do retângulo gerado, mostrando com isso que a maioria dos valores da taxa de acerto concentrou-se na região entre 0.93 e 0.935. Isto sugere que a variância obtida para os valores da taxa de acerto neste experimento foi baixa.

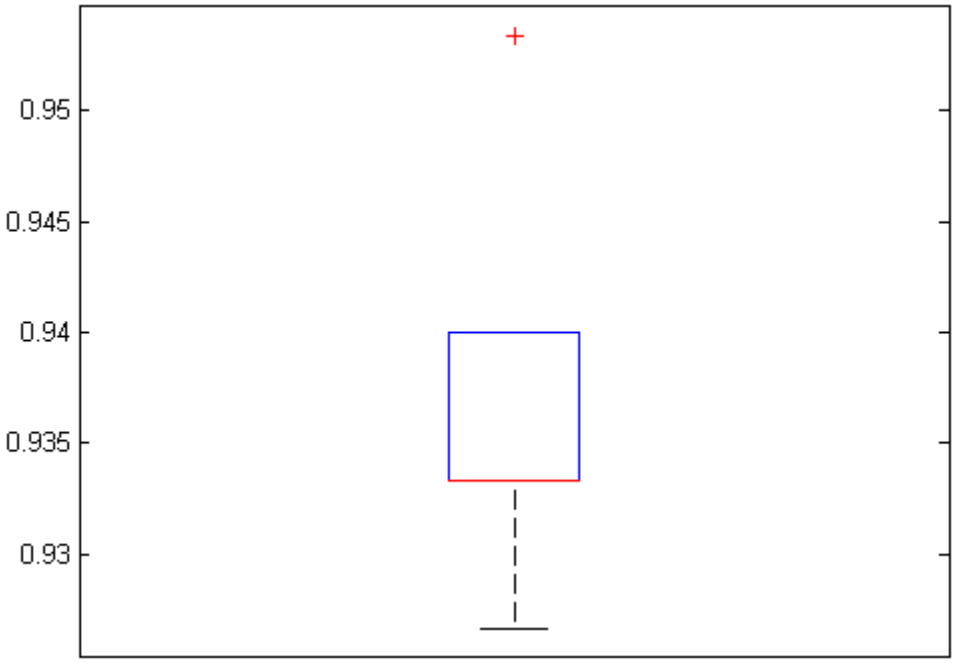

Figura 9. *Boxplot* da taxa de acerto do Experimento I

4.1.2 Experimento II

A parametrização neste segundo experimento apenas diferencia-se do primeiro com relação ao parâmetro α, porém, o critério de parada do algoritmo permanece o mesmo (número de iterações). Neste caso, α sofreu ajustes no decorrer de cada execução de acordo com a melhoria proposta na seção 2.4.5. A Tabela 4 detalha a configuração dos parâmetros.

Tabela 4. Parâmetros do Experimento II

$N_{objetos}$	$N_{iteracoes}$	$N_{agentes}$	N_{grupos}	**dimensão da grade**	k_p	k_d	α	σ
150	300000	10	3	39	0.1	0.25	ajustável no tempo	3

A Figura 10 exemplifica a formação dos agrupamentos para este cenário. Do ponto de vista gráfico, não houve mudanças significativas dos agrupamentos formados comparando com o gráfico do Experimento I. A média do valor de α ao longo das 33 iterações foi 0.44, valor muito próximo do experimento anterior. O

boxplot obtido está representado na Figura 11. A mediana coincidiu com o 3º quartil, representado na imagem pela linha na parte superior do retângulo. O 3º quartil corresponde ao valor atingido ao ser ordenar os 75% primeiros elementos da taxa de acerto, obtidos neste experimento.

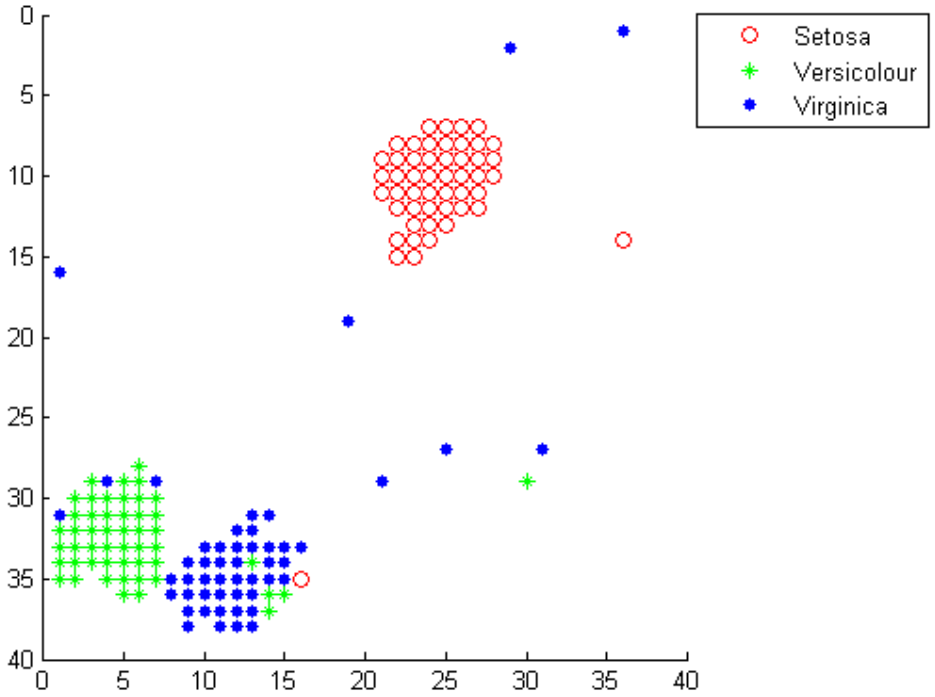

Figura 10. Clusterização do Experimento II

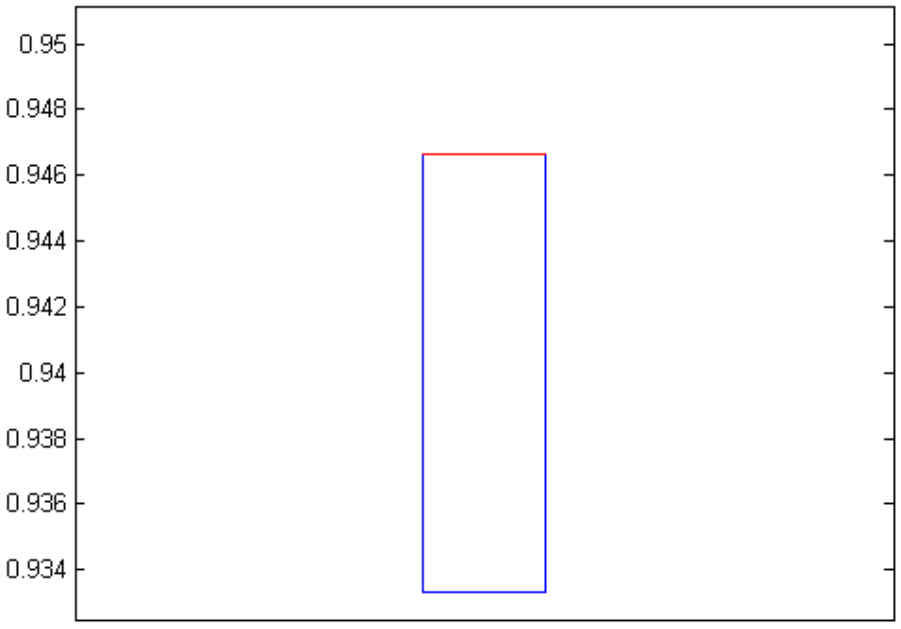

Figura 11. *Boxplot* da taxa de acerto do Experimento II

4.1.3 Experimento III

O principal motivo da construção deste terceiro cenário experimental era de ter-se um código que fizesse uso de um diferente critério de parada que não o número de iterações. Analisando o pseudocódigo do SACA na seção 2.3.3, pode-se observar que ao agente que acabou de deixar um objeto para sair à procura de um novo objeto, enquanto ele não pegá-lo, ficará nesta busca recursiva de modo que o programa poderá entrar em *loop* infinito. Este problema se torna mais passível de acontecer à medida que os *clusters* começam a ganhar forma e, consequentemente, se torna mais difícil pegar um novo objeto.

O cenário de execução deste Experimento III é o SACA com medição de insucessos, descrito na seção 3.2.2. A Tabela 5 mostra a parametrização que foi utilizada neste caso. O único parâmetro novo é o $Qtd_{insucessos}$ que será utilizado como critério de parada na execução. Para a base de dados Iris, o valor 60 para este parâmetro foi obtido através de execuções exaustivas em busca de formações na grade que promovessem a geração dos grupos. Valores acima e abaixo deste foram testados, porém, não foram satisfatórios. A Figura 12 descreve os *clusters* gerados.

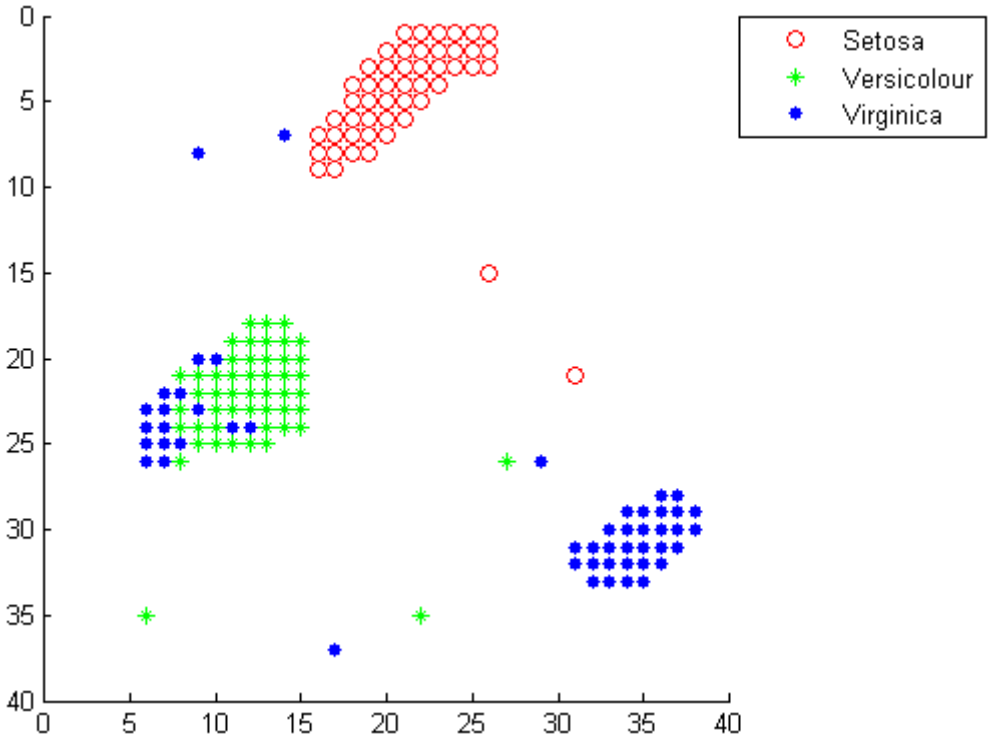

Figura 12. Clusterização do Experimento III

Tabela 5. Parâmetros do Experimento III

$N_{objetos}$	$Qtd_{insucessos}$	$N_{agentes}$	N_{grupos}	**dimensão da grade**	k_p	k_d	α	σ
150	60	10	3	39	0.1	0.25	0.4	3

Mais uma vez, pode-se ver que o gráfico de agrupamento obtido no Experimento III não oferece diferenças relevantes em relação aos gráficos de agrupamento dos Experimentos I e II. Na seção 4.3 será vista a aplicação dos testes estatísticos e então, será possível saber se de fato o algoritmo utilizado no Experimento III produz uma taxa de acerto melhor que os outros dois experimentos.

Entretanto, uma informação importante obtida como resultado da execução deste último experimento é que a média das iterações obtidas nas 33 execuções foi de aproximadamente 183000, valor pouco maior que a metade do número de iterações utilizado nos Experimentos I e II (300000). A Figura 13 é o resultado da aplicação de técnica de *boxplot* neste caso. Pode-se ver a existência de dois *outliers* representados pela cruz vermelha no gráfico. Assim como no Experimento II, no gráfico de *boxplot* do Experimento III pode-se ver que a mediana (linha de cor vermelha) coincide com o 3º quartil.

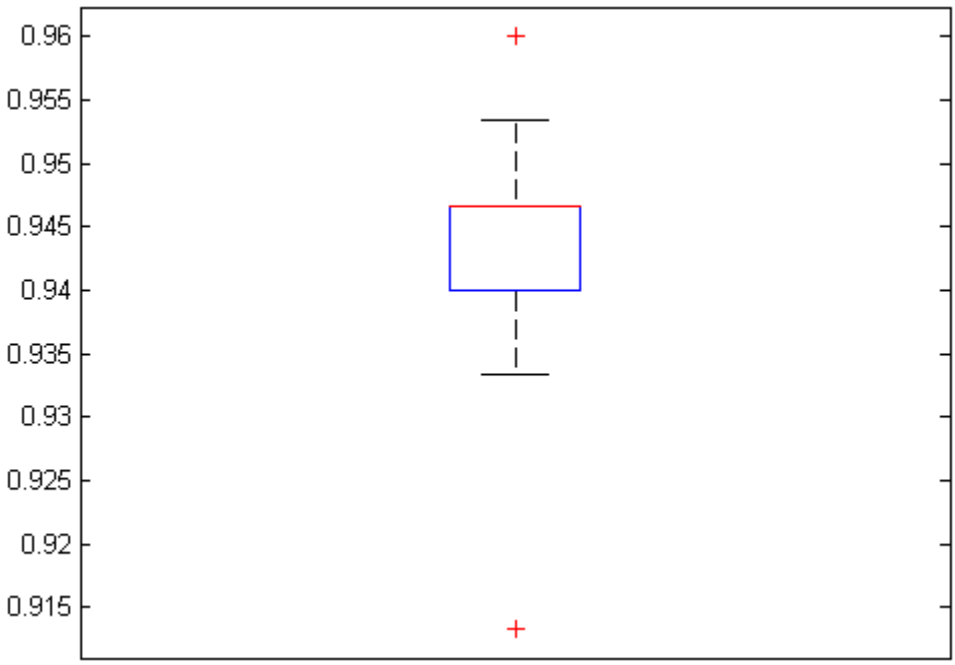

Figura 13. *Boxplot* da taxa de acerto do Experimento III

4.2 Experimentos na Base de Dados *Wine*

Esta parte do projeto contempla os experimentos que foram aplicados na base de dados *Wine*, descrita na seção 3.1. Assim como na seção 4.1, os cenários utilizados para a construção dos três experimentos são os que estão detalhados na seção 3.2.2. Os dados coletados são mostrados no Apêndice B.

4.2.1 Experimento IV

O algoritmo SACA em sua forma original foi utilizado neste caso. A Tabela 6 mostra os parâmetros que foram utilizados. Diferentemente do que aconteceu nos experimentos da base de dados Iris, o número de iterações necessário nos experimentos da base *Wine* foi de 1000000, possivelmente, em decorrência do fato de neste caso existir mais objetos a serem agrupados. Os parâmetros k_p, k_d e α também foram alterados sendo os valores mostrados os que proporcionaram o melhor resultado na formação dos grupos.

Tabela 6. Parâmetros do Experimento IV

$N_{objetos}$	$N_{iteracoes}$	$N_{agentes}$	N_{grupos}	**dimensão da grade**	k_p	k_d	α	σ
178	1000000	10	3	42	0.07	0.1	0.9	3

Um exemplo dos agrupamentos formados pode ser visto através da Figura 14. Através da imagem, é possível ver que nenhum dos *clusters* gerados distingue-se bem dos demais, como acontece com os agrupamentos formados para a base de dados Iris. Observa-se também que os objetos da Classe 2 são os que mais ficaram suscetíveis a ficarem dispersos, muito embora essa seja a maior classe.

Com relação ao cálculo da taxa de acerto, o gráfico obtido através da técnica de *boxplot* pelo Matlab está representado na Figura 15. Não foram detectados *outliers*. Pode-se ver também que a mediana ficou bem próxima da região central entre os 1º e 3º quartis, diferentemente do que foi obtido nos Experimentos I, II e III.

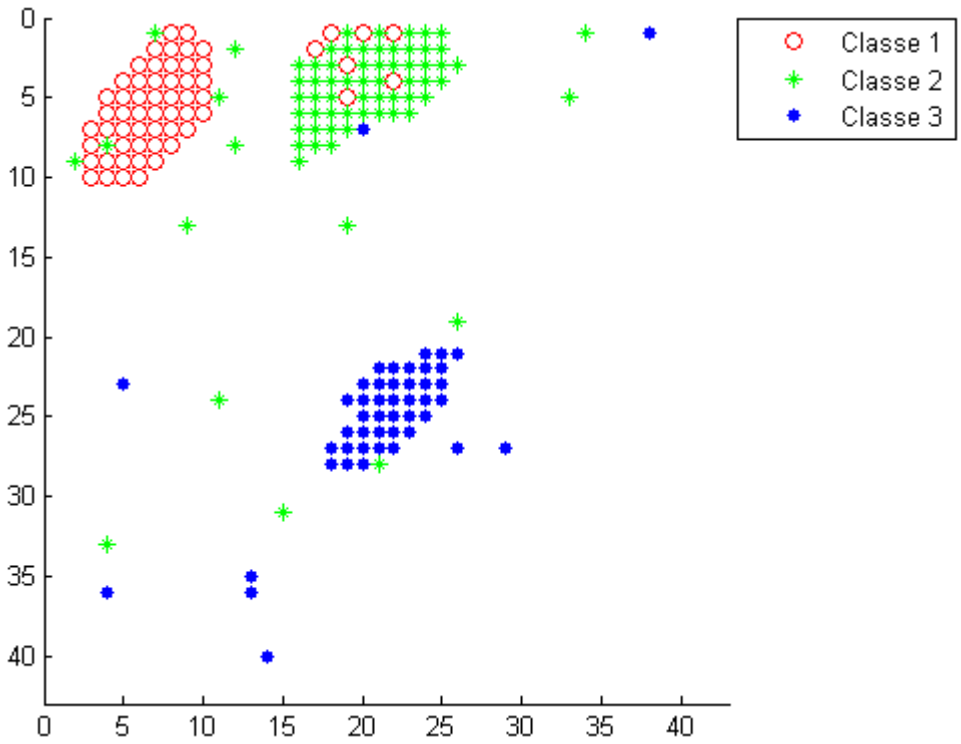

Figura 14. Clusterização do Experimento IV

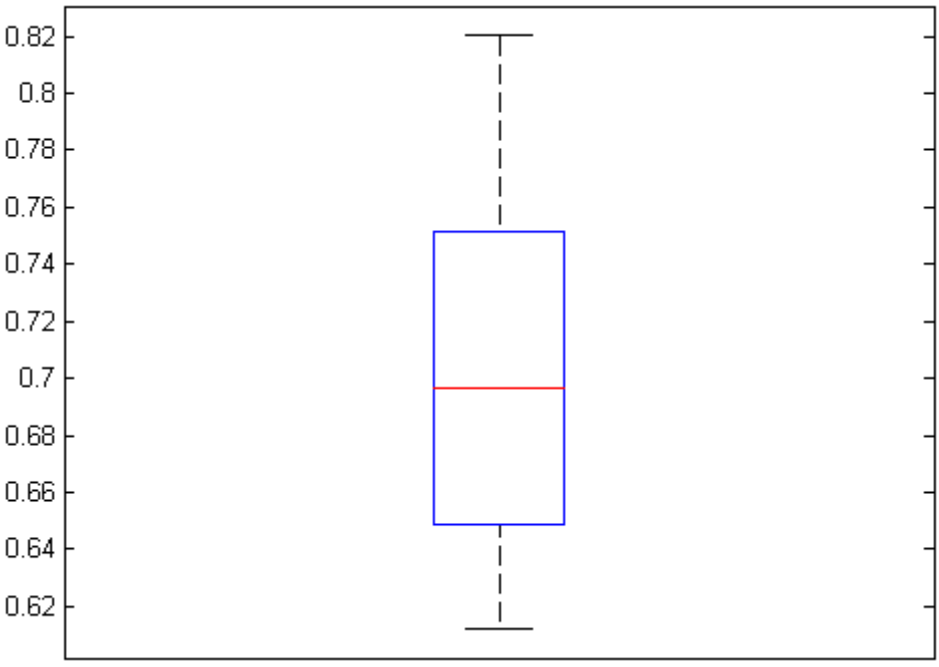

Figura 15. *Boxplot* da taxa de acerto do Experimento IV

4.2.2 Experimento V

Neste segundo experimento envolvendo a base de dados *Wine* se utilizou o SACA com melhoria no parâmetro α. A Tabela 7 mostra a parametrização utilizada neste caso.

Tabela 7. Parâmetros do Experimento V

$N_{objetos}$	$N_{iteracoes}$	$N_{agentes}$	N_{grupos}	**dimensão da grade**	k_p	k_d	α	σ
178	1000000	10	3	42	0.07	0.1	ajustável no tempo	3

Ao aplicar-se o código várias vezes utilizando-se esta modificação em α, os resultados obtidos no processo de agrupamento não foram bons. Os *clusters* não puderam ser gerados na grade. A Figura 16 exemplifica um dos resultados.

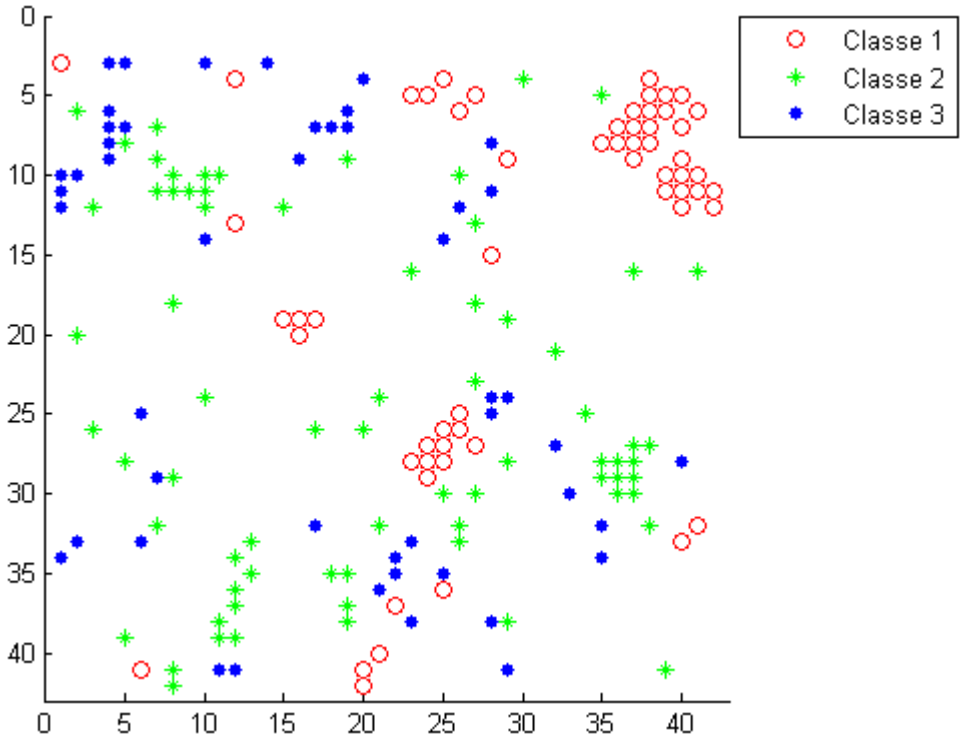

Figura 16. Clusterização do Experimento V

A variação do parâmetro α ao longo da execução da rotina não mostrou ser uma boa solução no agrupamento dos dados desta base. Isto mostra certo grau de

sensibilidade da base *Wine* com relação a mudanças deste parâmetro. Após algumas execuções, pôde-se observar que α tendeu para valores próximos de 0.5, diferente do α utilizado no Experimento IV (0.9).

Como os grupos não puderam ser formados, consequentemente, não foi possível calcular a taxa de acerto para este experimento, motivo pelo qual os testes estatísticos e a técnica de *boxplot* não se aplicam neste caso.

4.2.3 Experimento VI

O último experimento fez uso do SACA com medição de insucessos (ver seção 3.2.2), cujos parâmetros utilizados são mostrados na Tabela 8. A $Qtd_{insucessos}$ foi ajustada para o valor 59 após uma série de testes em busca do melhor cenário de agrupamento para esta base de dados. A Figura 17 exemplifica um dos agrupamentos que foram gerados. Observa-se que o comportamento de agrupamento obtido permaneceu praticamente o mesmo em relação ao gráfico da Figura 14.

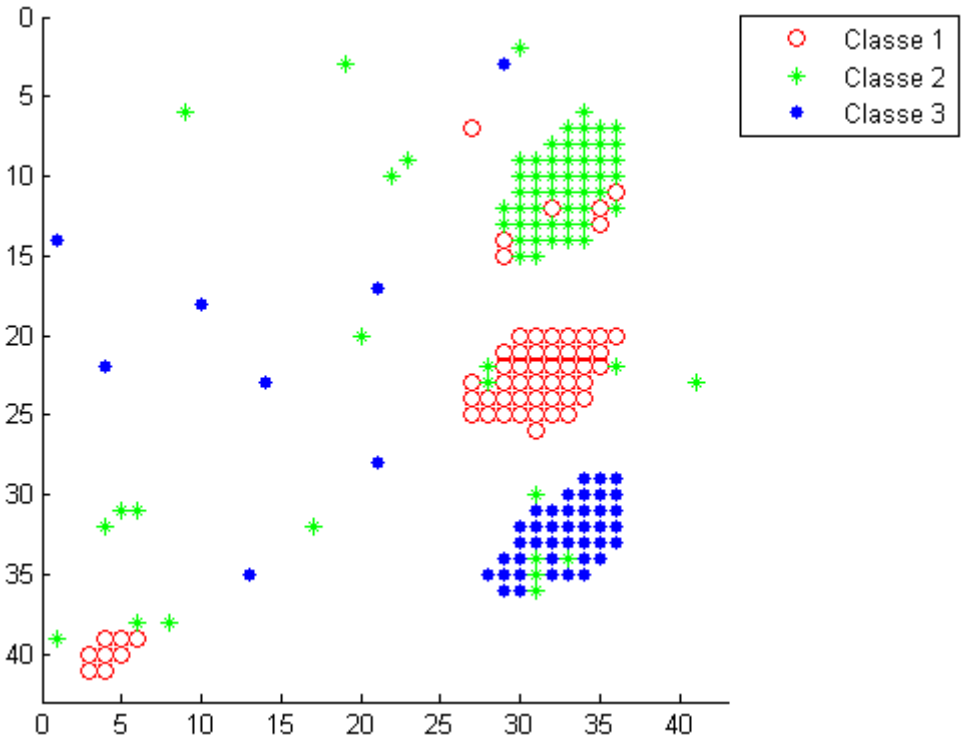

Figura 17. Clusterização do Experimento VI

Tabela 8. Parâmetros do Experimento VI

$N_{objetos}$	$Qtd_{insucessos}$	$N_{agentes}$	N_{grupos}	**dimensão da grade**	k_p	k_d	α	σ
178	59	10	3	42	0.07	0.1	0.9	3

Neste cenário, foi possível executar o código 33 vezes de modo que os dados necessários para o cálculo da taxa de acerto, que estão disponíveis no Apêndice B, puderam ser obtidos. Mediante a aplicação do *boxplot* disponível no Matlab, a Figura 18 pôde ser construída. Assim como no Experimento IV, não foram detectados *outliers* e a mediana ficou entre o 1º e 3º quartil. Entretanto, observou-se também que neste caso a mediana foi menor que no outro experimento. Além do mais, a média da quantidade de iterações foi de aproximadamente 1300000, valor superior ao dos Experimentos IV e V. Verificou-se também que a alternância no número de iterações entre números maiores e menores que 1000000 foi alta.

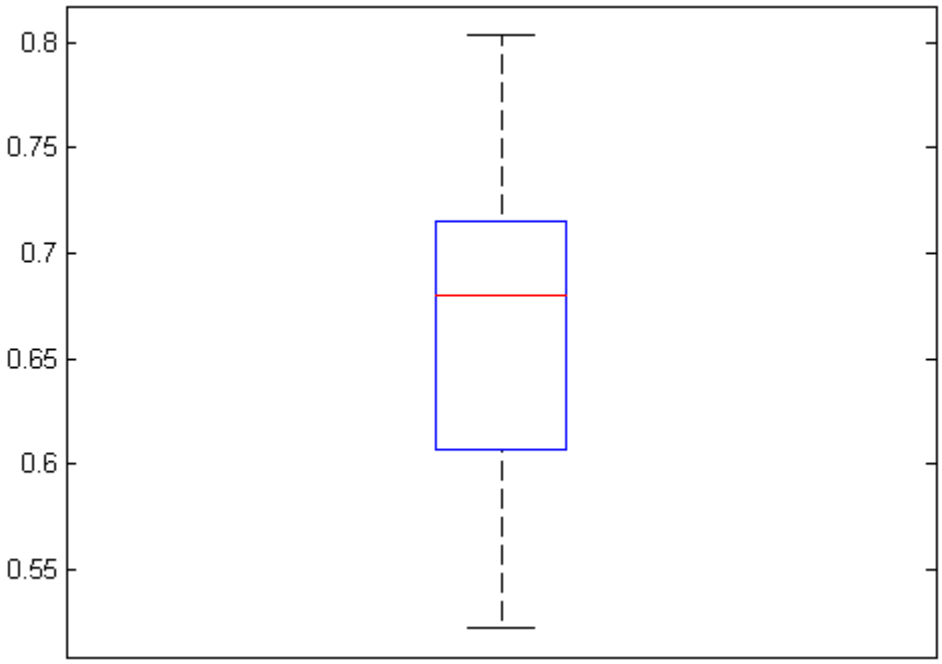

Figura 18. *Boxplot* da taxa de acerto do Experimento VI

4.3 Aplicação dos Testes e Análise dos Resultados

Como anteriormente mostrado na seção 3.3.4, os testes estatísticos aplicados aos dados coletados nos experimentos foram os testes t de *Student* e *Wilcoxon*. A seção 4.3.1 descreve a aplicação destes dois testes nos experimentos envolvendo a base de dados Iris enquanto que a seção 4.3.2 faz uso dos experimentos da base de dados *Wine*. Na seção 4.3.3 é feita uma análise dos resultados obtidos com os testes estatísticos.

Os testes foram realizados através do software R. Cada teste consiste em comparar-se o valor p-*value* obtido com o R com o valor do nível de significância proposto para os experimentos. No presente projeto, como o intervalo de confiança escolhido foi de 95% então o nível de significância corresponde a 5% (ou 0.05). Caso o p-*value* obtido seja maior que o nível de significância então a hipótese nula é aceita, do contrário, a hipótese alternativa é que deve ser aceita.

4.3.1 Testes Estatísticos na Base de Dados Iris

Os dados utilizados nesta primeira parte foram os obtidos a partir dos Experimentos I, II e III, disponibilizados no Apêndice A (foram utilizados os valores reais da taxa de acerto, não os valores percentuais). Como para os dados obtidos pelo algoritmo SACA com medição de insucessos foram detectados 2 *outliers* (ver seção 4.1.3), foram consideradas para uso nos testes estatísticos as 31 primeiras amostras coletadas nos referidos experimentos, desconsiderando os casos que foram categorizados como *outliers*.

As hipóteses nulas e alternativas que foram consideradas nos testes estão elencadas na seção 3.3.2. O Parâmetro P1 (ver seção 3.3.1) considerado neste caso foi a base de dados Iris. Os resultados obtidos são mostrados na Tabela 9. Pode-se ver que em todos os testes que foram feitos, as hipóteses alternativas foram aceitas e, consequentemente, as hipóteses nulas foram rejeitadas. A Hipótese Alternativa $H_{1\text{-}1,1}$, que foi aceita tanto no teste t de *Student* quanto no de *Wilcoxon*, mostrou que o desempenho do algoritmo SACA com medição de insucessos em relação à taxa

de acerto foi melhor que o do SACA na versão original. Observa-se que a ordem de grandeza do p-*value* obtido nestes dois casos foi muito baixa comparando com o nível de significância. Por sua vez, a Hipótese Alternativa $H_{1\text{-}1,2}$, também aceita no teste t de *Student* e no de *Wilcoxon*, mostrou que o desempenho do SACA com medição de insucessos, em relação à taxa de acerto, foi melhor que o SACA com melhoria no parâmetro α.

Tabela 9. Resumo dos Testes Estatísticos na Base de Dados Iris

Teste Estatístico	**Hipóteses Nulas**	**Hipóteses Alternativas**	**p-*value* obtido no R**	**Nível de Significância**	**Resultado do Teste**
t de *Student*	$H_{0\text{-}1,1}$	$H_{1\text{-}1,1}$	8.389E-9	0.05	Hipótese $H_{1\text{-}1,1}$ é aceita
	$H_{0\text{-}1,2}$	$H_{1\text{-}1,2}$	0.02534	0.05	Hipótese $H_{1\text{-}1,2}$ é aceita
Wilcoxon	$H_{0\text{-}1,1}$	$H_{1\text{-}1,1}$	1.784E-7	0.05	Hipótese $H_{1\text{-}1,1}$ é aceita
	$H_{0\text{-}1,2}$	$H_{1\text{-}1,2}$	0.03925	0.05	Hipótese $H_{1\text{-}1,2}$ é aceita

4.3.2 Testes Estatísticos na Base de Dados *Wine*

A segunda parte dos testes estatísticos foi aplicada a partir dos dados coletados nos Experimentos IV, V e VI, que estão disponíveis no Apêndice B (foram utilizados os valores reais da taxa de acerto, não os valores percentuais). Como descrito na

seção 4.2.2, não foi possível a obtenção da taxa de acerto no Experimento V (SACA com melhoria no parâmetro α aplicada na base de dados *Wine*), motivo pelo qual os testes estatísticos não foram aplicados neste caso.

Mais uma vez, as hipóteses nula e alternativa utilizadas estão disponíveis na seção 3.3.2 (como o Experimento V não foi utilizado, as hipóteses usadas foram $H_{0\text{-}1,1}$ e $H_{1\text{-}1,1}$). O Parâmetro P1 foi a base de dados *Wine*. A Tabela 10 resume os resultados dos dois testes. Através dela, pode-se ver que a Hipótese Nula $H_{0\text{-}1,1}$ foi aceita tanto no teste t de *Student* quanto de *Wilcoxon*. Em outras palavras, o resultado mostrou que o desempenho do algoritmo SACA com medição de insucessos em relação à taxa de acerto foi pior que o do SACA na versão original.

Tabela 10. Resumo dos Testes Estatísticos na Base de Dados *Wine*

Teste Estatístico	**Hipóteses Nulas**	**Hipóteses Alternativas**	**p-*value* obtido no R**	**Nível de Significância**	**Resultado do Teste**
t de *Student*	$H_{0\text{-}1,1}$	$H_{1\text{-}1,1}$	0.9895	0.05	Hipótese $H_{0\text{-}1,1}$ é aceita
Wilcoxon	$H_{0\text{-}1,1}$	$H_{1\text{-}1,1}$	0.9834	0.05	Hipótese $H_{0\text{-}1,1}$ é aceita

4.3.3 Análise dos Resultados

Os resultados obtidos pela aplicação dos testes estatísticos na base de dados Iris refletem que o algoritmo SACA com medição de insucessos produziu agrupamentos cuja taxa de acerto foi maior que o obtido a partir dos agrupamentos formados pelos algoritmos SACA em sua forma original e SACA com melhoria do parâmetro α. Entretanto observa-se também que, apesar de ser inferior, os testes estatísticos mostraram certa proximidade da taxa de acerto do SACA com melhoria do

parâmetro α em relação ao SACA com medição de insucessos. Por fim, além de ter sido o mais assertivo, o SACA com medição de insucessos também fez uso em média de menos iterações nas execuções do código (aproximadamente 183000 enquanto que os demais algoritmos utilizaram 300000). O fato de os resultados obtidos terem sido os mesmos nos dois testes estatísticos reforça os argumentos usados nesta análise.

Por sua vez, os testes estatísticos aplicados na base de dados *Wine* produziram resultados diferentes em relação à base de dados Iris. Como dito na seção 4.3.2, nesta segunda parte dos testes estatísticos apenas uma comparação foi realizada (SACA com medição de insucessos e SACA em sua forma original). Neste caso, o SACA com medição de insucessos foi menos assertivo que o SACA em sua forma original, tanto no teste t de *Student* quanto *Wilcoxon*, além de a média de iterações nas execuções do código ter sido maior (aproximadamente 1300000 enquanto que o outro algoritmo utilizou 1000000).

É possível que esta diferença nos cenários obtidos entre estas duas bases de dados esteja relacionada justamente a natureza dos dados envolvidos. Nota-se que o número de atributos referentes aos objetos pertencentes a base de dados *Wine* é um valor bem maior que o da base de dados Iris (13 contra 4), consequentemente, dificultando a formação de agrupamentos. Importante lembrar que o SACA com melhoria no parâmetro α não foi capaz de produzir agrupamentos na base de dados *Wine*, mostrando uma possível sensibilidade por parte destes dados a variações de α ao longo das execuções. Outra possibilidade também é que a escolha do parâmetro $Qtd_{insucessos}$ (ver seção 4.2.3) para a base de dados *Wine* pode não ter sido a melhor visto que o SACA com medição de insucessos obteve menor desempenho em relação à taxa de acerto, contrariando a expectativa do projeto e os resultados obtidos pelos testes na base de dados Iris.

Capítulo 5

Conclusões e Trabalhos Futuros

Este trabalho de conclusão de curso teve por objetivo estudar melhorias no algoritmo de agrupamento SACA inspirado no comportamento de colônia de formigas da espécie *Pheidolle pallidula*. Para este fim, foi utilizado o SACA em sua forma original, com modificação no parâmetro α e com medição de insucessos em um agente encarregado de deixar o objeto em uma nova posição. Estes três cenários foram aplicados em duas diferentes bases de dados.

Os resultados coletados mostraram que diferentes parametrizações do algoritmo SACA foram melhores ou piores em alguns casos. Isto mostrou que o sucesso deste tipo de mecanismo de agrupamento depende da escolha dos parâmetros envolvidos para a consequente execução do código, e que esta escolha tem forte dependência com a natureza dos dados envolvidos.

O algoritmo SACA com medição de insucessos obteve melhor desempenho no tocante a taxa de acerto comparando com os demais casos, pelo menos em uma das bases de dados utilizadas. Apesar de ainda não ser conclusivo, este resultado serviu para mostrar uma possível importante alternativa ao algoritmo SACA em sua forma original, resolvendo um provável problema de recursão infinita no mesmo.

Além do mais, soma-se ao ganho anteriormente citado o fato de que neste tipo de configuração do SACA, o número de iterações ter sido menor que nos demais cenários em experimentos aplicados a base de dados Iris, o que indica um menor esforço computacional ou, em aplicações práticas, esforço operacional necessário para a formação dos agrupamentos. Entretanto, como isto não aconteceu também com a base de dados *Wine*, outros estudos na área devem ser feitos antes de qualquer conclusão.

Outra importante contribuição nesta diferente forma de agrupamento é o critério de parada do algoritmo basear-se em uma condição, diferentemente do critério até então utilizado baseado no número de iterações. Bom lembrar também

que a simplicidade no algoritmo SACA, mesmo com essa modificação, se mantem, motivo pelo qual esse pode ser um novo caminho à ser usado em aplicações de clusterização de objetos, em especial, as que têm sido desenvolvidas na área de robótica.

Certamente, mais estudos envolvendo estes mecanismos de agrupamento devem ser feitos. Como melhorias futuras, pode-se destacar o uso de novos critérios de parada para o algoritmo SACA. Dentre eles, um bom caminho seria o cálculo do Erro Médio Quadrático (EMQ) ao longo das iterações com o objetivo de parar a execução quando este parâmetro atingir o menor valor possível. Outra possível vertente seria fazer a medição do percentual de modificação dos objetos pertencentes a um *cluster* no decorrer de cada iteração. Neste caso, o critério de parada utilizado seria quando este percentual atingisse certo valor previamente estabelecido.

Bibliografia

[1] BECKERS, R.; HOLLAND, O.; DENEUBOURG, J. L. **From local actions to global tasks: Stigmergy and collective robotics.** In: Proceedings of the Fourth International Conference on Artificial Life, 1994, Cambridge, p. 181-189.

[2] CASTRO, L. N. **Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications.** Taylor & Francis, 2006. 696 p.

[3] DENEUBOURG, J. L., et al. **The dynamics of collective sorting robot-like ants and ant-like robots.** In: Proceedings of The First International Conference on Simulation of Adaptive Behavior on From animals to animats, 1991, Cambridge, p. 356-363.

[4] DORIGO, M. Di Caro; GAMBARDELLA, L. M. Ant algorithms for Discrete Optmization. **Artificial Life**, MIT, Michigan, vol. 5, p. 137-172. 1999.

[5] FRANKS, N. R.; SENDOVA-FRANKS, A. B. Brood sorting by ants: distributing the workload over the work-surface. **Behavioral Ecology and Sociobiology.** 1992. V. 30, n. 2, p. 109-123.

[6] HAN, J.; KAMBER, M. **Data Mining Concepts and Techniques.** San Francisco, USA: Morgan Kaufmann Publishers, 2001. 533 p.

[7] HANDL, J. **Ant-based methods for tasks of clustering and topographic mapping: extensions, analysis and comparison with alternative techniques**. 2003. Dissertação de mestrado, Universidade Erlangen-Nürnberg, Erlangen, Alemanha.

[8] HARTMANN, V. **Evolving agents warms for clustering and sorting.** In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, 2005, Washington DC, p. 217-224.

[9] LAURO, A. L. **Agrupamento de Dados Utilizando Algoritmo de Colônia de Formigas**. 2008. Dissertação de mestrado do Curso de Engenharia Civil, UFRJ/COPPE, Rio de Janeiro.

[10] LEVINE, D. M.; BERENSON, M. L.; STEPHAN D. **Estatística: Teoria e Aplicações.** Rio de Janeiro: Editora LTC, 1998. 811 p.

[11] LUMER, E. D.; FAIETA, B. **Diversity and adaptation in populations of clustering ants.** In: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, 1994, Brighton, p. 501-508.

[12] MONMARCHÉ, N., **On Data clustering with artificial ants.** In: A.A. Freitas, editor, AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions, 1999, Orlando, Florida, p. 23-26.

[13] UCI. **Machine Learning Repository.** Disponível em: <http://archive.ics.uci.edu/ml/>. Acesso em: 17 de outubro de 2012.

[14] VIZINE, A. L., et al. Towards Improving Clustering Ant: An Adaptative Ant Clustering Algorithm. **Informatica**, n.29, p. 143-154. 2005.

[15] WATTHAYU, W. **Ant Colony Algorithm-Based Travelling Route Problems: A Case Study in Bangkok.** In: Computing Technology and Information Management (ICCM), 8th International Conference on, 2012, Seoul, p. 289-292.

Apêndice A
Resultados dos Experimentos na Base de Dados Iris

Tabela 11. Taxa de acerto nos experimentos da base de dados Iris

Caso de teste	Taxa de acerto (%)		
	SACA original	SACA com melhoria no parâmetro α	SACA com medição de insucessos
1	93	93	95
2	93	95	93
3	94	95	95
4	93	93	95
5	93	95	94
6	94	95	94
7	93	93	93
8	93	95	94
9	94	93	95
10	93	95	95
11	93	93	94
12	93	95	95

13	93	93	95
14	94	95	95
15	94	93	94
16	93	95	93
17	93	93	95
18	93	95	91
19	93	93	94
20	94	95	94
21	93	93	95
22	93	95	93
23	95	93	95
24	94	95	95
25	93	93	95
26	93	95	96
27	93	93	95
28	94	95	95
29	93	93	95
30	94	95	95
31	93	93	93
32	94	95	94

33	93	93	95

Apêndice B
Resultados dos Experimentos na Base de Dados *Wine*

Tabela 12. Taxa de acerto nos experimentos da base de dados *Wine*

Caso de teste	Taxa de acerto (%)	
	SACA original	SACA com medição de insucessos
1	78	70
2	78	69
3	72	60
4	69	69
5	71	75
6	76	62
7	62	63
8	66	74
9	62	58
10	72	52
11	77	69
12	71	57
13	63	75

14	62	72
15	72	75
16	75	61
17	78	61
18	70	68
19	73	60
20	69	69
21	78	62
22	70	74
23	67	66
24	61	56
25	68	69
26	62	71
27	71	67
28	63	65
29	65	71
30	67	80
31	81	58
32	64	72
33	82	66

Apêndice C

Código do Matlab para Agrupamento da Base de Dados Iris através do SACA Original

```
% Código do SACA na versão original aplicado a base de dados Iris
% by Ruben Araújo


% Definição dos atributos

Nobjetos = 150; % Número de objetos a serem agrupados
Niteracoes = 2000*Nobjetos; % Número de iterações
dim = cast(sqrt(10*Nobjetos),'int8'); % dimensão da grade
grade = zeros(dim,dim); % inicialização da matriz que representa o espaço
dos objetos
agentes = zeros(dim,dim); % inicialização da matriz que representa a
posição dos agentes
agentes_com_objetos = zeros(dim,dim); % inicialização da matriz que
representa os agentes
                                      % segurando ou não algum objeto
objetos = zeros(dim,dim); % inicialização da matriz que armazena cada
objeto individualmente na grade
Nagentes = 10; % Número de agentes
Ngrupos = 3; % Número de grupos distintos de objetos

% Agentes
vetor_agentes = [11,12,13,14,15,16,17,18,19,20];

% Posição atual de cada agente
    % Linha
    vetor_agentes_linhas = [0,0,0,0,0,0,0,0,0,0];

    % Coluna
    vetor_agentes_colunas = [0,0,0,0,0,0,0,0,0,0];

% Quantidade de objetos em cada grupo
grupos = [50,50,50];

% Valoração de cada grupo
valor_grupos = [5,50,500];  % 5-Iris-setosa (vermelho)  50-Iris-versicolour
(verde)  500-Iris-virginica (azul)

% Parâmetros de probabilidade
pp = 0.1;  % Probabilidade de pegar
pd = 0.1;  % Probabilidade de deixar
```

```
% Matriz com atributos de cada objeto
arquivo = fopen('C:\bases\Iris.txt');
dados = textscan(arquivo,'%f,%f,%f,%f,%u','Delimiter','\n');
fclose(arquivo);

% Matriz com atributos normalizados de cada objeto
max_coluna1 = max(dados{1});
max_coluna2 = max(dados{2});
max_coluna3 = max(dados{3});
max_coluna4 = max(dados{4});

dados_normalizados = zeros(Nobjetos,4);

for i = 1:Nobjetos
    for j = 1:4
        if (j==1)
            dados_normalizados(i,j) = dados{j}(i)/max_coluna1;
        end
        if (j==2)
            dados_normalizados(i,j) = dados{j}(i)/max_coluna2;
        end
        if (j==3)
            dados_normalizados(i,j) = dados{j}(i)/max_coluna3;
        end
        if (j==4)
            dados_normalizados(i,j) = dados{j}(i)/max_coluna4;
        end
    end
end

% Matriz dissimilaridade dos objetos
matriz_dissimilaridade = zeros(Nobjetos,Nobjetos);

for i = 1:Nobjetos
    for j = 1:Nobjetos
        if (i~=j)
           matriz_dissimilaridade(i,j) = sqrt((dados_normalizados(i,1) -
dados_normalizados(j,1))^2 + (dados_normalizados(i,2) -
dados_normalizados(j,2))^2 + (dados_normalizados(i,3) -
dados_normalizados(j,3))^2 + (dados_normalizados(i,4) -
dados_normalizados(j,4))^2);
        end
    end
end

% Parâmetros para cálculo de probabilidade de pegar objeto ou deixar

kp = 0.1;  % pegar
kd = 0.25;  % deixar

alfa = 0.4; % Escalar que estabelece a dependência em relação aos dados
sigma = 3; % Define o tamanho da vizinhança (neste caso, são as 8 células
ao redor da célula central)

% Carrego inicial da grade com objetos

x = 1;
```

```
for i = 1:Ngrupos

    aux = grupos(i);
    while (aux > 0)
        a = cast(dim*rand,'int8');
        while (a == 0)
            a = cast(dim*rand,'int8');
        end
        b = cast(dim*rand,'int8');
        while (b == 0)
            b = cast(dim*rand,'int8');
        end
        if (grade(a,b) == 0)
            grade(a,b) = valor_grupos(i);
            objetos(a,b) = x;
            x = x + 1;
            aux = aux - 1;
        end
    end

end


% Seleção inicial aleatória dos objetos por cada agente

for i = 1:Nagentes

   flag = true;
   % Selecionando objeto aleatoriamente
   while (flag)
        a = cast(dim*rand,'int8');
        while (a == 0)
            a = cast(dim*rand,'int8');
        end
        b = cast(dim*rand,'int8');
        while (b == 0)
            b = cast(dim*rand,'int8');
        end
        if (grade(a,b) ~= 0 && agentes_com_objetos(a,b) == 0)
            agentes(a,b) = vetor_agentes(i);
            agentes_com_objetos(a,b) = 1;
            vetor_agentes_linhas(i) = a;
            vetor_agentes_colunas(i) = b;
            flag = false;
        end
    end

    % Escolhendo aleatoriamente nova posição na grade
        a = cast(dim*rand,'int8');
        while (a == 0)
            a = cast(dim*rand,'int8');
        end
        b = cast(dim*rand,'int8');
        while (b == 0)
            b = cast(dim*rand,'int8');
        end

     while (grade(a,b) ~= 0)
        a = cast(dim*rand,'int8');
```

```
        while (a == 0)
            a = cast(dim*rand,'int8');
        end
        b = cast(dim*rand,'int8');
        while (b == 0)
            b = cast(dim*rand,'int8');
        end
     end

    % Nova posição
    grade(a,b) = grade(vetor_agentes_linhas(i),vetor_agentes_colunas(i));
    objetos(a,b) =
objetos(vetor_agentes_linhas(i),vetor_agentes_colunas(i));
    agentes(a,b) =
agentes(vetor_agentes_linhas(i),vetor_agentes_colunas(i));
    agentes_com_objetos(a,b) =
agentes_com_objetos(vetor_agentes_linhas(i),vetor_agentes_colunas(i));
    % Antiga posição
    grade(vetor_agentes_linhas(i),vetor_agentes_colunas(i)) = 0;
    objetos(vetor_agentes_linhas(i),vetor_agentes_colunas(i)) = 0;
    agentes(vetor_agentes_linhas(i),vetor_agentes_colunas(i)) = 0;
    agentes_com_objetos(vetor_agentes_linhas(i),vetor_agentes_colunas(i)) =
0;
    % Atualiza vetor de posições do agente
    vetor_agentes_linhas(i) = a;
    vetor_agentes_colunas(i) = b;

end

% Rotina principal

for i = 1:Niteracoes
    % Escolha aleatória do agente
    c = cast(Nagentes*rand,'int8');
    while (c == 0)
       c = cast(Nagentes*rand,'int8');
    end

    % Escolha aleatória da posição para mudança

        a = cast(dim*rand,'int8');
        while (a == 0)
           a = cast(dim*rand,'int8');
        end
        b = cast(dim*rand,'int8');
        while (b == 0)
           b = cast(dim*rand,'int8');
        end
        while (grade(a,b) ~= 0 || agentes(a,b) ~= 0)
            a = cast(dim*rand,'int8');
            while (a == 0)
                a = cast(dim*rand,'int8');
            end
            b = cast(dim*rand,'int8');
            while (b == 0)
                b = cast(dim*rand,'int8');
            end
        end
```

```
   % Decisão de deixar ou não o objeto na nova posição

       % Descobrindo a vizinhaça da nova posição

         if (a==1 && b~=1 && b~=dim)
            linhas_vizinho = [0,a,a+1];
            colunas_vizinho = [b-1,b,b+1];
         elseif (a==dim && b~=1 && b~=dim)
            linhas_vizinho = [a-1,a,0];
            colunas_vizinho = [b-1,b,b+1];
         elseif (b==1 && a~=1 && a~=dim)
            linhas_vizinho = [a-1,a,a+1];
            colunas_vizinho = [0,b,b+1];
         elseif (b==dim && a~=1 && a~=dim)
            linhas_vizinho = [a-1,a,a+1];
            colunas_vizinho = [b-1,b,0];
         elseif (a==1 && b==1)
            linhas_vizinho = [0,a,a+1];
            colunas_vizinho = [0,b,b+1];
         elseif (a==1 && b==dim)
            linhas_vizinho = [0,a,a+1];
            colunas_vizinho = [b-1,b,0];
         elseif (a==dim && b==1)
            linhas_vizinho = [a-1,a,0];
            colunas_vizinho = [0,b,b+1];
         elseif (a==dim && b==dim)
            linhas_vizinho = [a-1,a,0];
            colunas_vizinho = [b-1,b,0];
         else
            linhas_vizinho = [a-1,a,a+1];
            colunas_vizinho = [b-1,b,b+1];
         end

        % Função densidade
        funcao_densidade = 0;

        for g = 1:sigma
            for h = 1:sigma
                if (g~=h && linhas_vizinho(g)~=0 && colunas_vizinho(h)~=0
&& objetos(linhas_vizinho(g),colunas_vizinho(h))~=0)
                    funcao_densidade = funcao_densidade + (1/(sigma^2))*(1-
((matriz_dissimilaridade(objetos(vetor_agentes_linhas(c),vetor_agentes_colu
nas(c)),objetos(linhas_vizinho(g),colunas_vizinho(h))))/alfa));
                end
            end
        end
         if (funcao_densidade <=0)
            funcao_densidade = 0;
         end

        % Função probabilidade de deixar o objeto
        probabilidade_deixar = (funcao_densidade/(kd +
funcao_densidade))^2;

        % Atualiza nova posição
        grade(a,b) =
grade(vetor_agentes_linhas(c),vetor_agentes_colunas(c));
        objetos(a,b) =
objetos(vetor_agentes_linhas(c),vetor_agentes_colunas(c));
```

```
        agentes(a,b) =
agentes(vetor_agentes_linhas(c),vetor_agentes_colunas(c));
        agentes_com_objetos(a,b) =
agentes_com_objetos(vetor_agentes_linhas(c),vetor_agentes_colunas(c));
        % Atualiza antiga posição
        grade(vetor_agentes_linhas(c),vetor_agentes_colunas(c)) = 0;
        objetos(vetor_agentes_linhas(c),vetor_agentes_colunas(c)) = 0;
        agentes(vetor_agentes_linhas(c),vetor_agentes_colunas(c)) = 0;

agentes_com_objetos(vetor_agentes_linhas(c),vetor_agentes_colunas(c)) = 0;
        % Atualiza vetor de posições do agente
        vetor_agentes_linhas(c) = a;
        vetor_agentes_colunas(c) = b;

        % Decisão de deixar objeto
        if (probabilidade_deixar >= pd)

            % Agente vai sair a procura de novo objeto
            flag = true;
               % Selecionando objeto aleatoriamente
               while (flag)
                    x = cast(dim*rand,'int8');
                    while (x == 0)
                        x = cast(dim*rand,'int8');
                    end
                    y = cast(dim*rand,'int8');
                    while (y == 0)
                        y = cast(dim*rand,'int8');
                    end
                    if (grade(x,y) ~= 0 && agentes_com_objetos(x,y) == 0)

                         % Decisão de pegar ou não o objeto na nova posição

                                  % Descobrindo a vizinhaça da nova
posição


                                    if (x==1 && y~=1 && y~=dim)
                                       linhas_vizinho = [0,x,x+1];
                                       colunas_vizinho = [y-1,y,y+1];
                                    elseif (x==dim && y~=1 && y~=dim)
                                       linhas_vizinho = [x-1,x,0];
                                       colunas_vizinho = [y-1,y,y+1];
                                    elseif (y==1 && x~=1 && x~=dim)
                                       linhas_vizinho = [x-1,x,x+1];
                                       colunas_vizinho = [0,y,y+1];
                                    elseif (y==dim && x~=1 && x~=dim)
                                       linhas_vizinho = [x-1,x,x+1];
                                       colunas_vizinho = [y-1,y,0];
                                    elseif (x==1 && y==1)
                                       linhas_vizinho = [0,x,x+1];
                                       colunas_vizinho = [0,y,y+1];
                                    elseif (x==1 && y==dim)
                                       linhas_vizinho = [0,x,x+1];
                                       colunas_vizinho = [y-1,y,0];
                                    elseif (x==dim && y==1)
                                       linhas_vizinho = [x-1,x,0];
                                       colunas_vizinho = [0,y,y+1];
                                    elseif (x==dim && y==dim)
```

```
                                        linhas_vizinho = [x-1,x,0];
                                        colunas_vizinho = [y-1,y,0];
                                    else
                                        linhas_vizinho = [x-1,x,x+1];
                                        colunas_vizinho = [y-1,y,y+1];
                                    end

                                    % Função densidade
                                    funcao_densidade = 0;

                                    for g = 1:sigma
                                        for h = 1:sigma
                                            if (g~=h &&
linhas_vizinho(g)~=0 && colunas_vizinho(h)~=0 &&
objetos(linhas_vizinho(g),colunas_vizinho(h))~=0)
                                                funcao_densidade =
funcao_densidade + (1/(sigma^2))*(1-
((matriz_dissimilaridade(objetos(x,y),objetos(linhas_vizinho(g),colunas_viz
inho(h))))/alfa));
                                            end
                                        end
                                    end
                                    if (funcao_densidade <=0)
                                        funcao_densidade = 0;
                                    end

                                    % Função probabilidade de pegar o
objeto
                                    probabilidade_pegar = (kp/(kp +
funcao_densidade))^2;

                            if (probabilidade_pegar>=pp)
                                % Atualiza nova posição do agente
                                agentes(x,y) = c;
                                agentes_com_objetos(x,y) = 1;
                                % Atualiza antiga posição do agente
                                agentes(a,b) = 0;
                                agentes_com_objetos(a,b) = 0;
                                % Atualiza vetor de posições do agente
                                vetor_agentes_linhas(c) = x;
                                vetor_agentes_colunas(c) = y;
                                flag = false;

                            end

                        end
                    end

                end

end



grade1 = zeros(dim,dim);
grade2 = zeros(dim,dim);
grade3 = zeros(dim,dim);
```

```
for x = 1:dim
    for y = 1:dim

        % Desenhando a grade
        if (grade(x,y)==valor_grupos(1))
            grade1(x,y)=grade(x,y);
        elseif (grade(x,y)==valor_grupos(2))
            grade2(x,y)=grade(x,y);
        elseif (grade(x,y)==valor_grupos(3))
            grade3(x,y)=grade(x,y);
        end
    end
end

% Geração do gráfico
hold on;
spy(grade1,'-.or');
spy(grade2,'-.*g');
spy(grade3,'-..b');
```