UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

SAMUEL LODETTI GHELLERE

ALGORITMO *STANDARD ANT CLUSTERING ALGORITHM* NA TAREFA DE CLUSTERIZAÇÃO DA *SHELL* ORION *DATA MINING ENGINE*

CRICIÚMA
2012

SAMUEL LODETTI GHELLERE

ALGORITMO *STANDARD ANT CLUSTERING ALGORITHM* NA TAREFA DE CLUSTERIZAÇÃO DA *SHELL* ORION *DATA MINING ENGINE*

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientadora: Profª. MSc. Merisandra Côrtes de Mattos Garcia.

CRICIÚMA

2012

SAMUEL LODETTI GHELLERE

ALGORITMO *STANDARD ANT CLUSTERING ALGORITHM* NA TAREFA DE CLUSTERIZAÇÃO DA *SHELL* ORION *DATA MINING ENGINE*

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em *Data mining*.

Criciúma, 27 de Junho de 2012.

BANCA EXAMINADORA

Profª. Merisandra Côrtes de Mattos Garcia – Mestre - (UNESC) - Orientador

Prof. Kristian Madeira - Mestre - (UNESC)

Fábio Bif Goularte - Especialista - (UNESC)

RESUMO

O constante avanço da tecnologia e a facilidade de armazenamento geram grandes bases de dados. Considerando que as informações contidas nessas bases são de grande interesse das organizações, tecnologias com o objetivo de explorar essas informações são necessárias para extrair conhecimento novo e útil dessas bases de dados. *Data mining* é uma dessas tecnologias, que utiliza diversos algoritmos com finalidade de descobrir conhecimento nas bases de dados onde é aplicada, sendo implementadas em ferramentas computacionais, denominadas *shell*, que em sua maioria não são gratuitas. A *Shell* Orion *Data Mining Engine* é uma dessas ferramentas, a mesma é mantida em desenvolvimento pelo Grupo de Pesquisa em Inteligência Computacional Aplicada do Curso de Ciência da Computação da UNESC, que implementa diversos métodos e tarefas de *data mining*. Dessa forma, o objetivo dessa pesquisa consiste em ampliar as funcionalidades da *Shell* Orion implementando e demonstrando o funcionamento do algoritmo *Standard Ant Clustering Algorithm* (SACA) para a tarefa de clusterização. Este algoritmo é baseado em um modelo encontrado na natureza de uma área da inteligência computacional que é denominada como inteligência de enxame. O SACA surgiu com a observação do comportamento coletivo de espécies de formigas, mais precisamente na organização de cemitérios, esse comportamento coletivo é utilizado pelo SACA para formar grupos de dados similares. Algoritmos como *Ant Based Clustering* e o A²CA se originaram de estudos sobre o comportamento do algoritmo SACA, a fim de melhorar os resultados obtidos pelo mesmo. Ao final da pesquisa foram efetuados testes que comprovaram, junto com os métodos de validação aplicados aos resultados, o correto funcionamento dos módulos implementados além de uma comparação de desempenho entre eles.

Palavras – Chave: Inteligência Computacional. *Data Mining*. Clusterização. Inteligência de enxame. Algoritmo SACA.

ABSTRACT

The constant advance of technology and the facility of storage generates large databases. Considering that the information contained in these databases are of great interest for organizations, technologies in order to explore these informations are necessary to extract the new and useful knowledge of these databases. Data mining is one of these technologies , which makes use of several algorithms with the purpose of discovering knowledge in databases where it is applied, being implemented in computational tools, called shell, which mostly are not free. Shell Orion Data Mining Engine is one of those tools, and is maintained under development by the Research Group at Applicated Computation Intelligence Course of Computation Science at UNESC, which implements several methods and data mining tasks. Thus, the objective of this research is to extend the functionality of Orion Shell implementing and demonstrating the operation of the algorithm Standard Ant Clustering Algorithm (SACA) to the task of clustering. This algorithm is based on a pattern found in nature in an area of computational intelligence that is called swarm intelligence. The SACA has emerged with the observation of the collective behavior of ant species, specifically the organization of cemeteries, this collective behavior is used by SACA to form groups of similar data. Algorithms such as Ant Based Clustering and A ² CA originated from studies on the behavior of the algorithm SACA in order to improve the results obtained by the same. At the end of the study tests were performed and they confirmed, along with validation methods applied to the results, the correct functioning of the implemented model.

Keywords: Computational Intelligence. Data Mining. Clustering. Swarm Intelligence. SACA Algorithm.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A²CA	*Adaptive Ant-Clustering Algorithm*
ACA	*Ant Clustering Algorithm*
ACO	*Ant Colony Optimization*
ACOC	*Ant Colony Optimization for Clustering*
API	*Application Programming Interface*
COPPE	Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
GTA	Grupo Técnico de Assessoramento
JDBC	Java *Database Connectivity*
KDD	*Knowledge Discovery in Databases*
PCA	*Principal Component Analysis*
RBF	*Radial Basis Function*
SACA	*Standart Ant Clustering Algorithm*
SGBD	Sistemas Gerenciadores de Bancos de Dados
SQL	*Structured Query Language*
TCC	Trabalhos de Conclusão de Curso
UFPR	Universidade Federal do Paraná
UML	*Unified Modeling Language*
UNESC	Universidade do Extremo Sul Catarinense

SUMÁRIO

1 INTRODUÇÃO

O constante avanço da tecnologia facilita o armazenamento de dados e permite cada vez mais que as instituições gerem grandes bases de dados. Essas bases se forem bem analisadas possuem informações valiosas sobre estratégia de negócios e tendências, sejam elas empresariais ou institucionais. Dessa forma, esse conteúdo é essencial para o planejamento das ações e tomadas de decisões futuras. Porém, devido ao tamanho dessas bases de dados a análise das informações tornou-se complexa para a capacidade humana. Em vista disso, foram desenvolvidas técnicas que são capazes de procurar e extrair informações significativas dos dados. Essa procura de relações entre os dados ficou conhecida como *Knowledge Discovery in Databases* (KDD), sendo o *data mining* a principal etapa desse processo.

Data mining é a exploração e análise, por meio automático ou semi-automático, de grandes quantidades de dados a fim de descobrir padrões. Nessa etapa são empregadas tarefas e métodos com características distintas, que são aplicadas de acordo com o objetivo da descoberta de conhecimento (BERRY, 2000, tradução nossa; GOLDSCHMIDT; PASSOS, 2005).

Na execução do *data mining* são necessários, além das tarefas, ferramentas computacionais que as implementem. Dentre as ferramentas existentes de *data mining*, encontra-se em desenvolvimento a *Shell* Orion *Data Mining Engine* que consiste em um projeto acadêmico que está sendo desenvolvido pelo Grupo de Pesquisa em Inteligência Computacional Aplicada do Curso de Ciência da Computação da UNESC. Este projeto objetiva a implementação de uma s*hell* de *data mining* gratuita, ferramenta esta que auxilia no processo de descoberta de conhecimento em base de dados, visto que a maioria destas ferramentas são comerciais.

A *Shell* Orion possui implementadas as tarefas de classificação (algoritmos ID3, C4.5, CART e *Radial Basis Function* (RBF)), de associação (algoritmo Apriori) e de clusterização (algoritmos *K-Means*, *Kohonen*, *Fuzzy C-Means*, *Gustafson-Kessel*, *Gath-Geva*, *Robust C-Prototypes (RCP)*, *Unsupervised Robust C-Prototypes (URCP)* e *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN)). A tarefa de clusterização tem por objetivo agrupar objetos similares em uma base de dados em subconjuntos ou *clusters* relativamente homogêneos. Um *cluster* consiste em uma coleção de objetos que são similares entre si e dissimilares entre objetos de outros *clusters* (LAROSE, 2005, tradução nossa).

Entre os vários métodos de clusterização existentes podem ser destacados os hierárquicos, de particionamento, baseados em grade e em densidade (HAN; KAMBER, 2006, tradução nossa).

Métodos tradicionais de clusterização geralmente encontram dificuldades para produzir *clusters* com formatos arbitrários e não retornam bons resultados quando a base de dados em questão está contaminada com ruídos e *outliers* (dados que não fazem parte do padrão geral da base). Outro ponto negativo de alguns destes métodos, é que o usuário tem a necessidade de informar o número de *clusters* que serão gerados, que na maioria das vezes não é uma tarefa simples. A maioria desses algoritmos são sensíveis aos parâmetros de entrada, pois definições diferentes desses parâmetros podem levar a outros agrupamentos de dados (HAN; KAMBER, 2006, tradução nossa).

Uma alternativa para esses problemas são os algoritmos de clusterização baseados em inteligência de enxames, que é a denominação aplicada ao desenvolvimento de algoritmos ou instrumentos para solução distribuída de problemas, inspirando-se no comportamento coletivo de colônias de insetos sociais e outras sociedades de animais. Esses algoritmos oferecem um caminho alternativo por possuírem flexibilidade[1], robustez[2] e auto-organização[3] (BONABEAU; THERAULAZ; DORIGO, 1999, tradução nossa).

Dentre os algoritmos baseados em inteligência de enxame tem-se o Algoritmo Simples de Clusterização por Formiga (*Standart Ant Clustering Algorithm* - SACA) que foi apresentado inicialmente por Lumer e Faieta (1994). Nesta pesquisa implementou-se a tarefa de clusterização, baseada em inteligência de enxame por colônia de formiga, pelo algoritmo SACA.

1.1 OBJETIVO GERAL

Disponibilizar o algoritmo *Standard Ant Clustering Algorithm* na tarefa de clusterização da *Shell* Orion *Data Mining Engine.*

---

[1] O grupo ou enxame pode se adaptar rapidamente as mudanças do ambiente (BONABEAU; THERAULAZ; DORIGO, 1999, tradução nossa).

[2] Quando um ou mais indivíduos falham, o grupo ou enxame continuam a executar as suas tarefas (BONABEAU; THERAULAZ; DORIGO, 1999, tradução nossa).

[3] Pouca supervisão ou controle é exigido pelo grupo ou enxame (BONABEAU; THERAULAZ; DORIGO, 1999, tradução nossa).

1.2 OBJETIVOS ESPECIFICOS

Os objetivos específicos desta pesquisa consistem em:

a) compreender o conceito de *data mining* e a tarefa de clusterização;
b) entender a inteligência de enxame e o método de colônias de formigas;
c) aplicar o algoritmo SACA na tarefa de clusterização de dados da *Shell* Orion *Data Mining Engine*;
d) demonstrar matematicamente o funcionamento do algoritmo SACA;
e) analisar o desempenho do algoritmo desenvolvido por meio de medidas estatísticas.

1.3 JUSTIFICATIVA

O *data mining* é o processo na etapa de descoberta de conhecimento em bases de dados que consiste na aplicação de algoritmos que sob limitações de eficiência computacional aceitáveis, produzem uma enumeração particular de padrões sobre os dados (FAYYAD, PIATETSKY-SHAPIRO, SMYTH, 1996, tradução nossa).

O *data mining* tem como objetivo a identificação do conhecimento em grandes bases de dados por meio de métodos específicos, podendo facilitar a tomada de decisões pela predição da ocorrência de padrões e relações entre estes dados (GOLDSCHMIDT; PASSOS, 2005).

A dificuldade em perceber e interpretar os fatos identificados durante o processo torna a descoberta de conhecimento bastante complexa, fazendo-se indispensável o uso de ferramentas que auxiliem esta atividade, tendo-se as *shells*. Além disso, várias delas apresentam limitações relacionadas à conexão com diferentes Sistemas Gerenciadores de Bancos de Dados (SGBD). Mediante isso, o projeto da *Shell* Orion *Data Mining Engine* tende a implementar as tarefas consideradas mais importantes no processo de *data mining* por meio de variados métodos em uma ferramenta gratuita. Permite também a conexão com os diferentes Sistemas Gerenciadores de Bancos de Dados (SGBD) *PostGreSQL*, *Firebird*, *HSQLDB*, *MySQL* e outros. Esta pesquisa amplia as funcionalidades da *Shell* Orion acrescentando na tarefa de clusterização o algoritmo SACA que consiste em um método baseado em inteligência de enxame por colônia de formiga.

A clusterização pode ser entendida como uma tarefa básica no processo de *data mining*, que auxilia na estruturação e compreensão do conjunto de dados original

identificando os grupos existentes em um conjunto de objetos. Outras tarefas de *data mining*, tais como a classificação e sumarização, podem realizar o seu trabalho nos *clusters* encontrados pela tarefa de clusterização (CARLANTONIO, 2001; HAN; KAMBER, 2006, tradução nossa).

Entre os métodos de clusterização se encontram os baseados em inteligência de enxame que estuda o comportamento coletivo dos insetos sociais e têm inspirado a ciência da computação a fazer simulações computacionais para tentar reproduzi-los. Há, pelo menos, dois grandes motivos para isto. O primeiro é que os mecanismos responsáveis por esses comportamentos são ainda desconhecidos e por meio de simulação, podem ser estudados, a fim de proporcionar melhor entendimento da natureza. O segundo motivo é que esses comportamentos coletivos possuem características de robustez e confiabilidade o que os tornam atraentes para pesquisas científicas. Além de que modelos computacionais baseados nas atividades de agrupamento e organização dos insetos têm levado à algoritmos de clusterização eficientes (DORIGO; STÜTZLE, 2004, tradução nossa).

O estudo do comportamento coletivo que levam a espécie de formiga *Pheidole pallidula*[4] a organizar seus corpos mortos em grupos de itens semelhantes resultou no desenvolvimento do algoritmo de clusterização SACA. O mesmo consiste, em uma simulação de colônia de agentes de formigas que se move aleatoriamente em uma grade toroidal (não possui extremidade) bidimensional, onde objetos (registros dos dados) estão posicionados. A principal vantagem desse método é o fato de que não é necessária nenhuma informação inicial a respeito da massa de dados que será particionada (LUMER; FAIETA, 1994, tradução nossa).

1.4 ESTRUTURA DO TRABALHO

Esta pesquisa é composta por seis capítulos, sendo que no capítulo 1 descreve-se o tema proposto, os objetivos pretendidos e a justificativa para a realização desse trabalho.

No capítulo 2 contextualizam-se os principais conceitos relacionados ao processo de KDD, bem como os referentes à etapa de *data mining* e a *Shell* Orion. Posteriormente, o capítulo 3 tem como tema a tarefa de clusterização em *data mining*, descrevendo-se os seus métodos. Neste capítulo também é apresentada a inteligência de enxame, abordando a origem

[4] Pertencem a um gênero que se alimenta de insetos vivos e carne podre, alimentos açucarados e resíduos de alimentos (WILSON, 2003).

dessa área na inteligência computacional e os algoritmos desenvolvidos, bem como o funcionamento do SACA, e alguns trabalhos correlatos ao desenvolvido nesta pesquisa e exemplos de aplicações do SACA.

No capítulo 4 são descritas as etapas do trabalho desenvolvido, a metodologia utilizada e os resultados obtidos pelo módulo do algoritmo SACA na Shell Orion. Finalizando, tem-se a conclusão da pesquisa e sugestões de trabalhos futuros.

Finalizando, tem-se a conclusão da pesquisa e sugestões de trabalhos futuros.

2 DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS

O avanço da tecnologia de coleta e armazenamento de dados permitiu que as organizações acumulassem uma vasta quantidade de dados em suas bases. No entanto, a extração de informação útil tem sido extremamente desafiadora devido ao tamanho dessas bases de dados (TAN; STEINBACH; KUMAR, 2009)

Na maioria das vezes a análise desses dados vem sendo realizada por recursos limitados como: planilhas eletrônicas, consultas *Structured Query Language* (SQL), ferramentas estatísticas, entre outros. Porém, esses recursos não facilitam a descoberta de padrões ou relações entre os dados (AMORIM, 2006).

Tendo esses acontecimentos, surgiu uma área para atender a necessidade de analisar essas informações armazenadas, cujo processo envolve banco de dados, inteligência artificial e estatística, denominada *Knowledge Discovery in Databases* (KDD) ou Descoberta de Conhecimento em Bases de Dados (DCBD) (GOLDSCHMIDT; PASSOS, 2005).

As atividades do KDD podem ser ordenadas em três grandes grupos (GOLDSCHMIDT; PASSOS, 2005):

a) **desenvolvimento tecnológico**: são os aspectos referentes as iniciativas de concepção, refinamento e desenvolvimento de algoritmos, ferramentas e tecnologias que possam ser empregadas em bases de dados, na busca por novos conhecimentos;
b) **execução de KDD**: consiste, efetivamente, no que se refere a busca do conhecimento na base de dados;
c) **aplicação dos resultados**: implica na utilização das informações úteis obtidas dos resultados alcançados pelo processo de KDD.

O Goldschmidt e Passos (2005) também indicam que o ponto de partida do processo de KDD deve ser a análise da base de dados onde se quer extrair o conhecimento. A partir dessa apreciação é possível, junto aos especialistas da área em que se irá analisar os dados, determinar os objetivos que devem ser alcançados neste processo e que deverão nortear todo o procedimento.

O processo de KDD é caracterizado como interativo, pois exige a presença de um especialista humano no domínio de aplicação da base de dados, e também iterativo, pois é executado repetidamente a fim de se aprimorar o resultado obtido (FAYYAD; PIATETSKY-SHAPINO; SMYTH, 1996, tradução nossa).

As etapas do processo de KDD (figura 1) vão desde a definição do domínio, seleção, preparação e transformação dos dados até a etapa de *data mining*, onde padrões podem ser descobertos e analisados para se tornarem conhecimento útil. Basicamente, no KDD consideram-se as seguintes etapas (HAN; KAMBER, 2006, tradução nossa):

a) **pré-processamento**: transforma os dados brutos de entrada em um formato apropriado para as análises subsequentes. A fusão de dados de múltiplas fontes, a limpeza e remoção de ruídos, a adequação de valores que estão fora de contexto, a seleção e o resumo das variáveis a serem utilizadas são as atividades compreendidas nesta fase (TAN; STEINBACH; KUMAR, 2009);

b) ***data mining***: Essa etapa busca por novos padrões que possam gerar conhecimento útil a partir da base de dados. É a parte principal do processo de descoberta de conhecimento (GOLDSCHIMIDT; PASSOS, 2005). Envolve a escolha da tarefa de *data mining* e dos algoritmos específicos para cada problema, que serão executados de forma iterativa (REZENDE, 2005);

c) **pós-processamento**: envolve a visualização, análise e interpretação do conhecimento gerado na etapa de *data mining.* Técnicas de visualização de dados podem ser utilizadas para gerar um resultado compreensível ao usuário. Essas técnicas estimulam a percepção e a inteligência humana e assim aumentam a capacidade de entendimento e a associação de novos padrões (GOLDSCHIMIDT; PASSOS, 2005).

Figura 1- Etapas do KDD

Fonte: Adaptado de Adrians e Zantinge (1996).

2.1 *DATA MINING*

A aquisição de padrões nos repositórios de dados é denominada *data mining*, essa nomenclatura muitas vezes é utilizada para todo o processo de descoberta de conhecimento, mas *data mining* é somente uma das etapas do KDD (HAN; KAMBER, 2001, tradução nossa).

Data mining consiste no processo de explorar e analisar grandes conjuntos de dados. Sendo responsável por extrair conhecimento sob a forma de novas relações e padrões úteis na resolução de problemas de um domínio de aplicação específico (SIVANANDAM; SUMATTI, 2006, tradução nossa).

Navega (2002) esclarece que na etapa de *data mining* são aplicados algoritmos diversos, que processam os dados na base, em busca de informações relevantes. Ele ainda aponta que, apesar dos algoritmos serem capazes de identificar dados válidos e novos, ainda não existe uma solução para determinar padrões valiosos. Nesse sentido a participação de especialistas é fundamental para completar o processo de identificação do conhecimento, determinar e conduzir a exploração da base de dados.

O *data mining* pode ser aplicado a vários campos de pesquisa, como por exemplo: segmentação de imagens de satélite, previsão de carga do sistema elétrico, mineração de dados na web, *marketing* e vendas, entre outros (WITTEN; FRANK; HALL, 2011, tradução nossa).

Tendo em vista os amplos domínios de sua aplicação, existem diversas tarefas de *data mining* que podem ser escolhidas dependendo do conhecimento que se deseja obter. Entre elas estão às tarefas de:

a) **associação**: baseia-se na ação de encontrar um grupo de itens afins, ou seja, descobrir quais atributos aparecem com maior frequência ou que ocorram concomitantemente e de forma repetida em uma base de dados (GOLDSCHIMIDT; PASSOS, 2005). Os padrões descobertos pela tarefa de associação são normalmente representados na forma de regras de implicação ou subconjuntos de características (TAN; STEINBACH; KUMAR, 2009);
b) **classificação**: busca por uma função que permita integrar corretamente cada registro de um conjunto de informações a um único rótulo de um conjunto de classes (GOLDSCHIMIDT; PASSOS, 2005). Conforme Han e Kamber (2006), tradução nossa, esta tarefa é dividida em duas etapas. A primeira define o

modelo de dados baseando-se nos registros da base a ser minerada, já a segunda utiliza este modelo para efetuar a classificação dos demais objetos;

c) **estimativa**: também conhecida como regressão, essa tarefa busca por funções, sejam lineares ou não, que possam mapear os registros de uma base de dados. Tem como objetivo determinar valores prováveis de índices em dados do passado ou de índices semelhantes que se tem conhecimento. Essa tarefa trabalha com resultados contínuos (GOLDSCHIMIDT; PASSOS, 2005; OLIVEIRA, 2008);

d) **previsão**: tarefa de previsão de um valor, baseando seu juízo nos dados históricos armazenados, para uma determinada variável. Ela prevê o comportamento dos registros que ainda serão inseridos em uma base de dados por meio da construção de modelos para avaliar os dados que já pertencem a base de dados (BARRETO, 2004);

e) **clusterização**: conhecida também por agrupamento, é utilizada para separar os registros de uma base de dados em *clusters* (subconjuntos), permitindo que os dados de um *cluster* compartilhem as mesmas propriedades e características (GOLDSCHIMIDT; PASSOS, 2005). Assim, forma conjuntos de objetos contendo similaridades entre os elementos do mesmo grupo e com diferenças com relação aos itens dos outros *clusters* (HAN; KAMBER, 2006).

Na tabela 1 tem-se uma visão geral das tarefas de *data mining*, citadas anteriormente, e dos seus exemplos de utilização.

Tabela 1-Principais tarefas de d*ata mining*.

Tarefa	Descrição	Exemplos
Associação	Baseia na ação de encontrar um grupo de itens afins, ou seja, descobrir quais atributos aparecem com maior frequência ou que ocorram concomitantemente e de forma repetida em uma base de dados.	• Determinar que produto costumam ser colocados juntos em um carrinho de supermercado.
Classificação	Constrói um modelo de algum tipo que possa ser aplicado a dados não classificados a fim de categoriza-los em classes, o objetivo e descobrir um relacionamento entre um atributo meta (cujo valor será previsto) e um conjunto de atributos de previsão.	• Classificar pedidos de crédito; • Definir pedidos de seguros fraudulentos; • Identificar a melhor forma de tratamento de um paciente.
Estimativa	Usada para definir um valor para alguma variável contínua desconhecida.	• Estimar o número de filhos ou a renda total de uma família; • Estimar um valor em tempo de vida de um cliente; • Estimar a probabilidade de que um paciente morrerá baseando-se nos resultados de diagnósticos médicos.
Previsão	Associa-se a uma previsão de valor, baseando seu juízo nos dados históricos armazenados, para uma determinada variável. Ela prevê o comportamento dos registros que ainda serão inseridos em uma base de dados construindo modelos para avaliar os dados que já estão inseridos na base de dados.	• Determinar o comportamento de um cliente de uma empresa nos próximos três meses.
Clusterização	Processo de partição de uma população heterogênea em vários subgrupos ou grupos mais homogêneos.	• Agrupar clientes por região do país; • Agrupar clientes com comportamento de compra similar; • Agrupar seções de usuários Web para prever comportamento futuro de usuário.

Fonte: Adaptado de Dias (2001).

Além das tarefas de *data mining*, tem-se os métodos que auxiliam na execução destas. O método pode ser definido como algoritmos implementados em ferramentas de *data mining*, que têm como objetivo a descoberta de conhecimento em bases de dados (SCOSS, 2006).

Dentre os métodos de *data mining* tem-se:

a) **árvores de decisão**: o objetivo desse método é dividir um conjunto de dados, por meio de regras simples de decisão, até formar pequenos grupos de dados. Os elementos que compõem os conjuntos resultantes se tornam cada vez mais semelhantes entre si a cada divisão (BERRY; LINOFF, 2004, tradução nossa);

b) **redes neurais artificiais**: esse método tenta simular a forma como o cérebro humano realiza uma tarefa. Tem como objetivo calcular determinadas funções

por meio de uma rede neural artificial composta por unidades de processamento simples, denominados de neurônios artificiais. No início as redes neurais artificiais eram inspiradas somente no cérebro humano, porém mais tarde, foram acrescentados conceitos de estatística e processamento de sinais (BOTELHO, 2011; MOTTA, 2004);

c) **lógica *fuzzy***: permite que um elemento pertença a um ou mais conjuntos de dados utilizando graus de pertinência, para isso emprega um raciocínio aproximado de forma semelhante a capacidade humana de tratar imprecisões, diferenciando-se da lógica tradicional, onde os elementos pertencem ou não a um conjunto (REZENDE, 2005).

Tarefas e métodos de *data mining* podem ser implementados em ferramentas também conhecidas como *shells*. Essas *shells* em sua maioria são ferramentas comerciais, porém, o Grupo de Pesquisa em Inteligência Computacional Aplicada, do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense (UNESC), mantém em desenvolvimento o projeto de uma destas ferramentas, denominada de *Shell* Orion *Data Mining Engine*.

2.1.1 *Shell* Orion *Data Mining Engine*

Entre os principais objetivos do desenvolvimento do projeto da *Shell* Orion está a disponibilização de uma ferramenta gratuita de *data mining*. Esse projeto foi iniciado no ano de 2005 e todos os métodos implementados foram desenvolvidos por acadêmicos em seus Trabalhos de Conclusão de Curso (TCC). Na tabela 2 pode-se observar a evolução da ferramenta até o momento.

Tabela 2- Evolução da *Shell* Orion *Data Mining Engine*

Ano	Tarefa	Método	Algoritmo	Atributos	Referência
2005	Associação	Regra de Associação	Apriori	Numérico	(CASAGRANDE, 2005)
2005	Classificação	Árvore de Decisão	ID3	Nominais	(PELEGRIN, 2005)
2007	Classificação	Árvore de Decisão	CART	Nominais e Numéricos	(RAIMUNDO, 2007)
2007	Clusterização	Particionamento	K-means	Numéricos	(MARTINS, 2007)
2007	Clusterização	Redes Neurais	Kohonen	Numéricos	(BORTOLOTTO, 2007)
2008	Clusterização	Lógica Fuzzy	Gustafson-Kessel	Numéricos	(CASSETARI JUNIOR, 2008)
2009	Clusterização	Lógica Fuzzy	Gath-Geva	Numéricos	(PEREGO, 2009)
2009	Classificação	Árvore de Decisão	C4.5	Nominais e Numéricos	(MONDARDO, 2009)
2010	Classificação	Redes Neurais	RBF	Numéricos	(SCOTTI, 2010)
2010	Clusterização	Lógica Fuzzy	RCP	Numéricos	(CROTTI JUNIOR, 2010)
2010	Clusterização	Lógica Fuzzy	URCP	Numéricos	(CROTTI JUNIOR, 2010)
2010	Clusterização	Lógica Fuzzy	FCM	Numéricos	(CROTTI JUNIOR, 2010)
2011	Clusterização	Densidade	DBSCAN	Numéricos	(GAVA, 2011)

Fonte: Adaptado de Gava (2011).

O desenvolvimento da S*hell* Orion ocorre por meio do ambiente de programação Java, pois é gratuita, multiplataforma e permite a reutilização de código (PELEGRIN, 2005).

A plataforma Java possui a sua Interface de Programação de Aplicações (*Application Programming Interface* - API) denominada Java *Database Connectivity* (JDBC). Essa API torna a *Shell* Orion bastante flexível no que se refere a conexão com diversos SGBD, pois basta possuir um *driver* disponível para ela.

Desde 2005 vários TCC fizeram parte do projeto da *Shell* Orion, tendo-se dentre as tarefas implementadas a clusterização. O foco dessa pesquisa é desenvolver o algoritmo SACA na tarefa de clusterização dessa ferramenta, a fim de ampliar as suas funcionalidades.

3 A TAREFA DE CLUSTERIZAÇÃO EM *DATA MINING*

A tarefa de clusterização agrupa dados de um conjunto de elementos de forma que os grupos formados, denominados *clusters*, apresentem a maior similaridade possível com os dados do mesmo *cluster* (DIAS, 2004).

Assim, esta tarefa tem como principal objetivo dividir um conjunto de dados em grupos, de forma que os elementos de cada *cluster* compartilhem propriedades semelhantes diferenciando-os dos outros subconjuntos (GOLDSCHMIDT; PASSOS, 2005).

Ochi, Dias e Soares (2004) abordam que a distância entre dois dados é considerada como um importante critério para identificar sua similaridade, onde as diferenças dos valores de cada atributo são trabalhadas, ou seja, maior é a similaridade entre o par dos dados quanto menor for a distância entre eles.

A clusterização frequentemente é usada como a primeira tarefa no processo de extração de padrões, podendo o seu resultado servir de entrada para alguma outra tarefa. Mediante isso, a clusterização pode facilitar e melhorar o desempenho de outros algoritmos de *data mining* no processo de descoberta de conhecimento (BERRY; LINOFF, 2004, tradução nossa).

Existem diferentes métodos de clusterização disponíveis para utilização no processo de *data mining*, devendo-se levar em consideração o tipo de dado disponível para definir o método mais apropriado para determinado problema. Os métodos de clusterização podem ser classificados em (HAN; KAMBER, 2000, tradução nossa; JAIN; MURTY; FLYNN, 1999, tradução nossa):

a) **hierárquicos**: decompõe a base de dados em forma de árvore, fazendo divisões recursivas em conjunto de dados menores. Essa divisão pode ser feita de duas formas: *top-down* e *botton-up*. Na divisão *top-down* todos os objetos iniciam no mesmo *cluster*, este, no entanto, vai sendo dividido sucessivamente até que cada *cluster* contenha um único elemento. Já na divisão *botton-up* cada objeto é um *cluster*, assim a cada passo do procedimento, os dois *cluster* mais similares (próximos) são fundidos até que exista somente um grande *cluster* contendo todos os objetos (HAN; KAMBER, 2006, tradução nossa);

b) **baseado em densidade**: consideram grupos como sendo regiões densas de objetos no espaço de dados que são separados por regiões de baixa densidade, essas regiões geralmente são representadas como ruídos. Dessa forma, um *cluster* é uma região em que a densidade dos elementos ultrapassa certo limite,

ou seja, para cada elemento de um agrupamento, sua vizinhança, em um raio, deve conter uma quantidade mínima de elementos. Esse método permite descobrir grupos de formatos arbitrários (HAN; KAMBER, 2006, tradução nossa; PATERLINI, 2011);

c) **baseado em grade**: os objetos são divididos em um número finito de células que formam um estrutura de grade multidimensional, onde todas as operações de agrupamento são realizadas. Algoritmos baseados nessa heurística possuem suas operações dependentes do número de células da estrutura da grade e não do número de objetos da base de dados, o que melhora o seu desempenho (HAN; KAMBER, 2006, tradução nossa);

d) **baseado em modelo**: consiste na suposição de que os objetos são gerados a partir da união de distribuições probabilísticas, construindo-se modelos matemáticos. No aprendizado de máquina é usualmente referenciado como uma maneira não supervisionada de aquisição de conhecimento, já que a formação dos *clusters* é independente da existência de modelos (HAN; KAMBER, 2006, tradução nossa).

e) **particionamento**: divide uma base de dados em *k* grupos, o valor de *k* é escolhido pelo usuário. Primeiramente, os algoritmos determinam os *k* objetos que serão os centros dos grupos. A partir de então, os itens são colocados nos *clusters* que possuam maior similaridade, ou seja, cujo valor de distância entre o centro e o elemento for o menor possível, conforme a medida de similaridade empregada. Logo após, estes algoritmos utilizam uma estratégia iterativa, verificando se os objetos devem ou não mudar de grupo, para que os *clusters* possuam apenas elementos similares entre eles (GOLDSCHMIDT; PASSOS, 2005).

Considerando que o objetivo dessa pesquisa consiste na implementação de um algoritmo baseado em inteligência de enxame, a seguir é apresentado o conceito dessa área da inteligência computacional.

3.1 INTELIGÊNCIA DE ENXAMES

Agentes simples os quais sozinhos são incapazes de realizar tarefas simples, quando se integram formam um sistema auto organizado capaz de desenvolver tarefas de

grande complexidade. Essa descrição é um modelo inteligente inspirado em processos do mundo real, como a construção e estruturação de colméia pelas abelhas, a construção de sistemas de túneis complexos pelos cupins, as formigas que podem encontrar o melhor caminho para buscar os alimentos. Modelos computacionais interessantes foram desenvolvidos baseados nesse comportamento coletivo. Esses estudos levaram a criação de uma nova área na inteligência computacional, a inteligência de enxame (*swarm intelligence*) (LOPES, 2006).

Denomina-se inteligência de enxame a tentativa de desenvolvimento de algoritmos inspirados no comportamento coletivo de colônia de insetos sociais e outras sociedades de animais. Esses algoritmos são caracterizados pela interação com um grande número de agentes que percebem e modificam seu ambiente localmente (BONABEU; THERAULAZ; DORIGO, 1999, tradução nossa).

Pode-se exemplificar a inteligência de enxame por meio dos processos naturais de construção de ninhos pelos cupins ou sociedade de abelhas, atividade forrageadora das formigas, divisão de trabalho e alocação de tarefas nas colônias de formiga, transporte cooperativo de comida, seleção de fontes com melhor néctar pelas abelhas, entre outros. Comumente são encontradas nestes comportamentos as características destacadas a seguir (OMRAN; ENGELBRECHT; SALMAN, 2005, tradução nossa):

a) **flexibilidade**: a colônia consegue responder a desafios externos e também a perturbações internas;
b) **controle distribuído**: não existem indivíduos centralizados que direcionam a realização das tarefas em colônias. A organização das mesmas surge com as interações entre os constituintes do sistema, os indivíduos e o ambiente;
c) **robustez**: uma falha individual não compromete o funcionamento do sistema;
d) **auto-organização**: o caminho para resolver soluções são emergentes e não pré-definidos.

Algoritmos de inteligência de enxame são denominados assim por usarem agentes simples, ou seja, com baixa complexidade de implementação e de atuação, apresentarem iteração com o ambiente e sem controle centralizado. Esses agentes são capazes de formar padrões complexos interagindo com o ambiente de forma coletiva e chegando a solução de determinado problema (RUSSELL, C., 2004, tradução nossa).

Devido a essas características os algoritmos baseados em sistemas de inteligência de enxame se tornam atrativos do ponto de vista computacional, a implementação não apresenta uma complexidade elevada, pois se baseia em comportamentos e agentes

igualmente simples. Dessa forma, o desempenho e os recursos em geral tendem, em alguns casos, ser tão ou até mais apropriado que os de outros sistemas (OMRAN; ENGELBRECHT; SALMAN, 2005, tradução nossa).

Esse algoritmos apresentam vantagens e desvantagens comparados aos métodos mais tradicionais de busca e otimização (COELHO, 2003). Pode-se destacar como vantagem:

a) a aplicação em propósito geral, assim é vastamente aplicável;
b) desempenho e aplicação de baixo custo;
c) outros métodos podem ser incorporados facilmente.

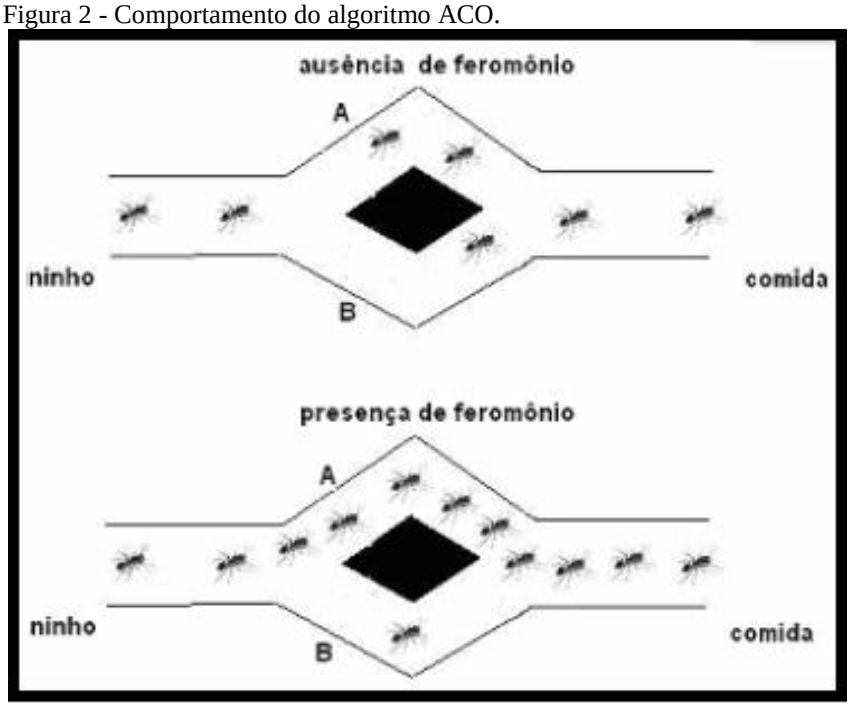
Dentre as desvantagens tem-se:

a) não tem uma garantia de solução ótima;
b) alguns parâmetros necessitam de sintonias dependendo da metodologia de comportamento coletivo adotado;
c) o seu desempenho e resultados variam a cada execução.

Usando heurísticas baseadas no comportamento coletivo, muitas soluções de diferentes problemas têm sido encontradas. O algoritmo *Ant Colony Optimization* (ACO) é um exemplo de uma meta-heurística que usa formigas artificiais para resolver problemas de otimização. Esse algoritmo se originou da observação de colônias de formigas que coletivamente encontram a menor rota entre uma fonte de alimento e seu ninho, para isso formam trilhas de feromônio[5] (DORIGO; DI CARO; GAMBARDELLA, 1999, tradução nossa). Na figura 2 é possível observar o seu comportamento. Entre os exemplos de aplicação do algoritmo ACO estão: roteamento de veículos e de redes de comunicação; coloração de grafos, entre outros.

[5] É uma substancia química, identificadas por animais de mesma espécie (VIZINE et al, 2005).

Figura 2 - Comportamento do algoritmo ACO.



Fonte: Adaptado de Vizine et al (2005, p 33).

A inteligência de enxame também pode oferecer soluções interessantes no processo de agrupamento de dados. Inspirados no ACO outros algoritmos foram propostos para esse problema, podendo-se citar:

a) ***shelokar***: esse algoritmo não tem nome definido por isso nesse trabalho é chamado de *shelokar*. Foi proposto por Shelokar, Jayaraman e Kulkarni (2004). Utiliza trilhas de feromônio para guiar as formigas fazendo com que elas selecionem um grupo para cada objeto. Os dados são inspecionados um de cada vez pelas formigas, de forma sequencial, e um grupo é selecionado para um objeto considerando a concentração de feromônio (SHELOKAR; JAYARAMAN; KULKARNI, 2004, tradução nossa);

b) **ACOC**: o algoritmo *Ant Colony Optimization for Clustering* (ACOC), é basicamente uma extensão do algoritmo proposto por Shelokar, Jayaraman e Kulkarni (2004). A diferença entre os dois é que o ACOC não avalia somente a quantidade de feromônio, ele também leva em consideração a distancia entre os objetos e o centro de uma formiga. Isso faz com que o ACOC tenha um desempenho melhor em relação a qualidade das soluções produzidas (JOHNSON, 2003, tradução nossa).

Além do ACO, outro comportamento das formigas como a formação de cemitérios e organização da ninhada inspiraram a implementação de algoritmos de agrupamento.

Deneubourg et al (1991) observaram que a espécie de formiga *Leptothorax unifasciatus*[6] organiza os ovos em pilhas próximos das larvas, porém afastadas dos casulos, ou ainda os ovos, larvas e casulo são colocadas em locais completamente diferentes do ninho (figura 3). Os itens da ninhada são arrumados em forma de anéis concêntricos. No centro dos anéis são colocados os ovos e micro larvas de tamanho padrão, os demais conforme aumentam o seu tamanho são colocados cada vez mais afastados do centro. Em outras palavras, as pupas[7] são distribuídas em posições externas entre os anéis das larvas correspondentes aos seus tamanhos (FRANKS; SEDOVA, 1992, tradução nossa; HARTMANN, 2005, tradução nossa).

Figura 3 - Organização da ninhada da espécie *Leptothorax unifasciatus*.

Fonte: Adaptado de Deneubourg et al (1991, p 360)

Diferentes espécies de formigas organizam seus corpos mortos em um comportamento reconhecido como formação de cemitérios. As formigas operárias carregam os corpos mortos para fora do ninho, onde são depositadas em montes. O principal

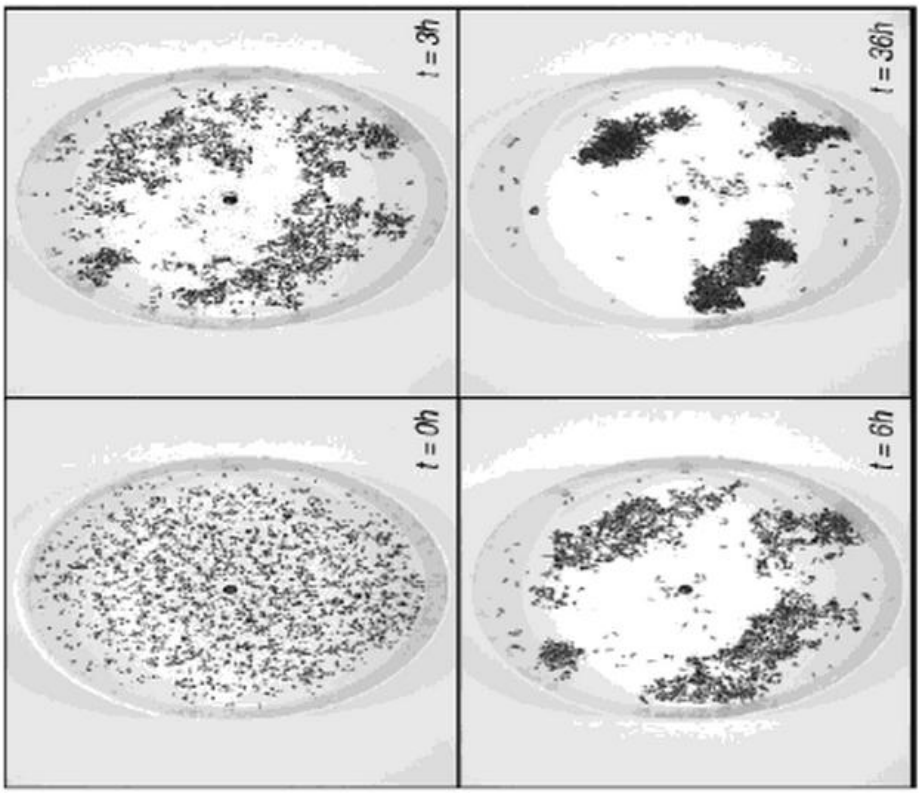
[6] Pertencem a um gênero que é notável por seu parasitismo social generalizada, ou seja, eles são dependentes da ajuda de trabalhadores de outras espécies de formigas durante uma parte ou a totalidade de seus ciclos de vida (BUSCHINGER; SCHULZ, 2008).

[7] Terceiro estado do desenvolvimento do ciclo de vida de um inseto (BUSCHINGER; SCHULZ, 2008).

mecanismo desse comportamento é a atração entre os corpos já depositados e as operárias que estão carregando outros corpos. É por meio desse mecanismo de *feedback* que pequenos grupos vão se formando e aumentando de tamanho (HARTMANN, 2005, tradução nossa).

Esse comportamento foi observado por Deneubourg et al (1991) na espécie de formiga *Pheidole pallidula* (figura 4) e o inspirou sobre um modelo de agrupamento por um grupo de agentes homogêneos. Nesse modelo os agentes (formigas) se movem aleatoriamente, e quando um agente descobre um objeto é verificada a probabilidade de ele pegar esse objeto, essa probabilidade aumenta se o objeto estiver isolado. Do mesmo modo, a probabilidade de deixar o objeto aumenta se ela estiver em uma área contendo outros objetos similares. Portando, a base do modelo é a verificação de duas probabilidades, uma de pegar e outra de deixar o objeto. Deneubourg et al (1991) denominou esse algoritmo como *Ant Clustering Algorithm* (ACA) (HANDL, 2003, tradução nossa).

Figura 4 - Organização do cemitério da espécie *Pheidole pallidula*.



Fonte: Adaptado de Vizine et al (2005, p 34).

O algoritmo ACA foi implementado em uma pesquisa que utilizava robôs para realizar tarefas de organização de objetos em grupos por semelhanças, baseando-se no comportamento de organização de cemitérios das colônias de formigas (DENEUBOURG et al, 1991, tradução nossa).

Muitos trabalhos de computação e robótica com coletividade de agentes que realizam o agrupamento ou organização de objetos se inspiraram nessa pesquisa. Beckers, Holland e Deneubourg (1994), realizaram um experimento que utilizava robôs para agrupamento, os quais não possuíam memória, porém os autores observaram que os robôs tinham um comportamento que levava ao agrupamento de objetos.

Deneubourg et al (1991) definiu a probabilidade de um agente pegar ou deixar um objeto calculando as seguintes funções:

$$p_{pegar} = \left(\frac{k_+}{k_+ + f} \right)^2 \quad (1)$$

$$p_{deixar} = \left(\frac{f}{k_- + f} \right)^2 \quad (2)$$

Onde *f* é a fração de objetos percebido na vizinhança do agente, pode-se dizer que é a visão que o agente possui. k_- e k_+ são constantes. Quando *f* é menor que $k+$, então p_{pegar} é próximo de 1, dessa forma, a probabilidade de pegar um objeto é alta quando não há muitos outros objetos na vizinhança, p_{pegar} é próximo de 0 quando *f* for maior que $k+$, e assim, torna difícil a remoção dos objetos de locais onde há concentração de objetos.

Utilizando essa regra grupos de objetos semelhantes são formados pelos agentes. Na robótica *f* foi definida como número *N* de objetos encontrados durante as últimas *T* unidades de tempo dividido pelo maior número possível de objetos que podem ser encontrados durante *T*. Para utilizar esse modelo como uma ferramenta de agrupamento de dados é necessários definir melhor *f*.

Lumer e Faieta (1994) realizaram modificações no modelo ACA que permitiu a manipulação de dados numéricos e melhorou a qualidade da solução e o tempo da convergência do algoritmo. No algoritmo ACA os objetos são similares se são idênticos e dissimilares se são diferentes, ou seja, não existem objetos parecidos e sim objetos iguais ou diferentes. Lumer e Faita (1994) definiram uma medida de similaridade ou dissimilaridade entre os objetos. Os autores denominaram esse novo modelo como *Standard Ant Clustering Algorithm* (SACA).

3.1.1 *Standard Ant Clustering Algorithm* (SACA)

O algoritmo SACA foi apresentado por Lumer e Faieta (1994). Esse modelo foi inspirado no modelo ACA proposto por Deneubourn (1991).

No algoritmo ACA, os agentes são espalhados aleatoriamente sobre uma grade bidimensional. Eles não se comunicam uns com os outros e só percebem o ambiente onde estão posicionados. Quando se deparam com um objeto, a probabilidade de pegar o objeto diminui com a densidade dos outros objetos que estão próximos. Já a probabilidade de deixar um objeto transportado aumenta com a densidade dos outros objetos próximos.

Este comportamento simples se traduz na organização de objetos idênticos. Deneubourg et al (1991) restrito os seus estudos em ambientes feitos de objetos idênticos ou dois tipos de objetos distintos. No entanto esse algoritmo pode ignorar objetos que tenham uma diferença mínima e assim não pode ser aplicado em base de dados multidimensional. (LUMER; FAIETA, 1994, tradução nossa).

Segundo Lumer e Faieta (1994), o algoritmo SACA assim como no algoritmo ACA, as formigas e os objetos (conjunto de dados) são espalhados em uma grade bidimensional. A cada iteração uma formiga é selecionada aleatoriamente e pode pegar ou deixar um objeto no seu local atual uma vez que, respectivamente: à um objeto naquele local, ou que a formiga esteja carregando um objeto e o lugar esta vazio. Supondo que o agente encontrou um objeto, a probabilidade de a formiga pegar esse objeto diminui com a similaridade dos demais objetos posicionados ao seu redor (No que se segue essa área é definida como um quadrado de d x d). Assim a probabilidade de escolher um objeto *i* é definida como:

$$p_p(i) = \left(\frac{k_p}{k_p + f(i)} \right)^2 \tag{3}$$

Onde k_p é uma constante e *f(i)* uma estimativa local da densidade de objetos e suas semelhanças com *i*. Da mesma forma, a probabilidade de uma formiga largar um objeto deve aumentar com a densidade de objetos semelhantes com o objeto que ela está carregando. A função que satisfaz esta tendência, sugerida por Lumer e Faita (1994), é dada por:

$$p_d(i) = \begin{cases} 2f(i) & se \quad f(i) < k_d \\ 1 & em \quad outros \quad casos \end{cases} \tag{4}$$

Onde k_d é uma constante. A função de similaridade que foi definida pelos autores como função vizinhança é dada por:

$$f(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j \left(1 - \frac{(\delta(i,j)}{\alpha}\right) & se \quad f(i) > 0 \\ 0 & em \quad outros \quad casos \end{cases} \tag{5}$$

A dissimilaridade entre os itens $\delta(i, j)[0,1]$, quantifica a distância entre os dois dados em seu espaço original N-dimensional. A métrica usada, no caso, é a distância euclidiana, dada por:

$$\delta(i,j) = \sqrt{\left|x_{i1} - x_{j1}\right|^2 + \left|x_{i2} + x_{j2}\right|^2 + ... + \left|x_{in} - x_{jn}\right|^2} \tag{6}$$

a) $i = (x_{i1}, x_{i2}, ..., x_{in})$, e $j = (x_{j1}, x_{j2}, ..., x_{jn})$ são os valores dos atributos de dois dados N-dimensionais;
b) $\alpha \in [0,1]$, é um parâmetro escalar que estabelece a dependência em relação aos dados;
c) σ^2, é o tamanho da vizinhança local (quantidade de células da grade que o agente consegue perceber, onde o agente é o centro). Este valor depende do raio de percepção do agente em cada direção, que, consequentemente, é dado por $\frac{\sigma - 1}{2}$.

A função vizinhança *f(i)* é levada em consideração no cálculo das probabilidades da formiga deixar ou pegar o objeto. A densidade de objetos próximos, bem como suas similaridades é computada no cálculo, as constantes k_p e k_d foram definidas como 0.1 e 0.3 respectivamente por Lumer e Faita (1994).

Além da função da vizinhança que permitiu que as formigas identificassem objetos ao seu redor, Lumer e Faieta (1994) introduziram ainda no algoritmo SACA os seguintes conceitos:

a) **memória de curta duração**: com essa alteração a formiga consegue lembrar os últimos itens que foram movimentados, assim como seus respectivos locais onde foram deixados. Após uma formiga pegar um objeto, ela leva em consideração os objetos anteriores transportados para comparar com o objeto atual. Isto permite que a formiga movimente o objeto atual para o local onde foi deixado o objeto com o qual mais se assemelha. Esse alteração acelera significativamente o agrupamento, quando um padrão é carregado, a posição *best matchining* (melhor combinação) memorizada, que é a posição de mínima dissimilaridade, será usada para indicar o sentido da direção aleatória da formiga;

b) **população heterogênea de agentes**: essa alteração foi limitada ao tamanho do passo que uma formiga pode dar no seu movimento aleatório. Passo significa quantidade de célula da grade bidimensional que a formiga é capaz de caminhar em um movimento. Com esse ajuste, existirá formigas mais rápidas e que podem realizar passos maiores na grade, dessas forma aumentando a velocidade de agrupamento do algoritmo.

O algoritmo SACA pode ser dividido em duas fases Handl (2003):

a) fase inicial:

- todos os objetos que representam os índices dos dados são espalhados aleatoriamente na grade bidimensional toroidal. Cada objeto ocupa uma célula,
- cada formiga pega um objeto aleatoriamente,
- cada formiga ocupa uma posição da grade, que é a mesma do objeto que pegou;

b) fase de agrupamento:

- uma formiga é selecionada aleatoriamente,
- a formiga se movimenta executando um passo, de acordo com o valor de passos definido, carregando o objeto, e assumindo uma nova posição na grade,
- a formiga decide, verificando a vizinhança em que ele esta, se deixará o objeto em sua nova posição.

Se a decisão for sim, ele deixa o objeto na célula em que ela está situada, e procura aleatoriamente um objeto que não esta sendo carregado por nenhuma outra formiga, ou seja, livre na grade. Ao selecionar o objeto livre, a formiga

decide probabilisticamente, considerando a posição do deste, se vai pega-lo. Caso decida não pegar, o agente seleciona outro objeto e faz a avaliação novamente. Este processo continua até a formiga pegar um objeto,

- volta-se ao passo inicial, e repete-se o processo para outras formigas, por um número de iterações pré-estabelecido.

Na figura 5 tem-se o pseudocódigo apresentado por SACA:

Figura 5 – Pseudocódigo do algoritmo SACA

```
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
```

Fonte: Adaptado de Handl (2003).

Com o passar do tempo o algoritmo SACA sofreu alterações de aperfeiçoamento por outros pesquisadores como Handl (2003), Vizine et al (2005) e Sherafat, Castro e Hruschka (2004). Essa pesquisa esta direcionada ao algoritmo SACA original proposto por Lumer e Faieta (1994) e aos aperfeiçoamentos propostos por Handl (2003) e ao algoritmo adaptativo proposto por Vizine et al (2005).

3.1.2 *Ant Based Clustering*

Handl (2003) verificou a existência de dois grandes problemas no algoritmo SACA proposto por Lumer e Faieta (1994). Um destes problemas era devido ao resultado apresentado pelo algoritmo: ele não gera partições, mas sim um distribuição dos elementos em uma representação gráfica. Apesar de o resultado ser óbvio para a observação humana, a avaliação do resultado requer uma interação humana. Outro problema identificado pela autora

foi a dificuldade de se ajustar os diversos parâmetros do algoritmo para diferentes tipos de dados.

Visando aperfeiçoar o algoritmo, a autora introduziu as seguintes alterações: adaptação da função vizinhança, memória curta dos agentes, raio de percepção crescente, separação espacial, vizinhança ponderada, modificação na probabilidade de pegar e deixar um item e adaptação de α

3.1.2.1 Adaptação da função vizinhança

As funções propostas por Lumer e Faieta (1994) que calcula a probabilidade de pegar ou largar um objeto continuam as mesmas. A autora continuou usando os valores estipulados para k_p e k_d, respectivamente 0.1 e 0.3. Porém, a função vizinhança *f(i)*, dado por (5), foi substituída pela seguinte função *f*(i)* (HANDL, 2003):

$$f^*(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j \left(1 - \frac{(\delta(i,j)}{\alpha}\right) & se \quad f^*(i) > 0 \wedge \forall j \left(1 - \frac{(\delta(i,j)}{\alpha}\right) > 0 \\ 0 \quad em \quad outros \quad casos \end{cases} \qquad (7)$$

Segundo Handl (2003), esta nova definição da função vizinhança combina duas importantes propriedades. Primeiramente, assim como na função original, a divisão pelo tamanho da vizinhança σ^2 penaliza as células vazias, permitindo a indução a um agrupamento compacto. Em segundo lugar, a restrição adicional $\forall j \left(1 - \frac{\delta(i,j)}{\alpha}\right) > 0$ serve para penalizar severamente altas dissimilaridades entre os objetos.

3.1.2.2 Memória curta dos agentes

Como citado na seção 3.1.1 a idéia de memória de curto prazo já havia sido abordada por Lumer e Faieta (1994). Nesse caso, cada agente se lembra dos últimos dados carregados e de suas posições da grade onde foram deixados. Quando um novo dado é pego, a posição do melhor dado recentemente pego (aquele com menor dissimilaridade) é utilizada para guiar o movimento randômico do agente.

A autora implementou o seguinte solução: Uma formiga, localizada numa célula da grade *p*, e carregando um item *i*, usa sua memória para percorrer todas as posições memorizadas, uma após outra. Essas posições são avaliada utilizando a função vizinhança *f*(i)*, e verificação da dissimilaridade de cada uma para o dado correntemente com o agente é examinada. A posição com melhor resultado de *f*(i)* pode fazer com que o agente migre para ela, de acordo com uma probabilidade de deixar calculada. Se isso não ocorrer o agente continua seu movimento aleatório.

3.1.2.3 Raio de percepção crescente

A quantidade de célula percebida pela formiga é determinado pelo campo de visão da mesma, essas células são informações utilizadas no processo de agrupamento. Com isso, é interessante informar um valor grande para o campo de visão a fim de melhorar a qualidade de agrupamento na grade. Porém, isso acarreta em um custo computacional elevado e impede a formação de grupos na fase inicial do processo.

Portanto, Handl (2003) tradução nossa, propôs uma modificação no raio de percepção fazendo com que ele aumente com o tempo. Essa modificação economiza custo computacional na primeira parte do processo, e facilita a formação de grupos no início. A autora utilizou em sua implementação um campo de visão inicial de 1 *($\sigma = 3$)* com um incremento linear até alcançar 5 *($\sigma = 11$)*.

3.1.2.4 Separação espacial

A separação espacial dos grupos na grade é muito importante para que eles sejam bem formados. Considerando que os grupos são formados em qualquer lugar da grade onde haja maior densidade de dados similares. Apos a sua formação inicial, os grupos tendem a se movimentarem muito lentamente pela grade (HANDL, 2003, tradução nossa).

Handl (2003) utilizou, após a primeira fase, durante um intervalo de iterações, entre os tempos *Tinicial* e *Tfinal*, uma modificação na função vizinhança. O parâmetro $\frac{1}{\sigma^2}$ foi substituído por $\frac{1}{\text{Nocc}}$, onde *Nocc* é o número de células ocupadas na vizinhança em que a formiga está, os valores *Tinicial* e *Tfinal* são determinados por *0.45 * N* e *0.55 * N* respectivamente, onde *N* é o total de iterações. Com isso a similaridade passa a ser levada em

consideração e não a densidade. Com isso os dados são espalhados novamente na grada. Isto causa o efeito de espalhar os dados novamente na grade, porém de uma forma organizada. Terminado esse período a função vizinhança volta a considerar a densidade da vizinhança. Dessa forma, os grupos são formados novamente, porém mais próximos dos centros das regiões.

3.1.2.5 Vizinhança ponderada

Handl (2003) estudou a utilização de uma ponderação no valor que cada célula contribui para o cálculo final da função vizinhança, com o objetivo de fazer com que células próximas ajudassem de forma mais assídua no calculo da vizinhança.

Nesse trabalho não foi implementado a função vizinhança ponderada devido ao alto custo computacional que ela apresenta e também pelo fato de não apresentar melhoras significativas na qualidade do agrupamento (LAURO, 2008).

3.1.2.6 Modificação na probabilidade de pegar e deixar um item

Handl (2003) modificou as funções para cálculo das probabilidades proposta por Lumer e Faieta (1994).

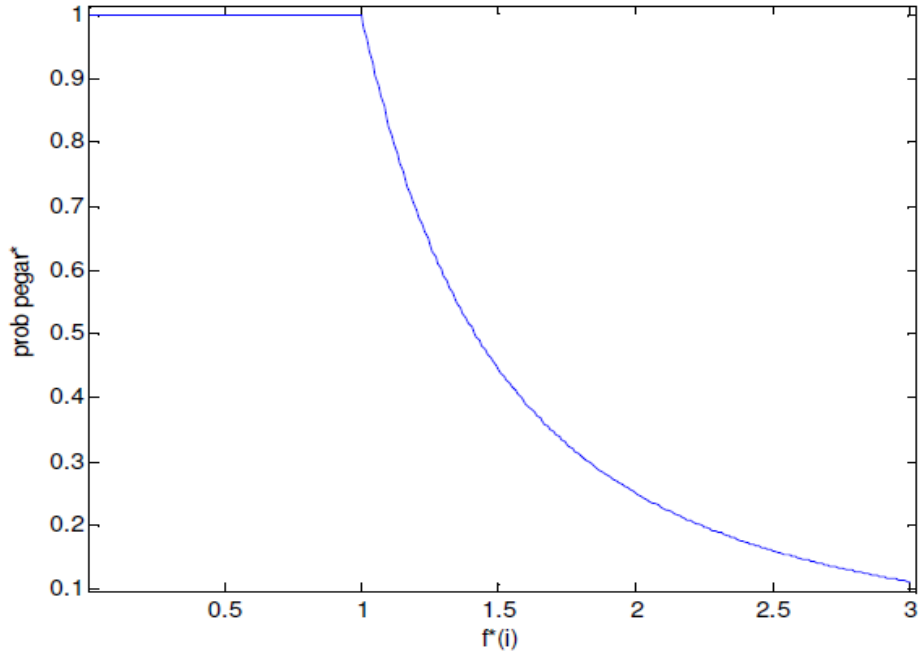
A probabilidade de se pegar o item, é determinada por:

$$p^*_{pegar}(i) = \begin{cases} 1.0 & se \quad f^*(i) \leq 1.0 \\ \dfrac{1}{f^*(i)^2} & em \quad outros \quad casos \end{cases} \tag{8}$$

E a probabilidade de deixar o item é:

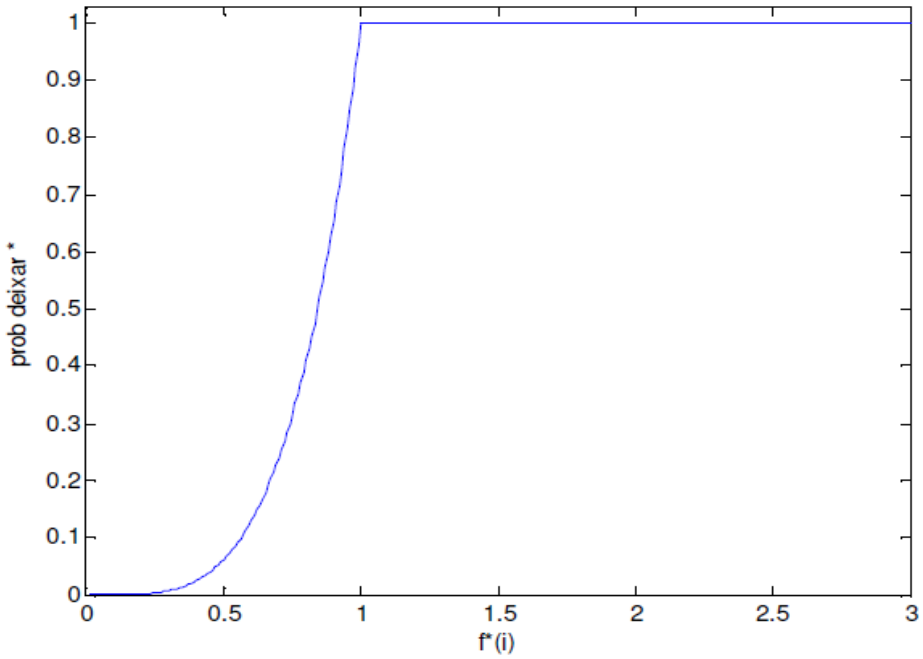
$$p^*_{deixar}(i) = \begin{cases} 1.0 & se \quad f^*(i) \geq 1.0 \\ f^*(i)^4 & em \quad outros \quad casos \end{cases} \tag{9}$$

Figura 6 - Comportamento da probabilidade modificada de pegar.



Fonte: Adaptado de Handl (2003)

Figura 7 - Comportamento da probabilidade modificada de deixar



Fonte: Adaptado de Handl (2003)

Essas funções se diferem das originais pelo fato de não utilizarem os parâmetros k_p e k_d que exigem certa complexidade para serem ajustados. Essas modificações contribuíram para a diminuição de parâmetros de entrada do algoritmo (HANDL, 2003, tradução nossa).

Observando as novas equações, pode-se identificar que a estratégia de se deixar o objeto torna-se mais cuidadosa que a versão definida por Lumer e Faieta (1994). O calculo da probabilidade em regiões que possuam objetos de baixa similaridade torna-se menor, e, em regiões que possuam objetos de alta similaridade torna-se maior. A probabilidade de se deixar o objeto torna-se determinístico ($p^*_{deixar} = 1$) para regiões onde a função vizinhança é maior ou igual a 1, assim, somente para essa região a operação de pegar o objeto é válida.

Conforme Hand (2003), essa modificações não são aplicáveis ao algoritmo básico proposto por Deneubourg et al (1991). Torna-se viável apenas se algumas das seguintes modificações

Essa modificação das equações de probabilidade, no entanto, conforme descrito pela autora, não é aplicável para o algoritmo básico, proposto por Deneubourg et al (1991). Se torna viável se implementado com algumas das seguintes alterações: aumento do raio de percepção, memória dos agentes e o pequeno intervalo da iteração com a função vizinhança modificada.

3.1.2.7 Adaptação de α

Handl (2003) destaca que o parâmetro α é importante na formação dos grupos, pois determina a dissimilaridade dentro da função vizinhança. Portanto, o seu correto ajuste é fundamental para o desempenho do algoritmo.

Durante o processo que o agente verifica a similaridade da vizinhança, α vai determinar a porcentagem de itens da base de dados que serão considerados similares: um valor de α muito pequeno pode inibir a formação de grupos na grade; já um valor excessivamente elevado pode resultar na fusão de grupos distintos. No pior caso, todos os itens formariam um único grupo (HANDL, 2003).

A escolha adequada do valor de α depende, no entanto, da dissimilaridade entre os dados que serão agrupados, e por isso, não deve ser escolhido sem uma prévia avaliação dos mesmos. Por esta razão, Handl (2003) propôs um ajuste automático do parâmetro, monitorando-se as intensidades das atividades dos agentes por meio do registro do sucesso e insucesso das operações de deixar e pegar os dados

Para isso, o seguinte esquema de adaptação foi estabelecido: gera-se uma população heterogênea de agentes, cada um com seu parâmetro α, gerado randomicamente dentro do intervalo [0,1]. Cada agente terá seu parâmetro corrigido após cada *Nefetivos*

movimentos. Durante esse tempo, será registrado o número de operações de deixar o dado que falharam *Nfalhas*. A razão de falhas será determinada como $rfalhas = \frac{Nfalhas}{Nefetivos}$, onde *Nefetivos* é fixado em 100.

Então o ajuste se dá seguindo-se a seguinte regra:

$$\alpha \longleftarrow \begin{cases} \alpha + 0.01 & se \quad r_{falhas} > 0.99 \\ \alpha - 0.01 & se \quad r_{falhas} \leq 0.99 \end{cases} \tag{10}$$

Essa adaptação torna o algoritmo bem robusto além de permitir que o α se altere em diferentes fases do processo (HANDL, 2003, tradução nossa).

3.2 *ADAPTIVE ANT-CLUSTERING ALGORITHM* (A²CA)

Segundo Vizine et al (2005), uma das dificuldades de se aplicar o algoritmo SACA em problemas complexos é devido ao fato do algoritmo gerar um número excessivo de grupos que não correspondem ao número efetivo. Além disso, esse algoritmo não estabiliza numa solução particular, pois durante as iterações os grupos são criados e destruídos.

A fim de superar a dificuldade citada, Vizine et al (2005) propuseram o algoritmo adaptativo, *Adaptive Ant-Clustering Algorithm* (A²CA), esse algoritmo é mais confiável na formação de número de grupos e melhora o seu resultado longo do processo.

Para isso, foram introduzidas no algoritmo original, SACA de Lumer e Faieta (1994), duas alterações principais:

a) uma rotina de adaptação do valor do parâmetro k_p;

b) um campo de visão que se adapta aos grupos formado durante o processo do algoritmo;

c) utilização de uma heurística de feromônio que é um valor informado em cada célula da grade, fazendo com que aumente a probabilidade da formiga deixar um objeto em áreas de alta densidade.

Com essas modificações o algoritmo se torna adaptativo ao processo e dessa forma os resultados do agrupamento são melhores.

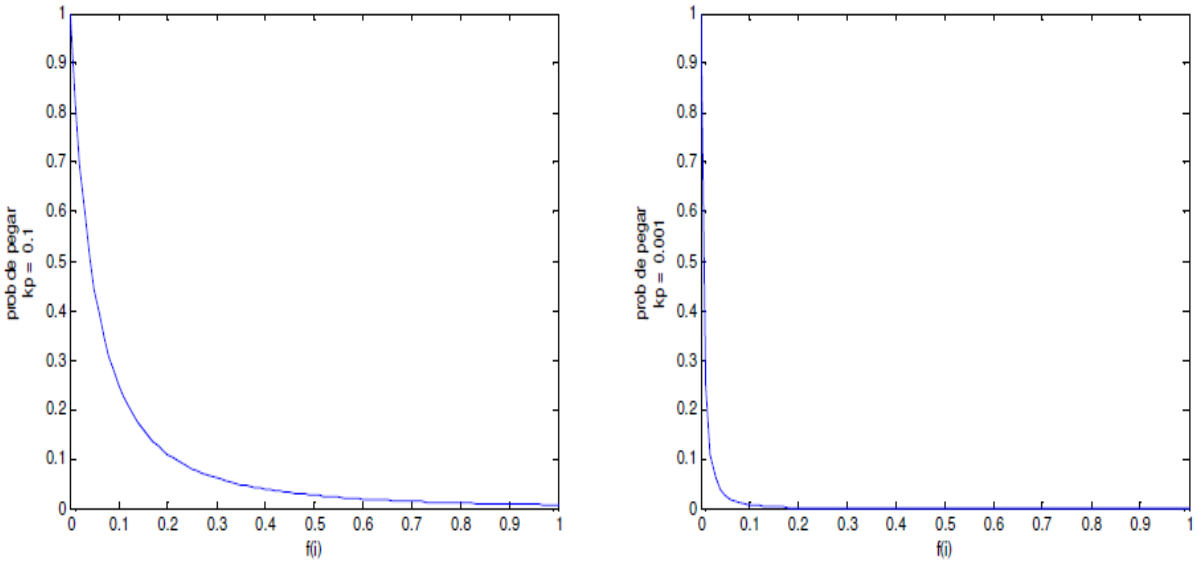
3.2.1 Resfriamento Gradativo de kp

Vizine et al (2005), introduziram no SACA uma alteração que tem o objetivo de minimizar o problema da instabilidade do algoritmo em manter os grupos já formados conforme vão passando as iterações. Essa modificação é uma lógica que resfria o parâmetro k_p, este influencia diretamente na probabilidade de a formiga pegar um item. Com essa, lógica o valor de k_p decresce até um valor mínimo permitido no decorrer das iterações.

Segundo Vizine et al (2005) adaptação de k_p acontece da seguinte forma: após um ciclo de 10.000 iterações, o valor de k_p decresce, a cada ciclo, até um valor mínimo $k_{p\min}$, que também é um critério de parada do algoritmo. O autor define k_p da seguinte forma:

$$\begin{cases} k_p \leftarrow \ k_p * 0.98, \\ k_{p\min} = 0.001. \end{cases} \tag{12}$$

Figura 8 - Probabilidades de pegar o item para kp=0.1 e kp=0.001.



Fonte: Adaptado de Vizine et al (2005).

3.2.2 Visão adaptativa

Segundo Vizine et al (2005), existe um problema enquanto ao parâmetro campo de visão σ no algoritmo SACA. Neste algoritmo esse valor é fixo para cada formiga correndo-se o risco de criar vizinhanças indesejáveis, isso por que um valor fixo para σ formiga não consegue diferenciar grupos grande e de grupos pequenos. Se o valor de σ for alto, no inicio do processo dificulta a formação de grupos além de aumentar o custo computacional do algoritmo. Se for um valor muito baixo, grupos pequenos e próximos um do outro são formados, pois a formiga não consegue visualizar uma área maior e identifica que próximo a ela à um grupo formado.

Sherafat, Castro e Hruschka (2004), definiram um campo de visão que se adapta a formação de grupos durante o processo do algoritmo. Essa adaptação acontece quando uma formiga detecta um grupo grande, ela aumenta o seu campo de visão para um limite pré-definido e recalcula a função vizinhança. Dessa forma o risco de formar grupos pequenos ou retirar um elemento que já esta em um grupo grade diminui. Para que a formiga possa detectar o tamanho do grupo que ela está verificando Vizine et al (2005) utilizaram a função vizinhança, essa função esta associada a média de valores da vizinhança, dessa forma os autores definiram que se o valor da função vizinhança atingir o valor θ definido como 0.6, o campo de visão da formiga é aumentado em 2 unidades até o valor 7, que é o valor máximo do campo de visão da formiga definido. Esse processo pode-se definir como:

$$\text{Se } f(i) > \theta \text{ e } \sigma^2 \leq \sigma^2 max,$$
$$\text{então } \sigma^2 \leftarrow \sigma^2 + ns$$
$$\text{onde } \sigma^2 max = 7 \text{ x } 7 \text{ e } \theta = 0.6$$

3.2.3 Heurística do feromônio

Para realizar o processo de agrupamento do SACA, o algoritmo trabalha com a distancia entre todos os dados que estão no campo de visão da formiga. O problema dessa abordagem segundo Vizine et al (2005), é que esse comportamento não leva em consideração para a tarefa o progresso do agrupamento em um nível global. Sherafat, Castro e Hruschka (2004) introduziram uma variável local $\phi(i)$ associada com cada posição i da grade que indica ausência ou presença de feromônio na célula i. Assim, segundo Vizine et al (2005), as formigas irão adicionar uma quantidade de feromônio nos objetos que estão carregando e esse

feromônio será transferido para a célula quando o objeto for depositado. Esse feromônio é evaporado a cada iteração.

A função feromônio $Phe(\phi_{\max}, \phi_{\min}, P, \phi(i))$ dada pela equação (13) foi introduzida por Sherafat, Castro e Hruschka (2004), essa função, conforme Vizine et al (2005), influencia a probabilidade de pegar e deixar o objeto na grade. A função de feromônio proposta varia linearmente com o nível de feromônio de cada posição da grade, $\phi(i)$, e depende de alguns parâmetros pré-definidos, tais como os valores de feromônio percebidos pelo agente, $\phi_{\max}$ e $\phi_{\min}$, e a máxima influência de feromônio permitida, *P*.

$$Phe(.) = \frac{2.P}{\phi_{mac} - \phi_{\min}}\phi(i) - \frac{2.P\phi_{\max}}{\phi_{\max} - \phi_{\min}} + P \qquad (13)$$

Sherafat, Castro e Hruschka (2004) modificaram as funções de probabilidade de pegar e deixar um objeto, do SACA propostas por Lumer e Faieta (1991), fazendo que seja considerado a adição de feromônio:

$$P_{pegar}(i) = (1 - Phe(\phi_{\min}, \phi_{\max}, P, \phi(i)))x\left(\frac{k_p}{k_p + f(i)}\right)^2 \qquad (14)$$

$$P_{pegar}(i) = (1 + Phe(\phi_{\min}, \phi_{\max}, P, \phi(i)))x\left(\frac{f(i)}{k_d + f(i)}\right)^2 \qquad (15)$$

Onde $\phi_{\max}$ representa a máxima quantidade de feromônio percebida pela formiga correntemente, $\phi_{\min}$ representa a menor quantidade de feromônio percebida pela formiga, *P* é a máxima influência do feromônio em alterar as probabilidades de pegar ou deixar o objeto, e $\phi(i)$ é a quantidade de feromônio na posição i Vizine et al (2005), Castro, Hruschka e Gudwin (2005) propôs a substituição das funções (14) e (15) pelas equações (16) e (17) a fim de reduzir o número de parâmetros a serem definidos:

$$P_{pegar}(i) = \frac{1}{f(i)\phi(i)}\left(\frac{k_p}{k_p + f(i)}\right)^2 \qquad (16)$$

$$P_{deixar}(i) = f(i)\phi(i)\left(\frac{f(i)}{k_d + f(i)}\right)^2 \quad (17)$$

Pode-se observar que nessa proposta, o único parâmetro novo, introduzido, em relação ao SACA é o nível de feromônio, $\phi(i)$, em cada posição da grade.

Conforme Vizine et al (2005), de acordo com a equação (16), a probabilidade de uma formiga pegar um objeto da grade é inversamente proporcional à quantidade de feromônio naquela posição e à densidade de objetos em torno de *i*. Essa equação também leva em consideração o reforço do nível de feromônio em áreas com objetos similares. Em regiões com objetos dissimilares, a incorporação de *f(i)* multiplicando $\phi(i)$ contrabalança o efeito de eventual alta concentração de feromônio. Seguindo o mesmo raciocínio, a equação (17) indica que regiões com alta concentração de feromônio são atrativas para o depósito de outros objetos similares.

Vizine et al (2005) ainda alerta que é importante observar que regiões com altas concentrações de feromônio tendem a ser tanto um grupo recentemente formado ou um grupo em formação. O feromônio é uma função discreta da grade. Isto é, cada posição da grade *i* possui uma função independente $\phi(i)$, para a qual um procedimento de evaporação e difusão é implementado. O feromônio é evaporado pela razão definida como:

$$\phi(i) \leftarrow \phi(i) * 0.99 \quad (18)$$

Além da evaporação Vizine et al (2005) acrescentaram que cada célula da grade *i* também possui uma conexão com as células vizinhas, responsável por uma percentagem de difusão de feromônio. Isto é feito de forma que a percentagem de feromônio difundido na célula que se distancia da célula contígua, em qualquer direção decai geometricamente na razão $\frac{1}{2}$, de forma que à terceira célula em relação à contígua, em qualquer direção, a quantidade de feromônio difundida será nula. Na implementação do autor, o valor máximo de quantidade de feromônio somado foi considerado 0.01. A abordagem proposta aumenta a probabilidade de se desfazer os relativamente pequenos grupos, e aumenta a probabilidade de se deixar objetos em grupos densos. Isto é diretamente influenciado pela similaridade entre o dado e o *cluster*.

3.3 TRABALHOS CORRELATOS

O interesse em desenvolver um algoritmo baseado no comportamento de espécies da natureza não é uma prática recente, várias pesquisas já foram desenvolvidas e encontram-se em desenvolvimento.

O algoritmo SACA é constantemente estudado e já sofreu diversas modificações a fim de melhorar seu desempenho sem que seu núcleo de funcionamento altere, ou seja, a proposta inicial foi mantida, apenas a maneira dos cálculos das probabilidades foram alteradas, com o objetivo de encontrar uma solução com melhores resultados.

3.3.1 *Ant Clustering Algorithms*

Artigo realizado por Urszula Boryczka do Instituto de Ciência da Computação da Universidade de Silesia, situado em Sosnowiec na Polônia, publicado no *Intelligent Information Systems* em 2008.

O autor implementa o algoritmo ACA proposto por Handl (2003). Na pesquisa todas as modificações sugeridas por Handl(2003) são descritas e apresentadas. O objetivo da pesquisa é comparar o desempenho do algoritmo ACA com o algoritmo K-*means*. A fim de realizar testes de performance o autor submete os algoritmos citados a algumas métricas de avaliação: índice randômico, medida F, índice de *Dunn* e variância.

Os resultados do ACA são muito próximos aos do K-*means* sobre os conjuntos de dados analisados. O autor ainda alerta que, diferente do K-*means*, para o algoritmo ACA não foi fornecido o número correto de clusters. A pesquisa centrou-se em estudar o esquema de adaptação de α que criar problemas para algoritmos ACA, pois ele é fundamental para a formação dos grupos (BORYCZKA, 2008, tradução nossa).

A pesquisa cita que o algoritmo de agrupamento formiga tem uma série de características que o tornam um forte candidato. Em primeiro lugar, devido ao seu comportamento escala linear, é atraente para utilização em grandes conjuntos de dados, por exemplo em info-recuperação de sistemas de produção. Em segundo lugar este algoritmo lida com os *outliers* dentro dos conjuntos de dados (BORYCZKA, 2008, tradução nossa).

3.3.2 Técnicas de Agrupamento e de Hierarquização no Contexto e KDD – Aplicação a Dados Temporais de Instrumentação Geotécnica-Estrutural da Usina Hidrelétrica de Itaipu

Essa tese de doutorado em Métodos Numéricos em Engenharia foi desenvolvida por Rosangela Villwock no ano de 2009, sendo apresentado a Universidade Federal do Paraná – UFPR.

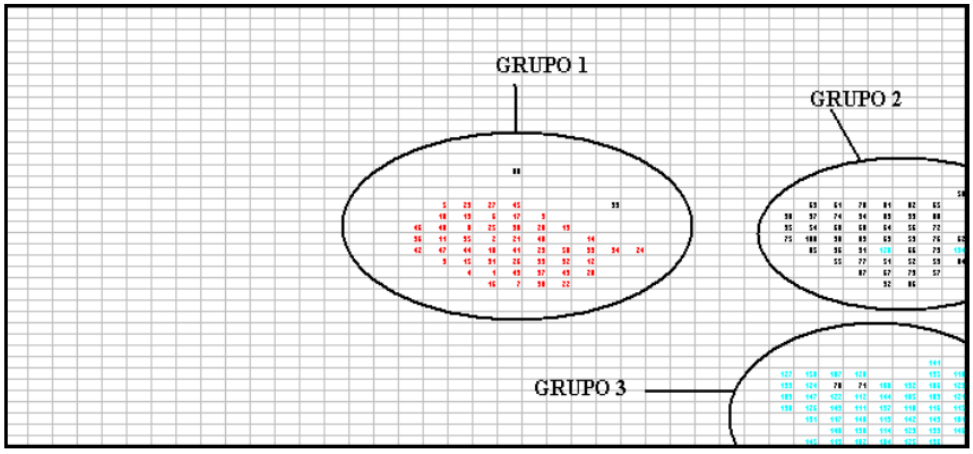
Nessa pesquisa algoritmo SACA foi submetido a algumas modificações como: a introdução de uma comparação da probabilidade de descarregar um padrão na posição escolhida aleatoriamente com a probabilidade de descarregar este padrão em sua posição atual; a introdução de uma avaliação da probabilidade de uma posição vizinha, quando a decisão de descarregar um padrão for positiva e a célula em que o padrão deveria ser descarregado estiver ocupada; e a substituição do padrão carregado por uma formiga, caso este padrão não seja descarregado em 100 iterações consecutivas (VILLWOCK; STEINER, 2009).

O objetivo da pesquisa é apresentar uma metodologia, enquadrada na área de *KDD*, Descoberta de Conhecimento em Bases de Dados, com o intuito de realizar a hierarquização de instrumentos de monitoramento de barragens, maximizando a eficácia e eficiência das análises das leituras, através da identificação de grupos de instrumentos semelhantes e, também, detectando os principais instrumentos (VILLWOCK; STEINER, 2009).

Em relação ao algoritmo proposto, este foi testado em três bases de dados reais (IRIS, WINE e PIMA Indians Diabetes) e em duas bases de dados reais de séries temporais (GUN e LIGHTNING-2), sendo que o seu desempenho foi comparado com o de outros dois métodos (Método Ward e Redes Neurais de Kohonen). Na aplicação da Análise de grupamento (pelo Método Ward) aos dados de instrumentação geotécnica-estrutural da Itaipu, mostrou-se que é possível encontrar justificativas técnicas para a formação dos grupos, inclusive identificando um grupo de hastes de maior importância. Já a aplicação da Análise Fatorial aos referidos dados, mostrou-se bastante eficaz para realizar a hierarquização das hastes de extensômetros, com base nas comunalidades (VILLWOCK; STEINER, 2009).

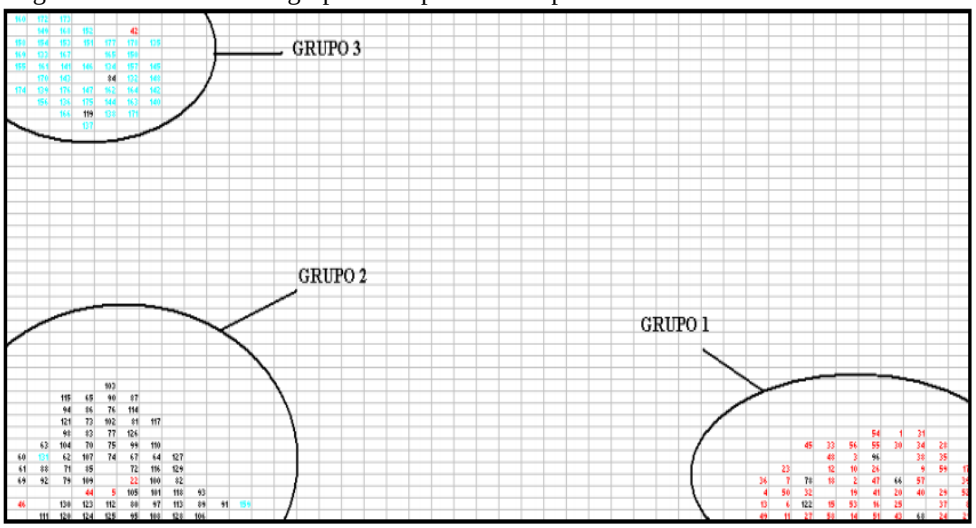
O algoritmo proposto apresentou resultados satisfatórios em relação aos resultados de Boryczka (2008) para as bases de dados reais e, quando aplicado aos dados de instrumentação geotécnica-estrutural da Itaipu, o mesmo foi capaz de identificar o grupo de hastes de maior importância (VILLWOCK; STEINER, 2009).

Figura 9 - Resultado do agrupamento para o exemplo IRIS – melhor resultado.



Fonte: Adaptado de Villwock e Steiner (2009).

Figura 10- Resultado do agrupamento para o exemplo WINE – melhor resultado.



Fonte: Adaptado de Villwock e Steiner (2009).

3.3.3 Agrupamento de Dados Utilizando Algoritmo de Colônia de Formigas

Dissertação de mestrado em Ciência em Engenharia Química realizada por André Luís Lauro apresentado ao Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (COPPE) da Universidade Federal do Rio de Janeiro no ano de 2008.

O autor faz uma abordagem sobre a origem do algoritmo SACA e suas principais modificações sugeridas por Handl(2003): Adaptação da função vizinhança, memória dos agentes, raio de percepção crescente, separação espacial, vizinhança ponderada, modificação nos cálculos das probabilidades originais e adaptação de α. Também implementou as principais modificações sugeridas por Vizine et al (2005): Modificação nos cálculos das probabilidades originais, adaptação de kp a cada 10.000 iterações e visão progressiva do agente. O trabalho tem o objetivo de esmiuçar o algoritmo e apresentá-lo de uma forma bem

clara, de forma a possibilitar a avaliação de sua evolução, dinâmica e desempenho (LAURO, 2008).

O algoritmo SACA foi implementado no software MATLAB, para a realização dos experimentos, foram escolhidas algumas bases de dados, e os parâmetros e as variações implementadas do algoritmo foram testados para cada caso. Para tentar desvendar o comportamento do algoritmo, alguns parâmetros e funções foram monitoradas ao longo das iterações (LAURO, 2008).

Cada alteração gera consequências diferentes nos resultados. Foram feitos muitos testes, sendo que nem todos estão relatados no trabalho. Se assim fosse feito, o trabalho se tornaria exaustivamente longo, e com uma quantidade de informações muito esparsa. Foram escolhidas, então, de acordo com a experimentação realizada, a utilização daquelas alterações que trouxeram melhores resultados para os casos estudados (LAURO, 2008).

O autor conclui que o SACA possui uma grande vantagem com relação a maioria dos outros algoritmos de agrupamento. Ele não precisa da informação inicial do número de grupos da base de dados a ser trabalhada. Apresenta, no entanto, o problema de ajuste de outros parâmetros que não são triviais. Algumas propostas de alterações relatadas já resolveram alguns desses problemas (LAURO, 2008).

Dentre as propostas comentadas que apresentaram melhor resultados nos experimentos realizados no trabalho, pode-se citar a rotina de adaptação de α, que é um parâmetro utilizado na "função densidade", e que é de fundamental importância para o bom desempenho do algoritmo, pois essa função, além de identificar as áreas mais "povoadas", deve reconhecer, também, a similaridade entre seus objetos. A nítida convergência deste parâmetro, quando se utiliza a rotina supracitada, demonstra o acerto da mesma (LAURO, 2008).

4 O ALGORITMO SACA NA TAREFA DE CLUSTERIZAÇÃO DA *SHELL* ORION *DATA MINING ENGINE*

A Shell Orion é mantida pelo Grupo de Pesquisa em Inteligência Computacional Aplicada da UNESC, é um projeto acadêmico que tem como objetivo disponibilizar uma ferramenta gratuita para auxilio ao processo de descoberta de conhecimento em bases de dados.

Dessa forma, essa pesquisa tem como objetivo aumentar as funcionalidades da *Shell* Orion, desenvolvendo os algoritmos SACA, *Ant Based Clustering* e A²CA para a tarefa de clusterização. O desempenho e a qualidade dos algoritmos desenvolvidos na ferramenta são avaliados utilizando uma base de dados, que foi escolhida nessa pesquisa, da área da ambiental, nesta á informações sobre a qualidade das águas dos rios da região carbonífera catarinense.

4.1 BASE DE DADOS

A base de dados utilizada foi a mesma utilizada pelo Bacharel em Ciência da Computação Éverton Marangoni Gava, em 2011, na apresentação do algoritmo DBSCAN, esta base possui indicadores ambientais da qualidade dos recursos hídricos das três bacias hidrográficas da região sul de Santa Catarina: Araranguá, Tubarão e Urussanga.

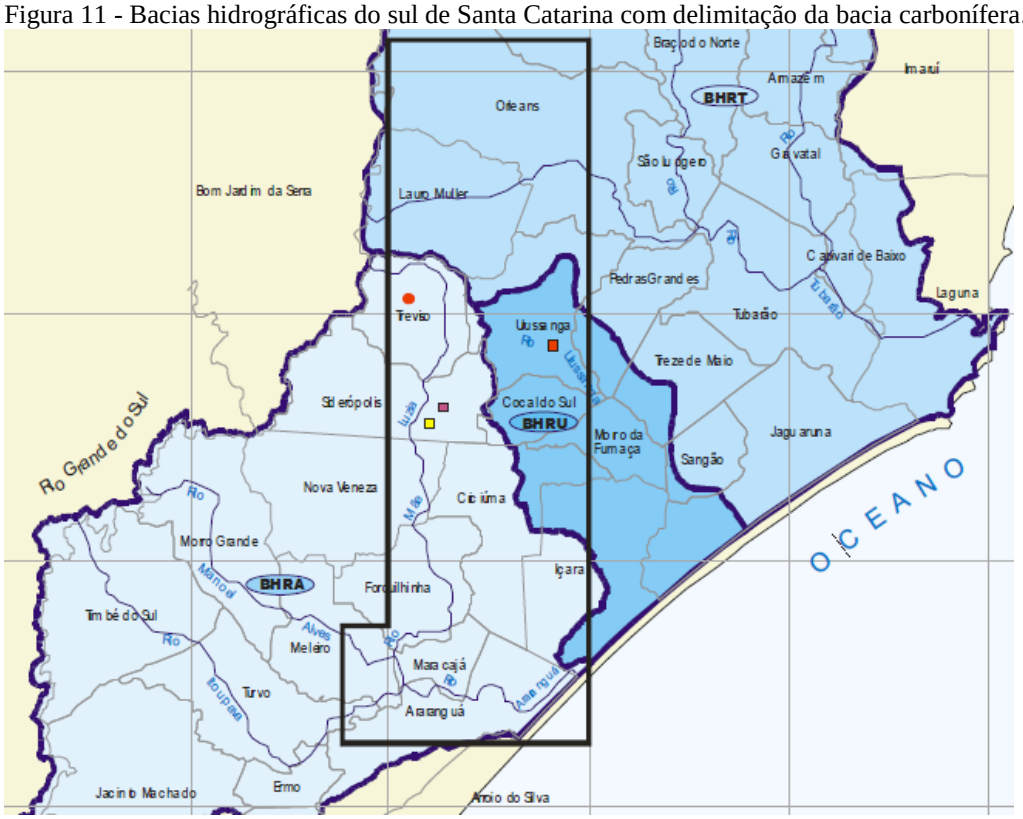
Segundo GTA (2009), mineração de carvão em Santa Catarina se desenvolveu, em seu período inicial, de 1895 a 1945, com a produção de carvão para fins energéticos e, a partir de 1945, para suprir a produção nacional de aço. O choque do petróleo, na década de 1970, fez aumentar o consumo do carvão energético que passou, graças aos subsídios governamentais aplicados a produção, ao consumo e ao transporte, a ser utilizado no pais inteiro como energético, substituindo o óleo combustível. Naquele período, o carvão passou a ser explorado em grandes minas, ampliando os problemas de contaminação decorrentes da disposição descontrolada de rochas da cobertura das camadas das minas a céu aberto e dos rejeitos do beneficiamento do carvão. Essas rochas e rejeitos contem minerais sulfetados (pirita e marcassita) que provocaram a acidificação e a contaminação dos recursos hídricos.

A base de dado utilizada nesse trabalho pertence ao terceiro relatório de monitoramento de indicadores ambientais, apresentado no ano de 2009 pelo Grupo Técnico de Assessoramento (GTA). Possui 1313 registros, cada registro contém 20 atributos onde constam diversos indicadores de monitoramento dos recursos hídricos superficiais das três

bacias hidrográficas da região sul catarinense. O monitoramento é realizado de forma sistemática por meio de análises físico-química de agua e de medidas de vazão em 140 pontos de monitoramento nas bacias hidrográficas dos rios Araranguá, Urussanga e Tubarão, distribuídos estrategicamente nas áreas impactantes pela mineração de carvão (GTA, 2009).

Pode-se observar na figura 11 as três básicas hidrográficas do sul de Santa Catarina junto com as delimitações da bacia carbonífera.

Figura 11 - Bacias hidrográficas do sul de Santa Catarina com delimitação da bacia carbonífera.



Fonte: Adaptado de GTA (2009).

Considerando os 20 atributos que estão contidos na base de dados, 19 deles são numéricos e um se refere a data da coleta das informações, essa data está entre março de 2002 e abril de 2009. Não existem valores ausentes entre os atributos dos registros, a base de dados se encontra normalizada. As características de cada atributo estão descrita na tabela 3 junto com suas considerações técnicas extraídas do relatório de monitoramento dos indicadores ambientais (GTA, 2009).

Tabela 3 - Base de dados de análise dos recursos hídricos.

Atributo	Descrição	Valor
Id	Atributo identificador do registro.	Número inteiro
id_cab	Identificador do ponto de monitoramento.	Número inteiro de 1 até 140
id_bacia	Atributo identificador da bacia hidrográfica onde o registro foi coletado.	1 para Araranguá, 2 para Tubarão e 3 para Urussanga
Campanha	Identificador da campanha de monitoramento.	Número inteiro de 1 até 20
Data	Atributo que faz referência a data de coleta do registro.	Data
Ph	Grandeza físico-química conhecida como "potencial hidrogeniônico". É um valor entre 0 e 14 que indica se uma solução qualquer é ácida ($pH < 7$), neutra ($pH = 7$), ou alcalina ($pH > 7$). A faixa para o *pH* recomendável está entre 6 a 9.	Número decimal
Vazão	Volume de água ou um fluido qualquer que passa, em uma determinada unidade de tempo, por meio de uma superfície.	Número decimal
Acidez	Quantidade de ácido necessária para titular uma amostra a um determinado *Ph*	Número decimal
Cond	Condutividade elétrica da água, ou seja, capacidade da água conduzir corrente elétrica.	Número decimal
so, al, fe, mn	Respectivamente representam as concentrações de sódio, alumínio, ferro e manganês que estão presentes nas águas analisadas.	Número decimal
c_acidez	Carga de acidez. Calculada pela multiplicação da vazão pela concentração de acidez. Possui relação direta com a carga de contaminantes presentes na água.	Número decimal
c_fe, c_al, c_so, c_mn	Respectivamente representam as cargas de ferro, alumínio, sódio e manganês presentes nas águas.	Número decimal
Precip	Precipitação regional (chuva) obtidos em estações meteorológicas das empresas carboníferas da região.	Número decimal
id_estacao	Atributo identificador da estação de monitoramento.	Número inteiro de 1 a 14

Fonte: (GAVA, 2011).

Os registros estão subdivididos entre as bacias hidrográficas da região sul de Santa Catarina, 679 registros são referente a o Rio Araranguá, 425 ao Rio Urussanga e 209 ao Rio Tubarão.

4.2 METODOLOGIA

As seguintes etapas metodológicas foram empregadas no desenvolvimento: levantamento bibliográfico; modelagem por meio do padrão UML; demonstração matemática do algoritmo SACA; implementação e realização de testes; análise de desempenho e validação dos resultados obtidos.

Foi levado em consideração no levantamento bibliográfico a fundamentação e o entendimento de todos os temas envolvidos na pesquisa, tais como o processo de descoberta de conhecimento em base de dados, *data mining*, inteligência de enxame, colônia de formiga,

o algoritmo SACA e seus aperfeiçoamentos, índices de avaliação de desempenho e qualidade para algoritmo de clusterização de dados.

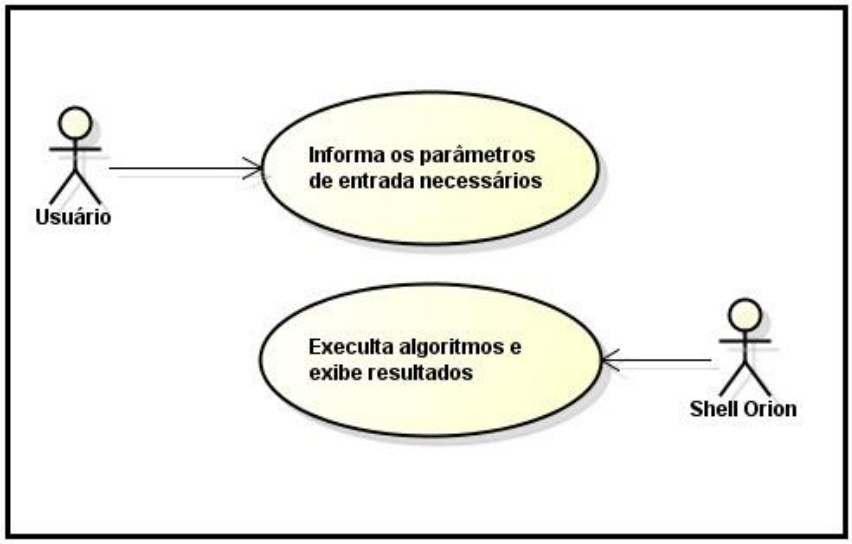
4.2.1 Modelagem do Módulo do Algoritmo SACA

O primeiro passo para o desenvolvimento foi a modelagem dos processos realizados pelo algoritmo SACA por meio do padrão *Unified Modeling Language* (UML) com o objetivo de facilitar a compreensão dos processos do algoritmo. Foram desenvolvidos os diagramas de caso de uso, sequência e atividades utilizando a ferramenta *Astah Community.*

Os diagramas de caso de uso apresentam as ações realizadas pelo usuário e pelo sistema.

a) **informar os parâmetros de entrada do algoritmo**: o usuário deve informar os parâmetros necessários para a execução do algoritmo, como: o número de formigas, total de iterações, o valor mínimo para pegar o objeto (k_p), o valor mínimo para deixar o objeto (k_d), o valor de alfa α que ira distinguir para mais ou para menos os grupos e o campo de visão da formiga que segundo Lauro (2008) geralmente são: 3, 5 ou 7;
b) **execução do algoritmo:** o algoritmo SACA é executado após a definição dos parâmetros entrada.

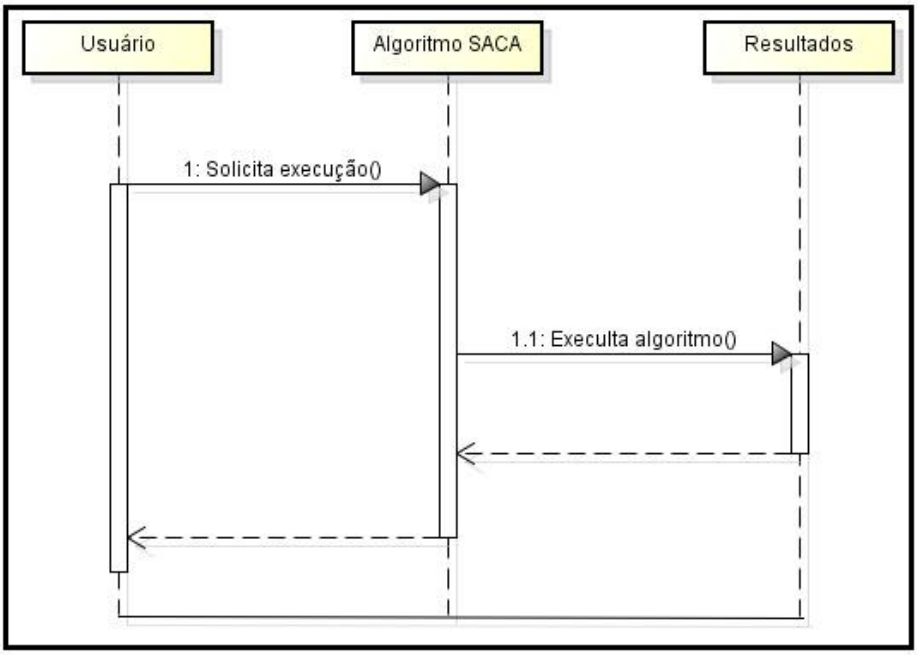
Figura 12 - Diagrama de casos de uso.



Fonte: Do autor.

O diagrama de sequencia tem o objetivo de mostrar interações entre objetos organizada em uma sequencia de tempo e de mensagens trocadas. As mensagens trocadas se definem baseando-se na documentação dos casos de uso (FURLAN, 1998).

Figura 13 – Diagrama de sequencia.

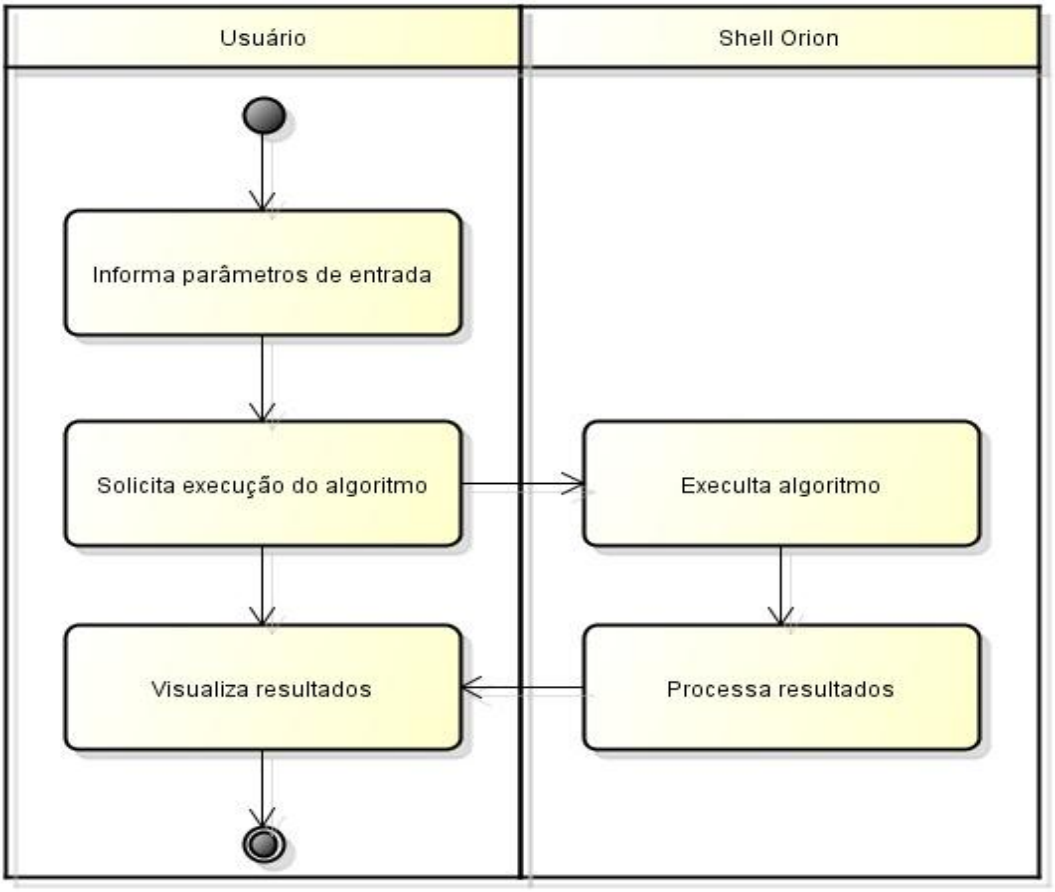


Fonte: Do autor.

A figura 13 demonstra o diagrama de sequência do algoritmo. O usuário solicita a interface inicial do algoritmo SACA, onde são informados os parâmetros necessários, após isto tem-se início a execução do algoritmo que gera os resultados a serem apresentados ao usuário.

Posteriormente modelou-se o diagrama de atividades do algoritmo SACA. Segundo Bezerra (2007), diagrama de atividades é um tipo de diagrama de estado que representa os estados de uma atividade, em vez dos estados de um objeto. Esses diagramas são orientados a fluxos de controle.

Figura 14 - Diagrama de atividades.



Fonte: Do autor.

Pela figura 14 pode-se verificar em um único processo o fluxo de tarefas realizado pelo usuário na utilização do sistema. O usuário informa os parâmetros de entrada necessários para execução do algoritmo, logo depois solicita ao sistema a execução do algoritmo, o sistema realiza a execução do algoritmo e gera os resultados da execução, a apresentação dos resultados obtidos é apresentada em gráfico e relatório.

A modelagem por meio da UML, contribui nessa pesquisa, para o entendimento do sistema desenvolvido, além de auxiliar na documentação do funcionamento do algoritmo SACA na Shell Orion *Data Mining Engine*

4.2.2 Demonstração Matemática do Algoritmo SACA, *Ant Based Clustering* e A²CA

Nesta etapa da pesquisa são demonstrados os algoritmos SACA, *Ant Based Clustering* e A²CA, por meio da modelagem matemática, permitindo a compreensão do seu funcionamento.

Para o algoritmo SACA, os conceitos foram baseados no artigo escrito por Erik D.

Lumer e Baldo Faieta em 1994, com o título *Diversity and Adptation in Population of Clustering Ant.* Já a modelagem do algoritmo *Ant Based Clustering* baseou-se no artigo *Ant-based methods for tasks of clustering and topographic mapping: extensions, analysis and comparison with alternative techniques* escrito por J. Hand em 2003, e para o A²CA o artigo escrito por André L. Vizine, Leandro N. de Castro, Eduardo R. Hruschka e Ricardo R Gudwin em 2005 com o título *Toward Improving Clustering Ants: An Adaptive Ant Clustering Algorithm*.

Nesta demonstração matemática foi utilizada uma base de dados contendo 5 elementos (x_1, x_2, x_3, x_4, x_5,) com 5 atributos cada um (*a, b , c, d, e*). A tabela 4 exemplifica os valores utilizados:

Tabela 4 - Base de dados utilizada na modelagem do algoritmo.

Atributo	X1	X2	X3	X4	X5
a	0.43	0.18	0.58	0.09	0.39
b	0.66	0.03	0.18	0.83	0.69
c	0.13	0.27	0.11	0.33	0.71
d	0.12	0.32	0.13	0.29	0.81
e	0.19	0.72	0.05	0.62	0.67

Fonte: Do autor.

Com a base de dados definida, o algoritmo SACA, *Ant Based Clustering* e A²CA devem construir a matriz de dissimilaridade, que é montada calculando a distancia euclidiana n-dimensional e normalizando a mesma para que seus valores fiquem entre 0 e 1. A distancia euclidiana é definida como a distancia entre pares de indivíduos em um plano cartesiano (CLIFFORD; STEPHENSON, 1975). A matriz de dissimilaridade representa a distancia entre pares de objetos calculados pela distancia euclidiana. Essa matriz é quadrada e de tamanho *n* x *n,* que é definido pela quantidade de objetos a ser clusterizado. A figura 15 apresenta uma matriz de dissimilaridade (HAN; KAMBER, 2006, tradução nossa):

Figura 15 – Matriz de dissimilaridade.

$$\begin{bmatrix} 0 & & & & \\ dist(2,1) & 0 & & & \\ dist(3,1) & dist(3,2) & 0 & & \\ : & : & : & 0 & \\ dist(n,1) & dist(n,2) & .... & .... & 0 \end{bmatrix}$$

Fonte: Adaptado de Han e Kamber (2006).

Onde *dist*(*i*,*j*) representa a distancia entre os objetos *i* e *j*. Quanto mais próximo de zero for o valor de *dist*(*i*,*j*), mais similares são os objetos (HAN; KAMBER, 2006, tradução nossa).

A matriz de dissimilaridade foi montada calculando a distancia euclidiana n-dimensional conforme a formula:

$$\delta(i,\, j) = \sqrt{\left|x_{i1} - x_{j1}\right|^2 + \left|x_{i2} + x_{j2}\right|^2 + ... + \left|x_{in} - x_{jn}\right|^2}$$

$$dist(x_1, x_1) = \sqrt{(0{,}43-0{,}43)^2 + (0{,}18-0{,}18)^2 + (0{,}58-0{,}58)^2 + (0{,}09-0{,}09)^2 + (0{,}39-0{,}39)^2} =$$
$$dist(x_1, x_1) = \sqrt{0+0+0+0+0} = 0$$

$$dist(x_1, x_2) = \sqrt{(0{,}43-0{,}66)^2 + (0{,}18-0{,}03)^2 + (0{,}58-0{,}18)^2 + (0{,}09-0{,}83)^2 + (0{,}39-0{,}69)^2} =$$
$$dist(x_1, x_2) = \sqrt{0{,}0529 + 0{,}0225 + 0{,}16 + 0{,}5476 + 0{,}09} = 0{,}9343$$

$$dist(x_1, x_3) = \sqrt{(0{,}43-0{,}13)^2 + (0{,}18-0{,}27)^2 + (0{,}58-0{,}11)^2 + (0{,}09-0{,}33)^2 + (0{,}39-0{,}71)^2} =$$
$$dist(x_1, x_3) = \sqrt{0{,}09 + 0{,}0081 + 0{,}2209 + 0{,}0576 + 0{,}1024} = 0{,}6921$$

$$dist(x_1, x_4) = \sqrt{(0{,}43-0{,}12)^2 + (0{,}18-0{,}32)^2 + (0{,}58-0{,}13)^2 + (0{,}09-0{,}29)^2 + (0{,}39-0{,}81)^2} =$$
$$dist(x_1, x_4) = \sqrt{0{,}0961 + 0{,}0196 + 0{,}2025 + 0{,}04 + 0{,}1764} = 0{,}7312$$

$$dist(x_1, x_5) = \sqrt{(0{,}14-0{,}19)^2 + (0{,}18-0{,}72)^2 + (0{,}58-0{,}05)^2 + (0{,}09-0{,}62)^2 + (0{,}39-0{,}67)^2} =$$
$$dist(x_1, x_5) = \sqrt{0{,}0576 + 0{,}2916 + 0{,}2809 + 0{,}2809 + 0{,}0784} = 0{,}9947$$

$$dist(x_2, x_2) = \sqrt{(0{,}66-0{,}66)^2 + (0{,}18-0{,}18)^2 + (0{,}58-0{,}58)^2 + (0{,}09-0{,}09)^2 + (0{,}39-0{,}39)^2} =$$
$$dist(x_2, x_2) = \sqrt{0+0+0+0+0} = 0$$

$$dist(x_2, x_3) = \sqrt{(0{,}66-0{,}13)^2 + (0{,}03-0{,}27)^2 + (0{,}18-0{,}11)^2 + (0{,}83-0{,}33)^2 + (0{,}69-0{,}71)^2} =$$
$$dist(x_2, x_3) = \sqrt{0{,}2809 + 0{,}0576 + 0{,}0049 + 0{,}25 + 0{,}0004} = 0{,}7706$$

$$dist(x_2, x_4) = \sqrt{(0{,}66-0{,}12)^2 + (0{,}03-0.32)^2 + (0{,}18-0{,}13)^2 + (0{,}83-0{,}29)^2 + (0{,}69-0{,}81)^2} =$$
$$dist(x_2, x_4) = \sqrt{0{,}2916 + 0{,}0841 + 0{,}0025 + 0{,}2916 + 0{,}0144} = 0{,}8272$$

$$dist(x_2, x_5) = \sqrt{(0{,}66-0{,}19)^2 + (0{,}03-0{,}72)^2 + (0{,}18-0{,}05)^2 + (0{,}83-0{,}62)^2 + (0{,}69-0{,}67)^2} =$$
$$dist(x_2, x_5) = \sqrt{0{,}2209 + 0{,}4761 + 0{,}0169 + 0{,}0441 + 0{,}0004} = 0{,}8709$$

$$dist(x_3, x_3) = \sqrt{(0{,}13-0{,}13)^2 + (0{,}27-0{,}27)^2 + (0{,}11-0{,}11)^2 + (0{,}33-0{,}33)^2 + (0{,}71-0{,}71)^2} =$$
$$dist(x_3, x_3) = \sqrt{0+0+0+0+0} = 0$$

$$dist(x_3, x_4) = \sqrt{(0{,}13-0{,}12)^2 + (0{,}27-0{,}32)^2 + (0{,}11-0{,}13)^2 + (0{,}33-0{,}29)^2 + (0{,}71-0{,}81)^2} =$$
$$dist(x_3, x_4) = \sqrt{0{,}0001+0{,}0025+0{,}0004+0{,}0016+0{,}01} = 0{,}1208$$

$$dist(x_3, x_5) = \sqrt{(0{,}13-0{,}19)^2 + (0{,}27-0{,}72)^2 + (0{,}11-0{,}05)^2 + (0{,}33-0{,}62)^2 + (0{,}71-0{,}67)^2} =$$
$$dist(x_3, x_5) = \sqrt{0{,}0036+0{,}2025+0{,}0036+0{,}0841+0{,}0016} = 0{,}5435$$

$$dist(x_4, x_4) = \sqrt{(0{,}12-0{,}12)^2 + (0{,}32-0{,}32)^2 + (0{,}13-0{,}13)^2 + (0{,}29-0{,}29)^2 + (0{,}81-0{,}81)^2} =$$
$$dist(x_4, x_4) = \sqrt{0+0+0+0+0} = 0$$

$$dist(x_4, x_5) = \sqrt{(0{,}12-0{,}19)^2 + (0{,}32-0{,}72)^2 + (0{,}13-0{,}05)^2 + (0{,}29-0{,}62)^2 + (0{,}81-0{,}67)^2} =$$
$$dist(x_4, x_5) = \sqrt{0{,}0049+0{,}16+0{,}0064+0{,}1089+0{,}0196} = 0{,}5475$$

Com todas as distâncias calculadas a matriz de dissimilaridade é montada:

$$\begin{bmatrix} 0 & 0{,}9343 & 0{,}6921 & 0{,}7312 & 0{,}9947 \\ 0{,}9343 & 0 & 0{,}7706 & 0{,}8272 & 0{,}8709 \\ 0{,}6921 & 0{,}7706 & 0 & 0{,}1208 & 0{,}5435 \\ 0{,}7312 & 0{,}8272 & 0{,}1208 & 0 & 0{,}5475 \\ 0{,}9947 & 0{,}8709 & 0{,}5435 & 0{,}5475 & 0 \end{bmatrix}$$

É preciso agora normalizar a matriz para que os valores fiquem entre 0 e 1, mesmo se os valores já estiverem entre 0 e 1, pois é uma exigência dos algoritmos os valores estarem normalizados dessa forma, para isso usa-se a normalização MIN-MAX:

$$z_i = \frac{v_i - \min_{vi}}{\max_{vi} - \min_{vi}} (n\_\max - n\_\min) + n\_\min$$

$$dist(x_1, x_2) = \frac{0{,}9343 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}9393$$

$$dist(x_1, x_3) = \frac{0{,}6921 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}6958$$

$$dist(x_1, x_4) = \frac{0{,}7312 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}7351$$

$$dist(x_1, x_5) = \frac{0{,}9947 - 0}{0{,}9947 - 0}(1-0) + 0 = 1$$

$$dist(x_2, x_3) = \frac{0{,}7706 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}7747$$

$$dist(x_2, x_4) = \frac{0{,}8272 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}8316$$

$$dist(x_2, x_5) = \frac{0{,}8709 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}8755$$

$$dist(x_3, x_4) = \frac{0{,}1208 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}1215$$

$$dist(x_3, x_5) = \frac{0{,}5435 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}5464$$

$$dist(x_4, x_5) = \frac{0{,}5475 - 0}{0{,}9947 - 0}(1-0) + 0 = 0{,}5505$$

Com todos os valores normalizados entre [0,1] a matriz é novamente montada:

$$\begin{bmatrix} 0 & 0{,}9393 & 0{,}6958 & 0{,}7351 & 1 \\ 0{,}9393 & 0 & 0{,}7747 & 0{,}8316 & 0{,}8755 \\ 0{,}6958 & 0{,}7747 & 0 & 0{,}1215 & 0{,}5464 \\ 0{,}7351 & 0{,}8316 & 0{,}1215 & 0 & 0{,}5505 \\ 1 & 0{,}8755 & 0{,}5464 & 0{,}5505 & 0 \end{bmatrix}$$

Para os algoritmos implementados nessa pesquisa cada linha da matriz representa um objeto que a formiga agrupa na grade. A fim de facilitar o entendimento no decorrer da modelagem matemática foi criada a tabela 5 para que cada linha seja representada com uma nomenclatura:

Tabela 5 – Representação de cada linha da matriz por uma nomenclatura.

	P1	**P2**	**P3**	**P4**	**P5**
P1	0	0,9393	0,6958	0,7351	1
P2	0,9393	0	0,7747	0,8316	0,8755
P3	0,6958	0,7747	0	0,1215	0,5464
P4	0,7351	0,8316	0,1215	0	0,5505
P5	1	0,8755	0,5464	0,5505	0

Fonte: Do autor.

Essa matriz normalizada foi usada na modelagem matemática do processo básico do algoritmo SACA, *Ant Based Clustering* e A²CA.

Para a demonstração funcional do algoritmo SACA, *Ant Based Clustering* e A²CA os seguintes parâmetros de entrada foram definidos:

a) **número de formigas**: parâmetro que determina quantas formigas irão trabalhar para agrupar os objetos na grade. Considerando que cada linha da tabela representa um objeto na grade, o total de objetos é 5, então o total de formigas é um número menor que 5, pois cada formiga deve iniciar carregando um objeto. A fim de demostrar todas as funções das formigas e considerando que a base de dados é pequena, o valor para esse parâmetro foi escolhido como 2, pois assim facilita a demonstração de sua funcionalidade;
b) **total de iterações**: Considerando que a base de teste é pequena e por se tratar de uma modelagem, esse valor foi definido como 3;
c) **memória**: o valor para esse parâmetro foi definido como 2 devido a quantidade de formigas e o tamanho da base de dados, o funcionamento desse parâmetro será apresentado na seção 6.2.2.2;
d) **alfa** α : parâmetro que determina a dissimilaridade na formação dos grupos, foi definido como 0.7;
e) **kp e kd:** foram definidos conforme os autores Lumer e Faieta (1994) determinaram, 0.1 e 0.3, respectivamente;
f) **campo de visão**: considerando que a grade onde os objetos e as formigas estão espalhados é pequena o valor para esse parâmetro foi definido como 3;
g) **número de passos**: esse valor é definido de forma aleatório, para essa demonstração os valores definidos foram: Formiga 1 (F1) = 3 e a Formiga 2 (F2) = 2.

4.2.2.1 Modelagem do processo básico do algoritmo SACA

Essa modelagem inicia considerando que todos os parâmetros já foram definidos e a base de teste já está padronizada. Com os parâmetros de entrada definidos o algoritmo espalha os dados na grade bidimensional que corresponde a uma matriz *n* x *n*, seu tamanho é definido por um número inteiro que representa $10 * \sqrt{Total \quad de \quad objetos}$, nesse caso o tamanho da grade corresponde a $10 * \sqrt{5} = 22$, ou seja, a matriz é de 22x22. A fim de facilitar

a demonstração do funcionamento do algoritmo, foi utilizada uma matriz 10x10. Na figura 16 pode-se observar os objetos espalhados na grade:

Figura 16 – Dados espalhados em uma grade 10x10.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	*	*	*	*	*
*	*	*	P2	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	P3	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Com todos os objetos espalhados na grade o algoritmo, aleatoriamente, carrega cada formiga com um objeto, posicionando-a no mesmo lugar do objeto que ela irá carregar. A figura 17 demonstra cada formiga, representada por F1 e F2, na mesma célula que o objeto que está sendo carregado por ela.

Figura 17 – Representação das formigas carregando os seus objetos.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	*	*	*	*	*
*	*	*	F1 P2	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	F2 P3	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

A primeira fase do algoritmo, que é espalhar os dados na grade e definir qual objeto cada formiga irá iniciar carregando, está concluída. A segunda fase é o agrupamento de dados, para isso uma iteração é iniciada e uma formiga é selecionada aleatoriamente para se mover ou deixar o objeto em sua posição.

Considerando que essa parte se trata de uma modelagem, foi escolhido selecionar as formigas, inicialmente, na seguinte ordem F1 e F2.

Com a primeira formiga selecionada F1, verifica-se se ela está carregando um objeto, conforme figura 17 a F1 está carregando o objeto P2, dessa forma verifica-se a possibilidade de deixar o objeto P2 na posição onde ele está, para isso é preciso analisar a vizinhança que a formiga se encontra. O parâmetro campo de visão foi definido como 3, nos parâmetros de entrada, isso significa que cada formiga tem a visão de 1 célula da grade nas suas 8 direções. Na figura 18 é demostrada a F1 no centro do seu campo de visão e observando apenas o objeto P1.

Figura 18 – Formiga F1 no centro do seu campo de visão definido como 3.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	*	*	*	*	*
*	*	*	F1 P2	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	F2 P3	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Conforme verificado na figura 18, a formiga F1 está carregando o objeto P2 e se encontra em uma vizinhança que contém apenas o objeto P1. Dessa forma, calcula-se a probabilidade de deixar o objeto P2 nessa vizinhança, para isso é utilizada a função vizinhança:

$$f(i)\begin{cases}\frac{1}{\sigma^2}\sum_j\left(1-\frac{\delta(i,j)}{\alpha}\right) & se \quad f(i)>0\\ 0 \quad em \quad outros \quad casos\end{cases}$$

A fim de facilitar o entendimento foi calculada a equação $\sum_j\left(1-\frac{\delta(i,j)}{\alpha}\right)$ e $\frac{1}{\sigma^2}$ separadamente. Analisando a tabela 5 o valor de $\delta(P2,P1)$ é 0,9393 e considerando que existe apenas o objeto P1, além do objeto carregado no campo de visão da formiga F1, então $\sum_j\left(1-\frac{\delta(i,j)}{\alpha}\right)$ é igual:

$$(P2, P1) = \left(1 - \frac{0,9393}{0,7}\right) = -0,3419$$

O valor de σ é o campo de visão, no caso 3, o resultado de $\frac{1}{\sigma^2}$ é:

$$\frac{1}{\sigma^2} = \frac{1}{3^2} = \frac{1}{9} = 0,1111$$

Como o valor do campo de visão é um valor fixo, o valor de $\frac{1}{\sigma^2}$ é apenas calculado uma vez, que no caso corresponde a 0,1111.

Aplicando os valores calculados na função vizinhança, tem-se:

$$f(i) = 0,1111 * -0,3419 = -0,0380$$

O valor de $f(i) = -0,0380$ então $f(i) < 0$, logo, seguindo a definição da função vizinhança, $f(i) = 0$.

Calculado o valor da função vizinhança e considerando que a formiga está tentando deixar o objeto, o cálculo da probabilidade de deixar um objeto é realizado pela equação:

$$p_d(i) = \begin{cases} 1 & se \quad f(i) \geq k_d \\ 2 * f(i) & em \quad outros \quad casos \end{cases}$$

O valor de k_d, conforme definido nos parâmetros de entrada, corresponde a 0.3. O valor de $f(i)$ é 0 logo é menor que k_d, então, $p_d = 2 * 0 = 0$, como o valor de p_d é 0, o objeto não deve ser deixado nessa posição, pois após o valor de p_d ser calculado ele deve ser comparado com um número aleatório entre 0 e 1, esse número representa a diferença de análise de uma formiga real quando a mesma encontra um objeto. Assim um objeto para uma formiga pode ser igual a outro, porém para outra formiga esse mesmo objeto pode ser diferente, esse é um comportamento da natureza, quando o valor é igual a 0 então o objeto em questão é muito diferente da sua vizinhança.

Não foi possível deixar o objeto nessa posição, logo a formiga F1 deve ser movimentada para uma direção aleatória, que foi definida como diagonal direita inferior. A

formiga F1 está configurada para dar 3 passos na grade, logo a sua nova posição pode ser observada na figura 19, juntamente com as posições que percorreu na grade.

Figura 19 – Representação dos passos da formiga F1.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	*	*	*	*	*
*	*	*	Ponto inicial	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	F2 P3	*	*
*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Após a tentativa de deixar o objeto na posição e a movimentação da formiga F1, outra formiga deve ser selecionada, como definido a próxima formiga é F2. Na figura 20 pode-se observar a nova formação da grade e o campo de visão da formiga F2.

Figura 20 – Representação da grade após a movimentação da formiga F1 e demonstração do campo de visão da formiga F2.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	F2 P3	*	*
*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

A formiga F2 está carregando o objeto P3, logo a função da formiga F2 nesse momento é verificar se é possível deixar o objeto P3 na posição em que ela se encontra. Conforme o campo de visão da formiga F2 o único objeto compreendido nele é o objeto P2, deve-se então calcular a função vizinhança da formiga F2 e a probabilidade p_d. Verificando a tabela 5 o valor de $\delta(P3, P2)$ é 0,7706, calculando:

$$(P3, P2) = \left(1 - \frac{0,7747}{0,7}\right) = -0,1067$$

$$f(i) = 0,1111 * -0,1067 = -0,0119$$

O valor de $f(i) = -0,0119$ então $f(i) < 0$, logo, seguindo a definição da função vizinhança $f(i) = 0$. O valor da p_d será 0, logo o objeto não deve ser deixado nessa vizinhança. Com isso, a formiga F2 é movimentada aleatoriamente carregando o objeto P3 para outra posição na grade. A formiga F2 está configurada para realizar 2 passos na grade. A direção escolhida é a diagonal esquerda inferior (figura 21).

Figura 21– Grade atualizada demonstrando o campo de visão da formiga F2.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	*	F2 P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Todas as formigas foram movimentadas, terminando-se a primeira iteração. Na segunda iteração a formiga F2 será selecionada primeiro. De acordo com a figura 21, a formiga F2 tem em seu campo de visão os objetos P2 e P4, a função da formiga F2 é de deixar o objeto P3 nessa vizinhança, então a função vizinhança e a probabilidade de deixar o objeto são calculadas, observando-se a tabela 5:

$$(P3, P2) = \left(1 - \frac{0,7747}{0,7}\right) = -0,1067$$

$$(P3, P4) = \left(1 - \frac{0,1215}{0,7}\right) = 0,8264$$

$$f(i) = 0{,}1111 * 0{,}7197 = 0{,}0800$$

$$p_d = 2 * 0{,}0800 = 0{,}1599$$

Foi considerado que o valor aleatório entre 0 e 1 é 0.1, então comparando o resultado de p_d com o valor 0.1, $p_d = 0{,}1599$ é maior que 0.1, logo o objeto pode ser deixado nessa vizinhança. Com isso, a formiga F2 deverá procurar por um item aleatório que não está sendo carregado e calcular a probabilidade de pegar o objeto, no caso a equação p_p.

Os objetos P2, P4 e P5 estão livres na grade, ou seja, nenhuma formiga está carregando eles. O objeto P3 não é válido para a formiga F2 tentar carregar, pois no momento ela acaba de deixá-lo. Dessa forma, foi considerado que a formiga F2 escolheu o objeto P4 para carregar (figura 22), assim, o cálculo da probabilidade de pegar um objeto é dado por:

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

Onde o valor de k_p está definido como 0.1 conforme os parâmetros de entrada.

Figura 22 – Grade após a F2 escolher o objeto P4 para carregar e demonstração do seu campo de visão.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	F2 P4	*	*	*	*	*

Fonte: Do autor.

O único objeto na vizinhança de F2 é o objeto P3, considerando a tabela 5 deve-se calcular a função vizinhança e depois a probabilidade de carregar (p_p) desse objeto:

$$(P4, P3) = \left(1 - \frac{0{,}1215}{0{,}7}\right) = 0{,}8264$$

$$f(i) = 0{,}1111 * 0{,}8264 = 0{,}0918$$

$$p_p = \left(\frac{0.1}{0.1 + 0{,}0918}\right) = 0{,}2718$$

Da mesma forma que p_d o resultado de p_p deve ser comparado a um valor aleatório entre 0 e 1 onde p_p deve ser maior que este, considerando que o valor aleatório seja 0.5, logo p_p é menor que o valor aleatório. Dessa forma, F2 deve escolher outro objeto livre na grade para carregar. Aleatoriamente o próximo objeto que a formiga F2 irá tentar carregar é o objeto P1 (figura 23).

Figura 23 - Grade após a F2 escolher o objeto P1 para carregar e demonstração do seu campo de visão.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Conforme a figura 23, a vizinhança da formiga F2 está vazia, não há objetos em seu campo de visão, logo $f(i)$ será 0, dessa forma, calcula-se a p_p:

$$p_p = \left(\frac{0.1}{0.1 + 0}\right) = 1$$

O valor de p_p é igual a 1, considerando que o valor aleatório é de 0 a 1, este nunca será maior que p_p, enquanto p_p for maior ou igual a 1. Dessa forma, o objeto deve ser

carregado pela formiga F2. Agora F2 tem um objeto para carregar e com isso termina a movimentação da formiga F2. A próxima formiga é selecionada, no caso F1. A formiga F1 ainda está carregando o objeto P2, então ela deve tentar deixar o objeto P2 na posição da grade onde ela se encontra. Conforme a figura 24, o objeto P3 é o único objeto no campo de visão da formiga F1.

Figura 24 – Formiga F1 com seu campo de visão.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Calculando a probabilidade de deixar o objeto P2 na posição da grade em que a formiga F1 se encontra:

$$(\text{P2, P3}) = \left(1 - \frac{0{,}7747}{0{,}7}\right) = \text{-}0{,}1067$$

$$f(i) = 0{,}1111 * -0{,}1067 = -0{,}0119$$

A função $f(i)$ é menor que 0, dessa forma, conforme definição da função vizinhança $f(i)$ = 0. De acordo com a função da probabilidade de deixar um objeto, p_d = 0. O objeto P2 não deve ser deixado nessa vizinhança, logo a formiga F1 deve ser movimentada para outra posição na grade carregando o objeto P2 com ela. F1 pode dar 3 passos na grade e a direção apontada aleatoriamente é para baixo. Na figura 25 pode-se observar a grade com a formiga F1 em sua nova posição e a trajetória que a mesma percorreu.

Figura 25 – Formiga F1 em sua nova posição na grade e a trajetória que a mesma percorreu.

*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Pode-se observar na figura 25 que a formiga F1 apareceu no topo da grade, isso ocorre porque a grade não tem limitação, então a parte superior se une com a inferior. Considerando que a formiga F1 esta configurada para caminhar 3 passos e abaixo dela existia apenas 2 células, a terceira célula da grade fica na parte superior da mesma. Com isso, todas as formigas já foram selecionadas, dessa forma termina a segunda iteração.

A terceira iteração começa selecionando a formiga F2, a mesma está carregando o objeto P1, conforme figura 26, logo deve-se tentar deixar esse objeto na posição em que ela se encontra.

Figura 26 – Formiga F2 em sua posição na grade e o campo de visão que a mesma possuí.

*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

De acordo com a figura 26, F2 não tem nenhum outro objeto em seu campo de visão, dessa forma $f(i)$ é igual a 0, com isso a probabilidade p_d também é 0, conclui-se então que o objeto P1 não deve ser deixado nessa posição. A formiga F2 deve então ser movimentada na grade carregando o objeto P1. Foi definido que a direção que a formiga F2 irá se movimentar é a diagonal superior direita. F2 está configurada para realizar dois passos

na grade. Na figura 27 pode-se observar que a nova posição da formiga F2 já está ocupada com o objeto P5. Como o algoritmo SACA não permite que dois objetos dividam a mesma posição, a formiga tem que escolher aleatoriamente uma célula livre, próxima a posição ocupada, essa célula foi definida como a próxima para cima (figura 28).

Figura 27 – Trajetória da formiga F2 para uma posição na grade ocupada pelo padrão P5.

*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Figura 28 – Formiga F2 em sua posição na grade após escolher aleatoriamente uma posição próxima a posição ocupada.

*	*	*	*	F2 P1	*	F1 P2	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Pode-se observar na figura 29 como se encontra a distribuição na grade.

Figura 29 - Distribuição dos objetos e formigas na grade.

*	*	*	*	F2 P1	*	F1 P2	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Assim termina nessa iteração a movimentação da formiga F2, considerando que ainda falta movimentar uma formiga, outra formiga deve ser selecionada. Como a seleção da formiga é aleatória, a formiga selecionada novamente foi a formiga F2, o algoritmo não impede selecionar a mesma formiga desde que seja aleatório. A figura 30 mostra a formiga F2 e seu campo de visão.

Figura 30 - Formiga F2 e seu campo de visão contendo o objeto P5 e P4.

*	*	*	*	F2 P1	*	F1 P2	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Conforme a figura 30, no campo de visão da formiga F2 está contido os objetos P5 e P4, baseado na tabela 5 calcula-se a probabilidade de deixar o objeto P1 nessa vizinhança.

$$(P1,P4) = \left(1 - \frac{0,7351}{0,7}\right) = -0,0501$$

$$(P1,P5) = \left(1 - \frac{1}{0,7}\right) = -0,4286$$

$$f(i) = 0{,}1111 * -0{,}4787 = -0{,}0532$$

O valor de $f(i)$ é menor que 0, então conforme a definição da função $f(i)$, o valor de $f(i)$ é 0, dessa forma, calculando a probabilidade de deixar um objeto, p_d é igual a 0, logo o objeto não deve ser deixado nessa posição. A formiga F2 deve ser movimentada para outra posição na grade, carregando o objeto P1, essa posição foi definida como diagonal superior esquerda, considerando que a formiga F2 está configurada para realizar 2 passos na grade, pode-se observar a nova posição da formiga F2 na figura 31.

Figura 31 – Formiga F2 na sua nova posição e a trajetória que a mesma percorreu.

*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Chegou ao fim da terceira iteração e considerando que o parâmetro total de iterações esta definido como 3, o algoritmo chegou ao fim de sua execução. A modelagem apresentada é o processo básico do algoritmo SACA, basicamente todas as situações que uma formiga pode encontrar durante as iterações foram apresentadas, essas situações devem ser encontradas em todos os algoritmos originados do SACA, como por exemplo, o algoritmo *Ant Based Clustering* e o A²CA. O objetivo dessa modelagem não foi chegar a uma conclusão de *clusters,* e sim demonstrar o funcionamento básico do algoritmo. A seguir é apresentado o uso da memória de curta duração do algoritmo, esse processo foi modelado separadamente para facilitar o entendimento.

4.2.2.2 Modelagem da memória de curta duração

A memória de curta duração é representada como um vetor de tamanho definido no parâmetro Memória do algoritmo. O objetivo dessa memória é possibilitar a formiga se lembrar da posição onde ela deixou os últimos objetos carregados. Dessa forma, a formiga se

projeta para essa posição e verifica a probabilidade de deixar o objeto que ela está carregando na melhor posição que ela se lembra. Essa melhor posição é calculada pela função vizinhança, ou seja, a formiga se projeta para cada posição que ela possui na memória e calcula a função vizinhança, a melhor função vizinhança é escolhida para ser usada para calcular a probabilidade de deixar o objeto. Se nenhuma posição for escolhida, ou seja, o objeto não se encaixa na vizinhança de nenhuma posição, a memória não é utilizada e a formiga continua seu movimento normalmente escolhendo uma direção aleatória.

A memória começa a ser usada a partir do momento que o vetor memória estiver completamente cheio, se a memória está definida como tamanho 3, ela só ira ser considerada na movimentação da formiga se todas as três posições do vetor memória estiverem com valores armazenados. Quando o vetor memória estiver completo a próxima posição armazenada irá substituir a memória que mais tempo esteve armazenada. A leitura da memória é realizada da mais recente até a mais antiga.

Na figura 32 é apresentado um caso em que estão espalhados na grade 5 objetos, os mesmos objetos são utilizados na sessão 4.2.2.1. Na figura 33 é apresentada a grade com valores x e y em cada célula para demonstrar as coordenadas que representam as posições onde a formiga deixou os últimos objetos carregados.

Figura 32 – Objetos e formigas espalhados na grade para demonstração do uso da memória.

*	*	*	*	*	*	*	F1 P2	*	*
*	*	*	*	F2 P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Figura 33 – Grade demostrando os valores x e y correspondentes a cada posição ou célula.

1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9	3,10
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9	4,10
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9	5,10
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9	6,10
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9	7,10
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9	8,10
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9	9,10
10,1	10,2	10,3	10,4	10,5	10,6	10,7	10,8	10,9	10,10

Fonte: Do autor.

Para realizar a modelagem foi definido que a formiga F2 possui em sua memória as posições {(5,8), (10,6), (9,7)}. A formiga F2 está carregando o objeto P5 e tenta deixá-lo em alguma posição, antes ela verifica a sua memória a fim de encontrar a melhor posição para deixá-lo. A primeira posição verificada é a posição (5,8), pode-se observar na figura 34 a formiga F2 projetada para a posição (5,8).

Figura 34 – Formiga F2 em sua posição atual e na posição x, y (5,8) demarcando o seu campo de visão.

*	*	*	*	*	*	*	F1 P2	*	*
*	*	*	*	F2 P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	F2 P5	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	P3	*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Conforme a figura 34, a formiga F2 não encontra nenhum objeto na sua vizinhança, isso se justifica pelo fato do objeto ter sido movimentado por outra formiga, logo $f(i)$ na posição (5,8) é 0. A próxima posição a ser verificada é a (10,6). Na figura 35 pode-se observar a formiga F2 na sua posição atual e projetada na posição (10,6).

Figura 35 – Formiga F2 em sua posição atual e na posição x, y (10,6) demarcando o seu campo de visão.

*	*	*	*	*	*	*	F1 P2	*	*
*	*	*	*	F2 P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	P3	*	*	*	*
*	*	*	*	P4	F2 P5	*	*	*	*

Fonte: Do autor.

Observando a figura 35, a formiga F2 tem em seu campo de visão os objetos P3 e P4, o cálculo da função $f(i)$ é realizado baseando-se na tabela 5.

$$(\text{P5,P3}) = \left(1 - \frac{0{,}5464}{0{,}7}\right) = 0{,}3907$$

$$(\text{P5,P4}) = \left(1 - \frac{0{,}5505}{0{,}7}\right) = 0{,}3815$$

$$f(i) = 0{,}1111 * 0{,}7722 = 0{,}0857$$

O valor de $f(i)$ em (10,6) é 0,0857, até então a melhor similaridade encontrada nas posições da memória. A próxima posição a ser verificada é a (9,7). Na figura 36 pode-se observar a formiga F2 na sua posição atual e projetada na posição (9,7).

Figura 36 – Formiga F2 em sua posição atual e na posição x, y (9,7) demarcando o seu campo de visão.

*	*	*	*	*	*	F1 P2	*	*	*
*	*	*	*	F2 P5	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	P1	*	*	P3	F2 P5	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

O objeto P3 está contido no campo de visão da formiga F2 projetada na posição (9,7), dessa forma calcula-se a função $f(i)$ dessa posição.

$$(P5, P3) = \left(1 - \frac{0,5464}{0,7}\right) = 0,2194$$

$$f(i) = 0,1111 * 0,2194 = 0,0244$$

O valor da função $f(i)$ na posição (9,7) é 0,0244, com o fim da varredura da memória, o melhor valor de $f(i)$ foi encontrado na posição (10,6). Dessa forma, a formiga calcula a probabilidade de deixar p_d, conforme apresentado na sessão 4.2.2.1, do objeto nessa posição escolhida. Caso a formiga F2 possa deixar esse objeto na posição escolhida, ela escolherá um objeto livre para carregar. Caso contrário, mesmo encontrando uma boa similaridade pela memória a formiga F2 não consiga deixar o objeto na posição escolhida, ela calcula a probabilidade de deixar na sua posição atual e continua com o processo normalmente, conforme demonstrado na sessão 4.2.2.1.

Esse processo de memória é utilizado também pelos algoritmos *Ant Based Clustering* e A²CA sem sofrer alteração.

4.2.2.3 Modelagem do algoritmo *Ant Based Clustering*

Como citado na seção 4.2.2.1, o algoritmo *Ant Based Clustering* utiliza o mesmo processo aplicado no SACA, a diferença deste são as mudanças nas equações p_p e p_d que descartam o uso das constantes k_p e k_d, além de uma variação na equação da função vizinhança em que o valor de $\frac{1}{\sigma^2}$ passa a ser $\frac{1}{Nocc}$, onde *Nocc* é o número de objetos que a formiga encontrou na sua vizinhança. Essa mudança ocorre baseada na iteração em que o algoritmo se encontra. O algoritmo é dividido em três partes, a partir do parâmetro *Total de Iterações* são realizados os seguintes cálculos: *0.45 * Total de Iterações*, o valor resultante desse cálculo determina que até o valor da iteração chegar neste número, o algoritmo irá utilizar $\frac{1}{\sigma^2}$ na equação da função vizinhança. A partir do momento que a iteração for maior

que este resultado o algoritmo passa a utilizar $\frac{1}{Nocc}$ na função vizinhança. O algoritmo utilizará $\frac{1}{Nocc}$ até que o valor da iteração alcance o valor calculado em *0.55*Total de Iterações*, passando novamente a utilizar o valor $\frac{1}{\sigma^2}$ e agora com valor do campo de visão σ valendo 5, conforme determinado por Handl (2003). A seguir são apresentadas as quatro equações utilizadas nesse algoritmo:

$$p_p(i) = \begin{cases} 1 & se \quad f(i) \leq 1 \\ \frac{1}{f(i)^2} & em \quad outros \quad casos \end{cases}$$

$$p_d(i) = \begin{cases} 1 & se \quad f(i) \geq 1 \\ f(i)^4 & em \quad outros \quad casos \end{cases}$$

Nota-se que as equações p_p e p_d não necessitam dos parâmetros k_p e k_d.

$$f(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j (1 - \frac{\delta(i,j)}{\alpha}) & se \quad f(i) > 0 \wedge \left(1 - \frac{\delta(i,j)}{\alpha}\right) > 0 \\ 0 & em \quad outros \quad casos \end{cases}$$

$$f^*(i) = \begin{cases} \frac{1}{Nocc} \sum_j (1 - \frac{\delta(i,j)}{\alpha}) & se \quad f(i) > 0 \wedge \left(1 - \frac{\delta(i,j)}{\alpha}\right) > 0 \\ 0 & em \quad outros \quad casos \end{cases}$$

Outra mudança nesse algoritmo é que o valor de alfa α é aleatório para cada formiga, ou seja, não é necessário informar esse parâmetro na configuração do algoritmo. Além disso, o valor de alfa α possui uma adaptação conforme descrita na sessão 3.1.2.7. O campo de visão é outro parâmetro que não tem necessidade de ser informado, pois inicia com o valor 3 e quando o número de iteração ultrapassa o valor calculado em *0.55*Total de Iterações*, o valor do campo de visão passa a ser 5. A seguir são apresentadas situações para demostrar o funcionamento das modificações propostas nesse algoritmo.

Primeiramente o algoritmo parametriza cada formiga com o campo de visão inicial que é 3 e cada formiga também recebe um valor aleatório entre 0 e 1 para alfa α. Na realização dessa modelagem utiliza-se 2 formigas e 5 objetos. O alfa α foi escolhido aleatoriamente para cada formiga, os valores foram F1 = 0.7 e F2 = 0.2. Calculando o valor que determina como será calculada a função vizinhança, considerando que o número *Total de Iterações* é 10000, temos *0.45 * 10000=4500* e o valor final é *0.55*10000=5500.* No primeiro caso a ser demonstrado o algoritmo está na iteração 2400 e a formiga F1 está tentando deixar um objeto. Na figura 37, pode-se observar a situação da formiga F1 que está carregando o objeto P5.

Figura 37 – Formiga F1 carregando o objeto P5 e seu campo de visão demarcado.

*	*	*	*	*	*	P2	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	P3	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	F1 P5		*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Conforme a figura 37 apenas o objeto P4 está contido no campo de visão da formiga F1, portando calcula-se a probabilidade p_d de deixar o objeto P5 nessa vizinhança, observando-se os valores da tabela 5.

$$(\text{P5, P4}) = \left(1 - \frac{0{,}5505}{0{,}7}\right) = 0{,}4439$$

Na função de $f(i)$ é utilizado o valor de $\frac{1}{\sigma^2}$, pois conforme definido o valor da iteração está em 2400, o valor de $\frac{1}{\sigma^2}$ é igual a 0,1111 .

$$f(i) = 0{,}1111 * 0{,}4439 = 0{,}0612$$

O valor de $f(i)$ é igual a 0.0612, observando a definição da equação p_d calcula-se a probabilidade de deixar o objeto da seguinte forma:

$$p_d = 0{,}0612^4 = 0{,}000014$$

Caso a formiga estivesse tentando pegar o objeto o cálculo de p_p, observando a definição da equação p_p, é feito da seguinte forma:

$$p_p(i) = \begin{cases} 1 & se \quad f(i) \leq 1 \\ \dfrac{1}{f(i)^2} & em \quad outros \quad casos \end{cases}$$

$$p_p = 1$$

Da mesma forma que o SACA, o valor de p_d e p_p devem ser comparados a um valor aleatório entre 0 e 1, esse valor foi definido como 0.03. Dessa forma, o valor de p_d é menor que o valor aleatório, logo o objeto não deve ser deixado nessa posição. E o valor de p_p é maior que 0.03, portanto, a formiga F1 poderia pegar o objeto nessa posição.

Considerando que o valor da iteração está em 5103, na função vizinhança é utilizado o valor de $\frac{1}{Nocc}$, só existe 1 objeto contido no campo de visão da formiga F1, portanto calcula-se $\frac{1}{Nocc}$ da seguinte forma:

$$\frac{1}{Nocc} = \frac{1}{1} = 1$$

Com o valor de $\frac{1}{Nocc}$ definido, calcula-se a função vizinhança e a probabilidade de deixar o objeto P5 nessa vizinhança.

$$f(i) = 1 * 0{,}4439 = 0{,}4439$$

$$p_d = 0{,}4439^4 = 0{,}1970$$

Caso a formiga estivesse tentando pegar o objeto o cálculo de p_p, observando a definição da equação p_p, é feito da seguinte forma:

$$p_p(i) = \begin{cases} 1 & se \quad f(i) \leq 1 \\ \dfrac{1}{f(i)^2} & em \quad outros \quad casos \end{cases}$$

$$p_p = 1$$

Calculado o valor de p_d e p_p, comparam-se estes com o valor aleatório definido como 0.03. O valor de p_d é maior que 0.03, logo o objeto deve ser deixado nessa posição. O valor de p_p também é maior que 0.03, dessa forma o objeto poderia também ser pego pela formiga F1.

A adaptação do valor de alfa α é demonstrada da seguinte forma, considerando que a formiga F2 tentou deixar um objeto 53 vezes e o seu valor de alfa α é 0.2, calcula-se o novo valor de α:

$$rfalhas = \frac{Nfalhas}{Nefetivos}$$

Conforme descrição na seção 3.1.2.7 o valor de *Nefetivos* é igual a 100, então:

$$rfalhas = \frac{53}{100} = 0{,}53$$

O valor de *rfalhas* é 0.53, portanto é subtraído 0.01 do valor alfa α, dessa forma o novo valor de α da formiga F2 é 0.19. Esse processo é realizado toda vez que a formiga está tentado descarregar e o *Nfalhas* é somado sempre que ela não consegue deixar o objeto, sendo zerado quando ela o deixa.

O algoritmo *Ant Based Clustering* necessita de muitas iterações para começar a formar os grupos, porém ele tem a vantagem de não necessitar de muitos dos parâmetros de entrada do algoritmo.

4.2.2.4 Modelagem do algoritmo A²CA

O algoritmo A²CA diferencia-se do SACA pelo fato de não necessitar informar o parâmetro *Campo de Visão*, ter um resfriamento do parâmetro k_p e utilizar uma heurística de feromônio que influencia no cálculo das probabilidades p_d e p_p. A função vizinhança é a mesma utilizada pelo SACA e a equação p_d e p_p são diferentes. A seguir são apresentadas as equações p_d e p_p utilizadas nesse algoritmo:

$$p_d = \frac{1}{f(i)\phi(i)}\left(\frac{f(i)}{f(i)+k_d}\right)^2$$

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

Na figura 38 pode-se observar como a grade é iniciada com os valores dos feromônio em cada célula, esse valor foi definido como 1 para demonstrar a evaporação do feromônio.

Figura 38 – Grade com representação do feromônio inicial.

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Fonte: Do autor.

Para modelar o algoritmo A²CA utilizaram-se 2 formigas e 5 objetos, os mesmos da tabela 5. Primeiramente, o algoritmo define aleatoriamente o campo de visão das formigas, F1=3 e F2 = 5. Na figura 39 tem-se a formiga F1 tentando deixar o objeto P5 em sua posição:

Figura 39 – Formiga F1 carregando o objeto P5 e seu campo de visão demarcado.

*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	P2	*	*
*	*	*	*	*	*	*	*	*	*
*	*	F2 P1	*	*	P3	*	*	*	*
*	*	*	*	F1 P5		*	*	*	*
*	*	*	*	P4	*	*	*	*	*

Fonte: Do autor.

Conforme a figura 39 a formiga F1 tem em seu campo de visão os objetos P3 e P4. Na posição atual da formiga F1 o valor para o feromônio $\phi(i)$, conforme figura 38, é 1, com isso calcula-se a probabilidade da formiga F1 deixar o objeto P5 na vizinhança atual, conforme tabela 5, calcula-se:

$$(\text{P5, P4}) = \left(1 - \frac{0{,}5505}{0{,}7}\right) = 0{,}4439$$

$$(\text{P5, P3}) = \left(1 - \frac{0{,}5464}{0{,}7}\right) = 0{,}5592$$

$$f(i) = 0.1111 * 1{,}003 = 0{,}1115$$

$$p_d = \frac{1}{0{,}1115 * 1}\left(\frac{0{,}1115}{0{,}1115 + 0.3}\right)^2 =$$

$$p_d = 8{,}9686 * 0.0734 = 0{,}6595$$

Se a formiga estivesse tentando pegar o objeto o cálculo de p_p seria dado por:

$$p_d = 0.1115 * 1\left(\frac{0.1}{0.1115 + 0.1}\right)^2$$

$$p_p = 0.1115 * 1 * 0.4728 = 0{,}0537$$

Comparando os valores de p_p e p_d com o valor aleatório 0.6, p_p é menor que 0.6, portanto, a formiga F1 não pode pegar o objeto. O valor de p_d é maior que 0.6, então a formiga F1 deve deixar o objeto na posição que ela se encontra. Assim, a grade de feromônio deve ser atualizada acrescentando 0.01 na posição onde a formiga F1 deixou o objeto e 0.005 nas posições vizinhas a esta, conforme definido por Vizine et al (2005). Na figura 40 pode-se observar a grade de feromônio atualizada.

Figura 40 – Grade de feromônio atualizada após a formiga F1 deixar o objeto P5.

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1.005	1.005	1.005	1	1	1	1
1	1	1	1.005	1.01	1.005	1	1	1	1
1	1	1	1.005	1.005	1.005	1	1	1	1

Fonte: Do autor.

A cada iteração o feromônio deve evaporar, para isso cada valor de feromônio deve ser multiplicado por 0.99, conforme definido por Vizine et al (2005). Na figura 41 pode-se observar a grade de feromônio após passar uma iteração.

Figura 41 – Grade de feromônio atualizada após passar por uma iteração.

0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0,99495	0,99495	0,99495	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0,99495	0,9999	0,99495	0.99	0.99	0.99	0.99
0.99	0.99	0.99	0,99495	0,99495	0,99495	0.99	0.99	0.99	0.99

Fonte: Do autor.

O resfriamento do parâmetro k_p ocorre a cada ciclo de 10000 iteração, conforme definido por Vizine et al (2005). Considerando que o número de iteração esta em 10001 o valor de k_p é reajustado da seguinte forma, considerando que k_p é igual a 0.1 conforme definido nos parâmetros de entrada:

$$k_p = 0.1 * 0.98 = 0.098$$

Esse reajuste acontece até que k_p alcance o valor de 0.001, que é o valor mínimo de k_p.

O *Campo de Visão* no algoritmo A²CA é acrescentado 2 unidades de acordo com a função $f(i)$, ou seja, quando $f(i)$ é maior que θ e o valor do campo de visão é menor que 7. O valor de θ é 0.6 e o tamanho máximo do campo de visão é 7, conforme definido por Vizine et al (2005). Dessa forma, considerando que a $f(i)$ é igual a 0.74 e o valor do campo de visão da formiga F1 é igual a 3, F1 agora tem um campo de visão de valor 5. Toda vez que o campo de visão é alterado, a função vizinhança é calculada novamente com o novo campo de visão.

Essas são as modificações propostas no algoritmo A²CA, a seguir serão apresentadas as medidas de validações utilizadas nessa pesquisa. Só foi possível utilizar essas medidas com a implementação de uma lógica que verifica a vizinhança de cada objeto na grade após a conclusão dos algoritmos. Essa lógica se baseia no processo de campo de visão das formigas e quando um número de objetos na vizinhança é alcançado esse objeto é considerado um *cluster* , assim como todos da sua vizinhança, quando um objeto que ainda não está classificado como *cluster* e tem uma vizinhança considerável é encontrado esse objeto é um novo *cluster*, assim como toda a sua vizinhança. Nessa pesquisa os valores utilizados para determinar o tamanho da vizinhança são: 11 para o SACA e A²CA e 7 para *Ant Based Clustering.*

4.2.3 Índices Empregados na Validação

Os índices de validação utilizados nesse trabalho foram: índice Dunn e C-index, implementados pelo Bacharel em Ciência da Computação Éverton Marangoni Gava, em 2011.

O índice de *Dunn* foi escolhido a fim de ter uma visão da compactação interna e da separação externa dos grupos formados pelos algoritmo. O *C-index* foi utilizado com o

propósito de avaliar a homogeneidade de cada *cluster* encontrado de forma separada (BENZEK, 2005, tradução nossa; MILLIGAN; COOPER, 1985, tradução nossa).

A validação do resultado geralmente é feita por métodos estatísticos com o objetivo de determinar a qualidade dos grupos encontrados pelo algoritmo de clusterização. Na clusterização de dados não existem grupos pré-definidos e exemplos que possam mostrar se os *clusters* encontrados por um determinado algoritmo são significativos (GAN; MA; WU, 2007, tradução nossa; JAIN; DUBES, 1988, tradução nossa).

4.2.3.1 Índice de *Dunn*

O índice *Dunn* foi proposto por J.C. *Dunn* em 1974 e tem como objetivo avaliar as partições para identificar o quanto os *clusters* encontrados são compactados e bem separados. Caso isso seja identificado é esperado que a diferença entre os *clusters* seja grande e o diâmetro dos grupos seja pequeno. Pode-se dizer que valores mais distantes de zero para o índice *Dunn* indicam presença de *clusters* compactados e bem separados no conjunto de dados (LAROSE, 2005, tradução nossa; JAIN; DUBES, 1988, tradução nossa).

O índice de *Dunn* é a razão entre a menor distância entre instâncias que não estejam no mesmo agrupamento e a maior distância intra-agrupamento (BROCK; PIHUR; DATTA; DATTA, 2008, tradução nossa).

$$Dunn = \min_{l<i<k}\left(\min_{i+l\le j\le k}\left(\frac{dist(c_i, c_j)}{\max \quad diam \quad (c_l)}\right)\right) \tag{18}$$

Onde:

a) *Dunn*: índice de *Dunn*;

b) $dist(c_i, c_j)$**:** distância entre o *i*-ésimo e o *j*-ésimo *cluster* de tal forma que $dist(c_i, c_j) = \min_{x_i \in c_i, x_j \in c_j} dist(x_i, x_j)$;

c) *k*: número total de *clusters*;

d) $diam(c_l)$**:** dispersão do *l*-ésimo *cluster* onde $diam(c_l) = \max_{x_{l_1}, x_{l_2} \in c_l} dist(x_{l_1}, x_{l_2})$.

O índice de *Dunn* é sensível à presença de *outliers,* dessa forma estes pontos não são considerados no cálculo. Além disso, para se encontrar o índice de *Dunn* é necessário ter no mínimo dois *clusters*.

4.2.3.2 *C-Index*

O índice *C-Index* apresenta valores para cada *cluster* encontrado separadamente, seus resultados estão no intervalo de [0,1], onde quanto mais próximo de zero for o resultado melhor é considerada a clusterização (MILLIGAN; COOPER, 1985, tradução nossa).

O cálculo do *C-index* é definido pela equação 19:

$$C - Index = \frac{d_w(C_i) - \min(n_w)}{\max(n_w) - \min(n_w)} \quad (19)$$

Onde:

a) $d_w(C_i)$: somatório das distâncias entre todos os pares de pontos no *cluster* C_i;

b) n_w: quantidade de pares de pontos no *cluster* C_i;

c) $\min(n_w)$: somatório das n_w menores distâncias entre os pares de objetos do conjunto de dados;

d) $\max(n_w)$: somatório das n_w maiores distâncias entre os pares de objetos do conjunto de dados.

C_i é a soma das distâncias entre todos os pares de pontos no *cluster,* definida por:

$$d_w(C_i) = \sum_{x_i, x_{,} \in C_i} dist(x_i, x_j) \quad (20)$$

4.2.4 Implementação e Realização de Testes

O algoritmo SACA foi desenvolvido no módulo de clusterização da *Shell Orion Data Mining Engine*, por meio da linguagem de programação Java e do ambiente de programação integrado *NetBeans* 7.0.1.

Além do algoritmo SACA foram implementados mais dois algoritmos que possuem aperfeiçoamentos do SACA, o *Ant Based Clustering* (HANDL, 2003) e o A²CA (VIZINE et al, 2005).

Está presente na *Shell* Orion a possibilidade de estabelecer conexões com diversos Sistemas Gerenciadores de Bancos de Dados (SGBD), contando que esse SGDB disponibilize um *driver* JDBC para realizar a conectividade.

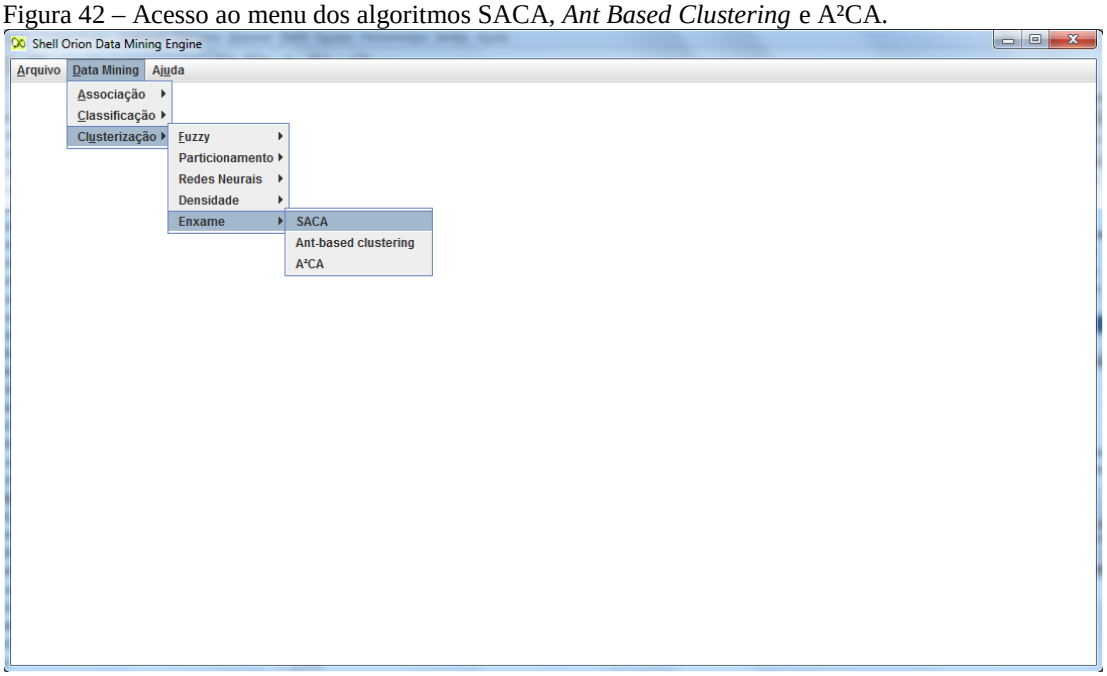
Na implementação e realização de testes do algoritmo SACA, *Ant Based Clustering* e A²CA o SGBD escolhido foi o *MySQL*, por ser gratuito e de simples portabilidade, além de ser muito difundido.

Com o SGBD definido, testes foram realizados com a base de dados que contém indicadores ambientais das bacias hidrográficas da região carbonífera catarinense, a fim de verificar os resultados obtidos dos algoritmos desenvolvidos.

A base de dados escolhida já contém os dados devidamente pré-processados e normalizados, devendo-se realizar a inserção dos mesmos no *MySQL* por meio de comandos SQL, e após isso é realizada a conexão da *Shell* Orion com o SGBD no menu *Arquivo*, submenu *Conectar*.

Assim que a Shell Orion estiver conectada a um SGBD, torna-se possível acessar os algoritmos SACA, *Ant Based Clustering* e A²CA por meio do menu *Data Mining* e dos submenus *Clusterização*, *Enxame*, e selecionando um dos algoritmos (figura 42).

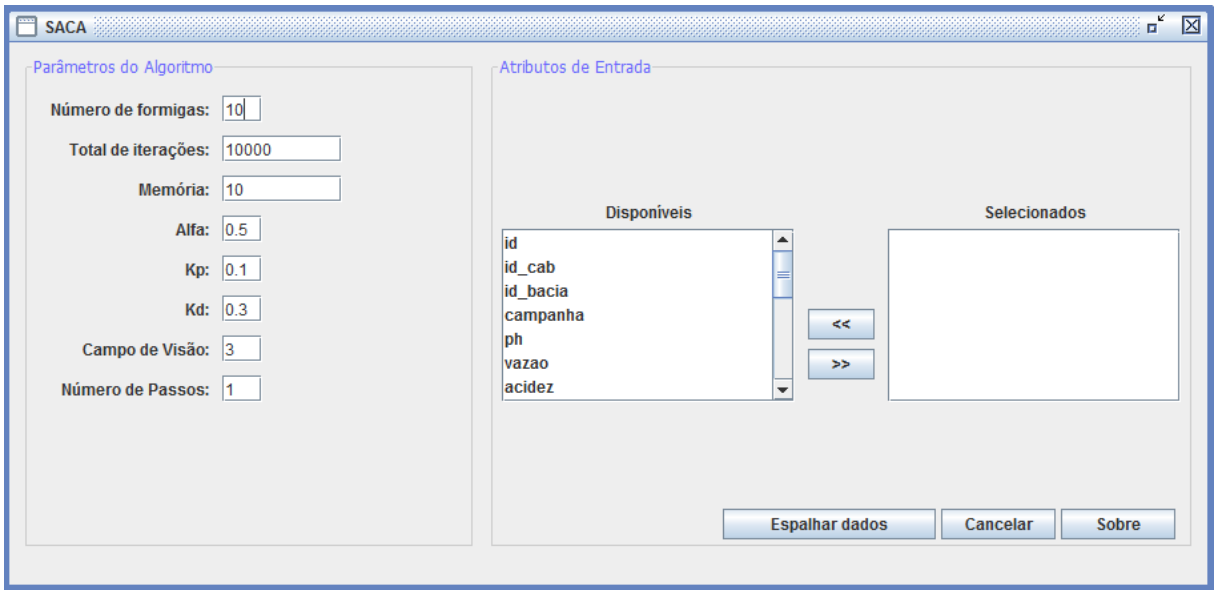
Figura 42 – Acesso ao menu dos algoritmos SACA, *Ant Based Clustering* e A²CA.



Fonte: Do autor.

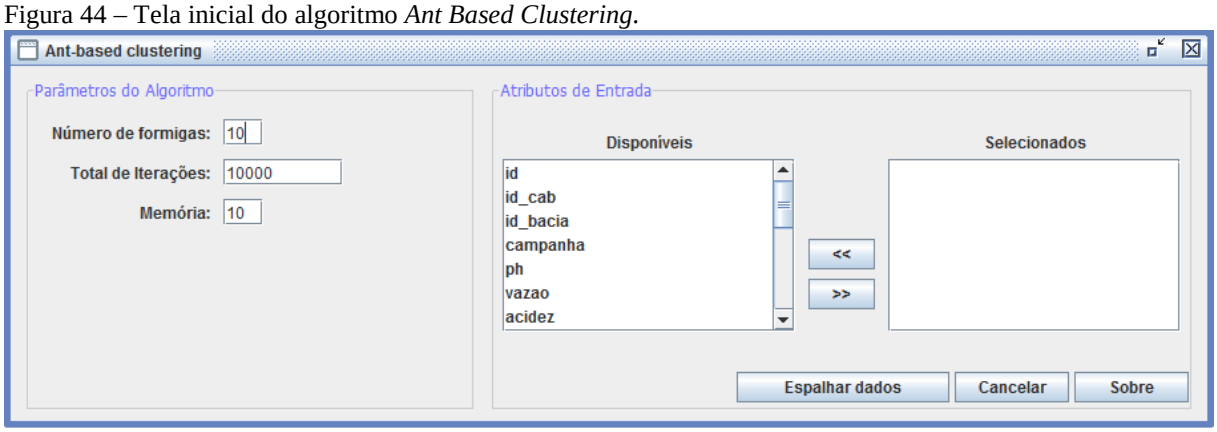
Para realizar a tarefa de clusterização por meio dos algoritmos implementados nessa pesquisa é preciso que o usuário informe alguns parâmetros de entrada que variam de acordo com o algoritmo. Como padrão esses parâmetros estão situados no quadrante esquerdo denominado *Parâmetros do Algoritmo* na tela inicial de cada algoritmo. As figuras 43, 44 e 45 apresentam as telas iniciais dos algoritmos.

Figura 43 – Tela inicial do algoritmo SACA.



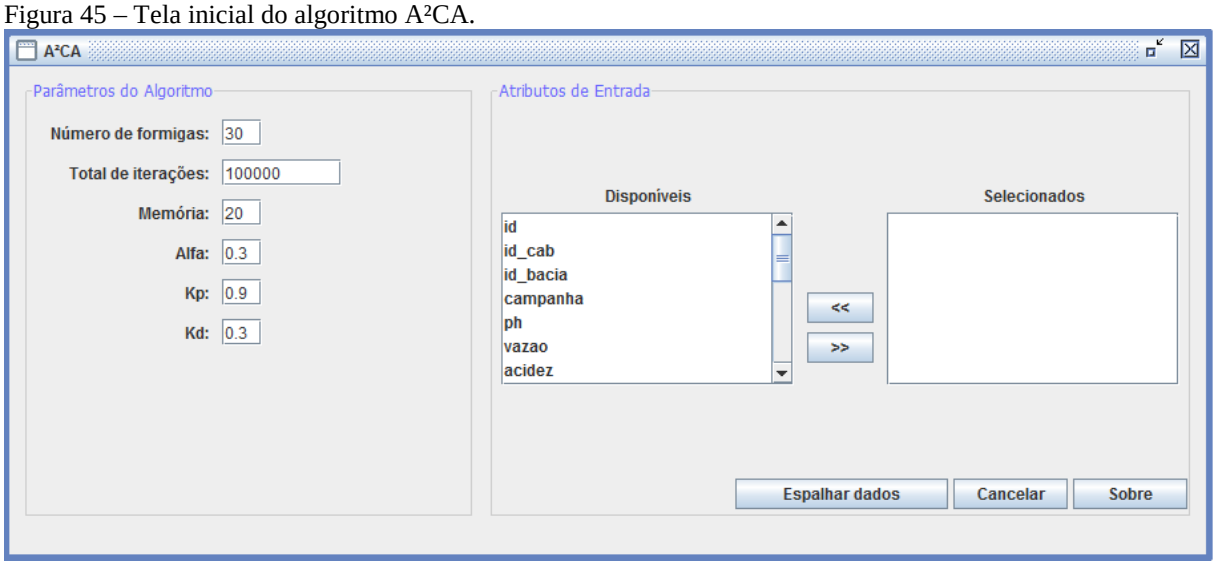
Fonte: Do autor.

Figura 44 – Tela inicial do algoritmo *Ant Based Clustering*.



Fonte: Do autor.

Figura 45 – Tela inicial do algoritmo A²CA.



Fonte: Do autor.

Na tabela 6 pode-se observar quais parâmetros devem ser informados em cada

algoritmo.

Tabela 6 - Parâmetros a serem informados em cada algoritmo.

Parâmetro	Definição	SACA	*ANT BASED CLUSTERING*	A²CA
Número de Formigas	Quantidade de formigas artificiais que irão trabalhar para agrupas os objetos.	X	X	X
Total de Iterações	Total de ciclos do algoritmo, sendo que um ciclo corresponde ao laço do tamanho do número de formiga.	X	X	X
Memória	Quantidade de posição que uma formiga pode gravar quando for deixar um objeto para que quando outro objeto for pego ela possa comparar a similaridade e decidir se a posição gravada na memoria é uma boa posição para o objeto pego.	X	X	X
Alfa [0,1]	Parâmetro fundamental para um bom agrupamento. Quanto mais próximo de zero mais compacto ficara o *cluster*.	X	Aleatório	X
Kp [0,1]	Valor mínimo para a formiga pegar o objeto.	X	Não existe	X
Kd [0,1]	Valor mínimo para a formiga deixar o objeto.	X	Não existe	X
Campo de Visão (3,5 ou 7)	Numero de células que cada formiga pode perceber em sua vizinhança.	X	Aleatório	Aleatório
Número de Passos	Quantidade de celular que cada formiga pode percorrer.	X	Aleatório	Aleatório

Fonte: Do autor.

Os algoritmos estão com valores padrões em cada parâmetro, o usuário tem a possibilidade de alterar de acordo com a necessidade. Não existe na literatura uma heurística que informa os parâmetros ideais para os algoritmos implementados. Além disso, os algoritmos têm resultados que variam de execução a execução, portanto uma forma de avaliar a parametrização é alterar cada parâmetro e verificar o resultado gerado pelo algoritmo. Os autores do algoritmo SACA sugerem para os parâmetros k_p e k_d os valores 0.1 e 0.3, respectivamente. Para o algoritmo *Ant Based Clustering* a autora sugere usar sempre 10 formigas, já que o algoritmo A²CA respeita os parâmetros sugeridos pelos autores do algoritmo SACA, ou seja, os valores de 0.1 e 0.3 para os parâmetros k_p e k_d, respectivamente. Os demais parâmetros têm de ser ajustados verificando os resultados do algoritmo.

Nos teste realizados nessa pesquisa foi constatado que valores altos para o parâmetro *Número de Formigas* tende a deixar o algoritmo mais lento e não beneficia o agrupamento, o ideal é que esse valor seja 10, essa regra aplica-se apenas para o algoritmo *Ant Based Clustering*, para o algoritmo SACA e A²CA o valor tem de ser testado e ajustado. A memória esta ligada a formação rápida de *clusters*, porém um valor muito elevado pode prejudicar o desempenho do algoritmos considerando que todas as formigas quando estão se

movimentando verificam cada informação gravada na memória para pode decidir a sua direção.

O valor de Alfa α é informado no algoritmo SACA e no A²CA, esse parâmetro é um valor no intervalo de [0,1] e é crucial para o bom desempenho do algoritmo, um valor muito alto pode gerar *clusters* grandes porém com dados que não possuem similaridade, já valores muito baixos o algoritmo pode gerar muitos *clusters* pequenos. O algoritmo *Ant Based Clustering* possui sua própria lógica de adaptação de Alfa α ,

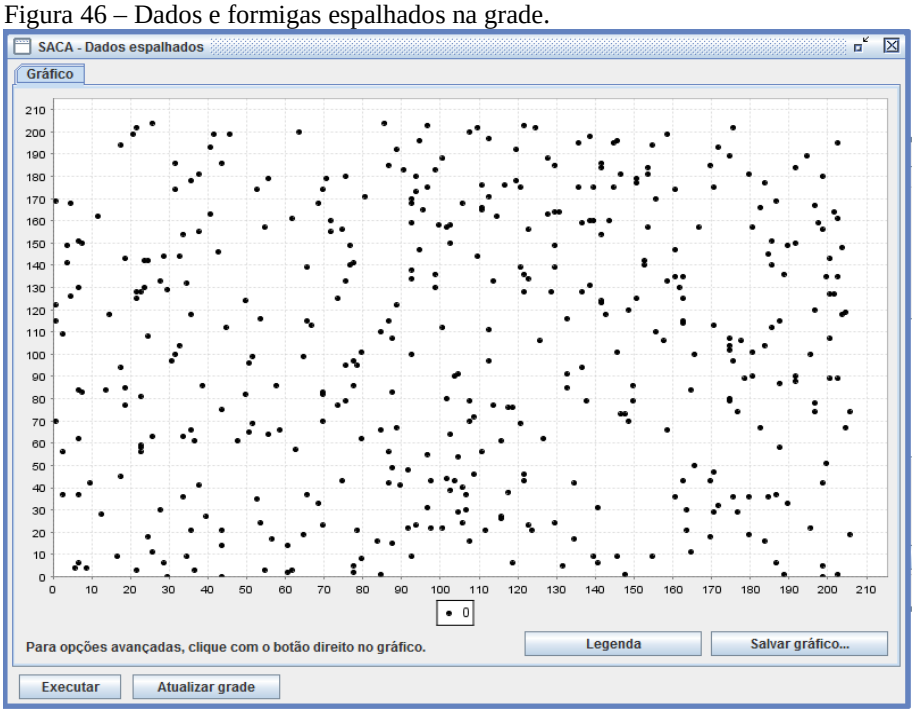
Os valores de k_p e k_d só não são informados no algoritmo *Ant Based Clustering*, pois o cálculo da probabilidade de pegar e deixar um objeto descarta esses dois parâmetros. No algoritmo SACA pode-se usar para inicio os valores sugeridos pelos autores, já para o A²CA o valor de k_p possue uma lógica de resfriamento. Como apresentado na seção 4.3, os autores não sugerem valor para ele, nos testes realizados nessa pesquisa o valor utilizado para k_p foi 0.9 e para k_d o mesmo valor sugerido para o SACA, ou seja, 0.3.

O parâmetros *Campo de Visão* também só é informado no algoritmo SACA, pois nos demais algoritmos esse parâmetro é aleatório e eles possuem lógicas que o alteram, conforme o andamento do algoritmo. Os valores mais comuns para o *Campo de Visão* são 3, 5 e 7, o valor mínimo para o *Campo de Visão* é 3 se for necessário um campo de visão maior esse novo valor deve ser o valor mínimo 3 mais 2, ou seja, 5 e assim por diante sempre incrementando 2 unidades.

O *Número de Passos* determina a velocidade das formigas, esse valor é aleatório em todos os algoritmos. Cada formiga possui o seu valor, dessa forma há uma mais rápidas e outras mais lentas, ou seja, algumas formigas vão trabalhar muito mais no local onde elas foram geradas inicialmente e outras irã percorrer toda a grade em menos iterações.

Após a parametrização deve-se espalhar os dados e as formigas aleatoriamente na grade. Para isso, na *Shell* Orion, no quadrante direito se encontra o botão *Espalhar Dados* que irá abrir a janela com os objetos e as formigas espalhados na grade. Os dados ainda não pertencem a nenhum *cluster*, então são representados na cor preta, as formigas ocupam o mesmo espaço do dado que ela está carregando por isso elas não aparecem, dando preferência aos dados (figura 46).

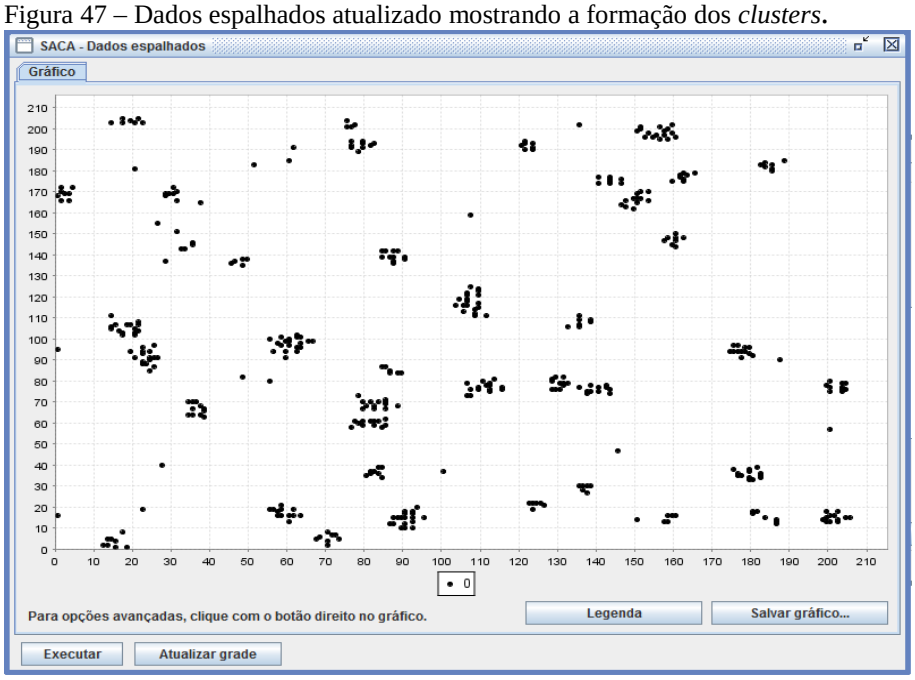
Figura 46 – Dados e formigas espalhados na grade.



Fonte: Do autor.

O algoritmo é inicializado quando acionado o botão *Executar* da tela de *Dados espalhados*, para visualizar a formação de *cluster* pode-se acionar o botão *Atualizar grade*, este recalcula a grade e mostra o andamento do algoritmo na formação de *cluster (*figura 47). Para não prejudicar o desempenho do módulo da *Shell* Orion essa funcionalidade foi implementada para recalcular a grade apenas quando o usuário desejar ver a formação de *clusters*.

Figura 47 – Dados espalhados atualizado mostrando a formação dos *clusters*.



Fonte: Do autor.

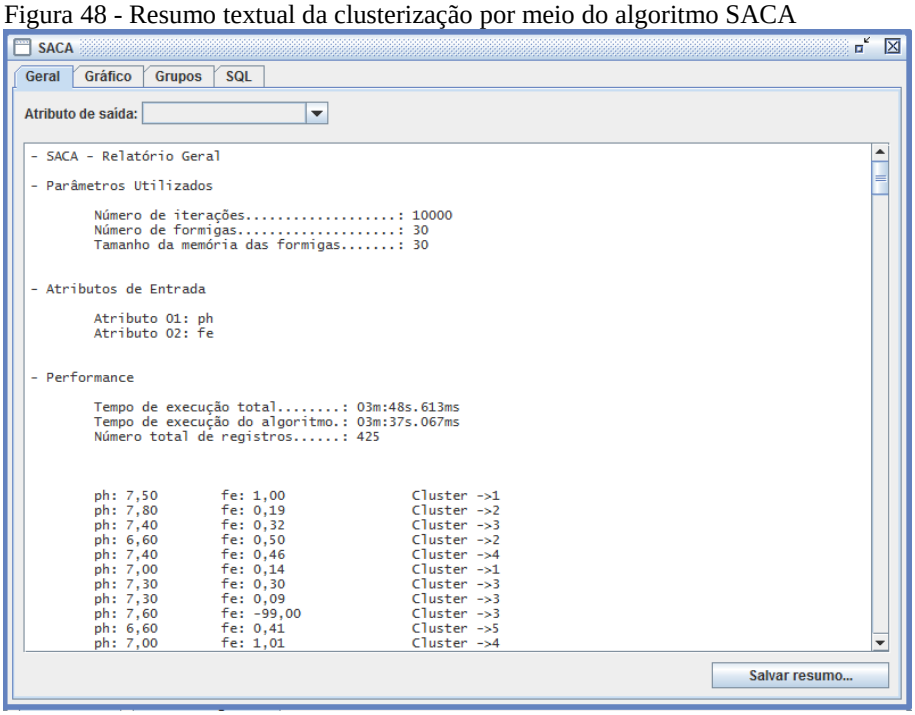
Os algoritmos baseados no SACA, *Ant Based Clustering* e A²CA, não determinam os *clusters*, eles fazem um agrupamento de dados graficamente. Para que seja possível avaliar os resultados obtidos pelos algoritmos, uma lógica foi implementada para determinar e classificar cada grupo de dados em *clusters*.

A lógica implementada percorre todos os objetos na grade e verifica a vizinhança de cada um, se o objeto estiver sozinho, sem vizinhança ou esta for muito baixa esse objeto é classificado como *outlier*, pois o algoritmo não conseguiu encaixar ele em nenhum grupo. Assim que uma vizinhança aceitável é detectada para um objeto e este não pertence a nenhum *cluster*, ele é classificado com um *cluster* e toda a sua vizinhança também é considerada como pertencente ao *cluster*. Quando um objeto já está classificado como um *cluster* a lógica implementada ignora esse objeto e verifica o próximo objeto sem *cluster*. Essa lógica apresentou bom desempenho com a base de dados de testes, conseguindo classificar todos os grupos de dados em *clusters*.

A ferramenta *Shell* Orion disponibiliza a análise dos resultados por meio de resumo, árvore e gráfico, além de permitir que o usuário exporte o resultado obtido pela clusterização para um arquivo SQL. No caso dos algoritmos SACA, *Ant Based Clustering* A²CA os *outliers* também são incluídos neste arquivo.

Os resultados por meio de resumo podem ser visualizados na figura 48, esse resumo textual possui informações sobre os atributos de entrada utilizados, o tempo de execução do algoritmo e o *cluster* que cada ponto da base de dados está inserido ou se é um *outlier*. No final do resumo tem-se os índices de validação, Índice de *Dunn* e *C-Index*, que tem por objetivo avaliar a qualidade das partições encontradas pelos algoritmos implementados.

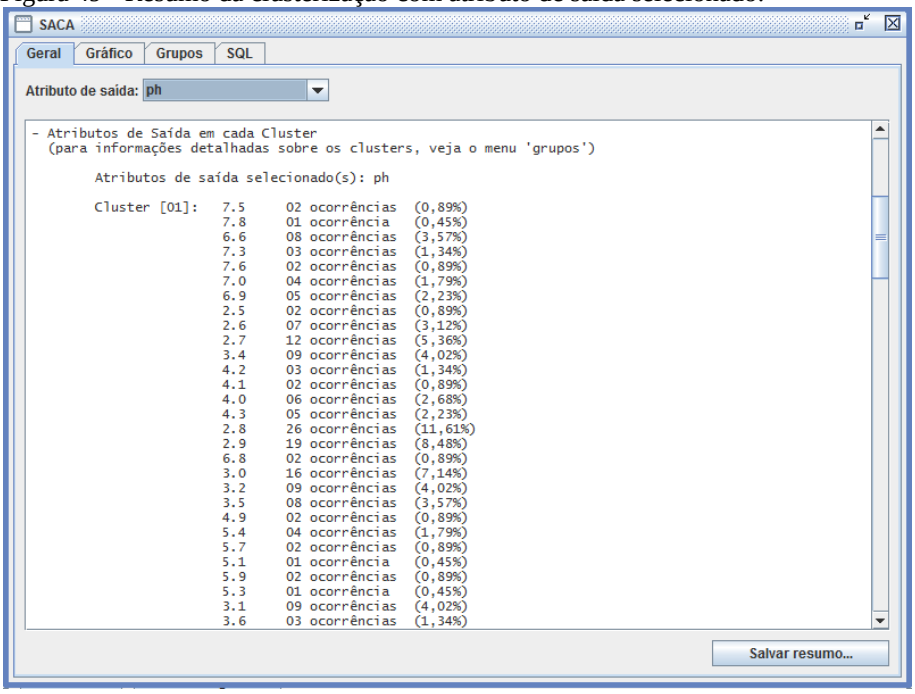
Figura 48 - Resumo textual da clusterização por meio do algoritmo SACA



Fonte: Do autor.

O resumo também pode ser apresentado de modo detalho caso o usuário desejar. No canto superior esquerdo da janela no campo *Atributo de Saída*, pode-se selecionar algum atributo de saída específico (figura 49).

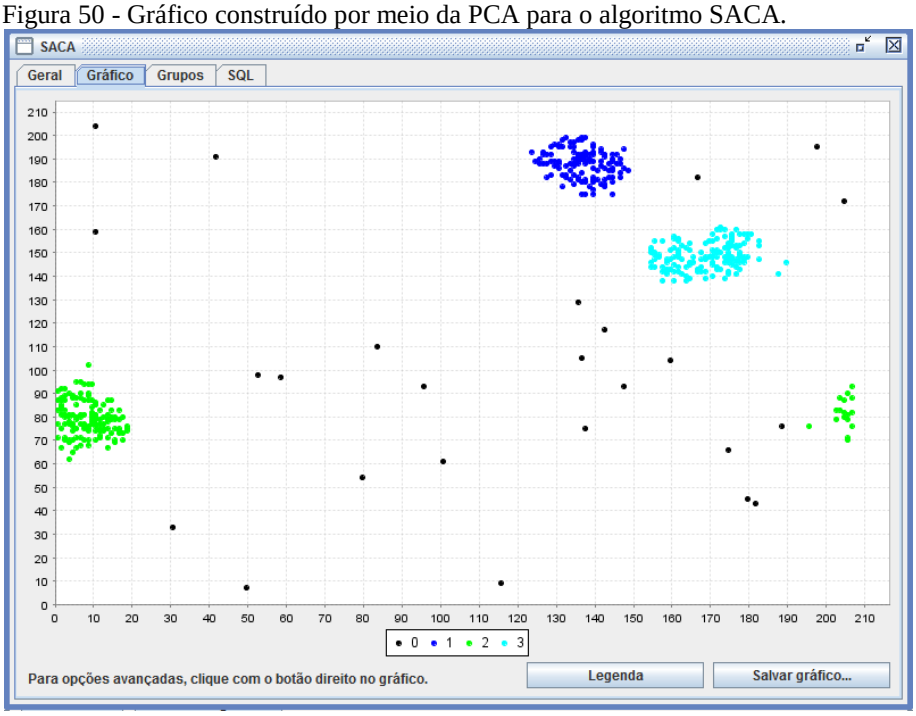
Figura 49 - Resumo da clusterização com atributo de saída selecionado.



Fonte: Do autor.

A visualização por gráfico pode ser observada na figura 50. O gráfico utiliza a técnica *Principal Component Analysis* (PCA), essa técnica tem por objetivo transformar uma base de dados com *n* dimensões em uma matriz de duas dimensões, por meio de sucessivas decomposições nos dados, dessa forma a projeção dos elementos no gráfico é possível. No caso dessa pesquisa, os elementos são representados de acordo com suas posições *x* e *y* finais.

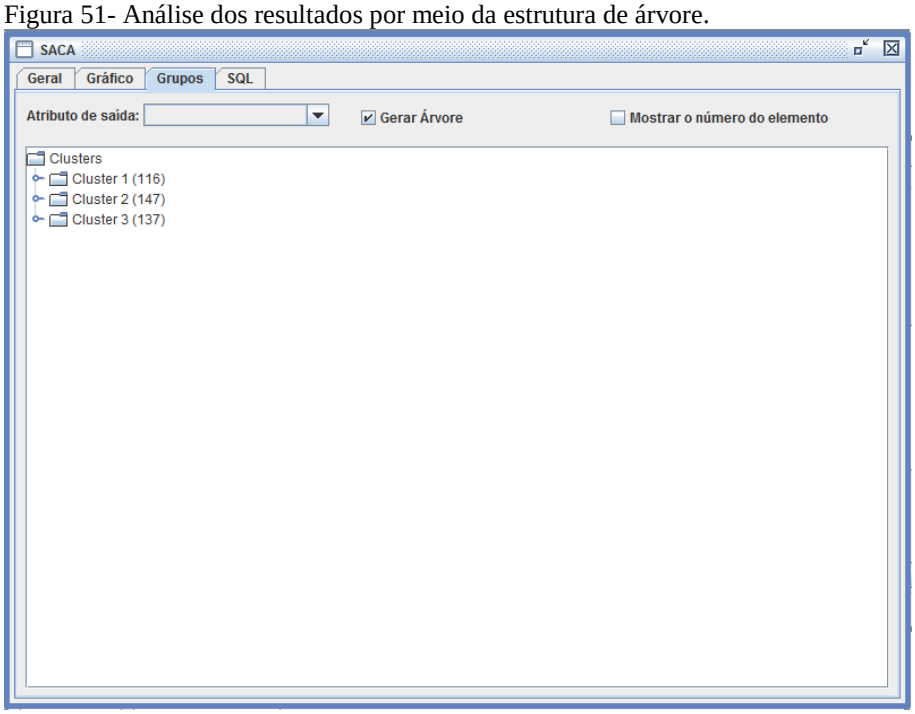
Figura 50 - Gráfico construído por meio da PCA para o algoritmo SACA.



Fonte: Do autor.

Por meio de uma estrutura de árvore é possível também ter uma visualização detalhada dos resultados gerados pelos algoritmos implementados. Por motivo de melhorar o desempenho do módulo desenvolvido, o campo *Gerar Árvore* na tela vem por padrão desmarcado, dessa forma o modulo não gera a árvore sempre que os resultados são gerados e sim quando o usuário deseja visualizar por meio da estrutura de árvore, para isso o mesmo terá que selecionar o campo *Gerar Arvore* (figura 51).

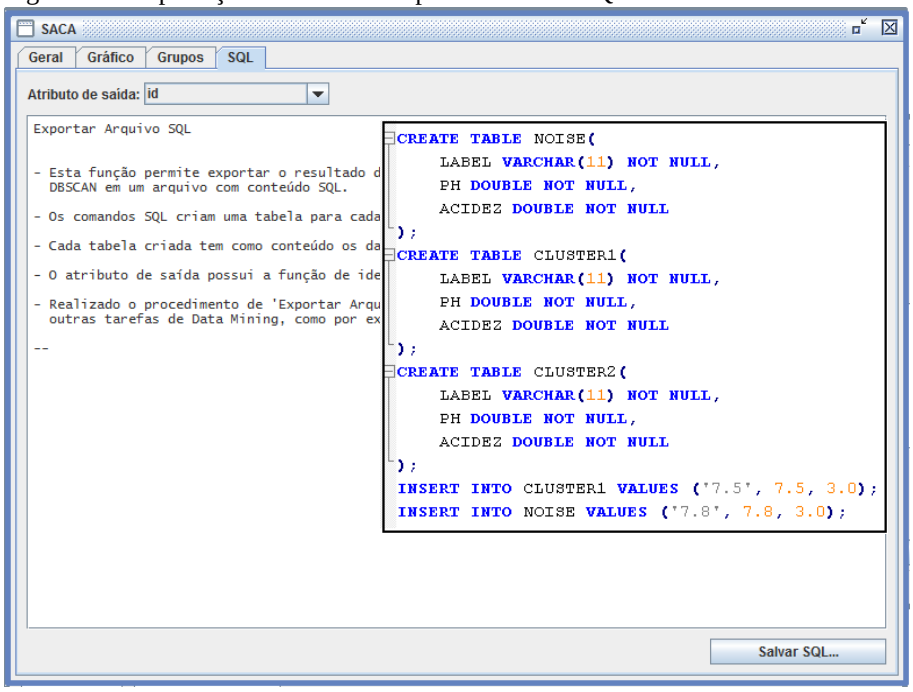
Figura 51- Análise dos resultados por meio da estrutura de árvore.



Fonte: Do autor.

Os resultados da clusterização podem ser usados posteriormente em outras tarefas. A *Shell* Orion possibilita a exportação dos resultados gerados para o formato SQL (figura 52).

Figura 52 - Exportação dos resultados para o formato SQL.



Fonte: Do autor.

A seguir são apresentados os resultados obtidos e a discussão dos testes que foram efetuados após a implementação a fim de comprovar os resultados e verificar os tempos do algoritmo.

4.3 RESULTADOS OBTIDOS

Na realização de teste com os algoritmo SACA, *Ant Based Clustering* e A²CA para clusterização de dados, foi utilizado um microcomputador com sistema operacional Windows 7, processador Intel Core i5 2.53 GHz e 4GB de memória RAM.

Não foi possível realizar uma comparação entre os algoritmos implementados na *Shell* Orion e outra ferramenta de *data mining* devido a dificuldade de encontrar o algoritmo SACA implementado. Essa dificuldade se justifica pelo fato do algoritmo SACA não ser um algoritmo de clusterização já definido, ele é um método de agrupamento de dados que tem bons resultados para clusterização de dados (LUMER; FAITA, 1994, tradução nossa).

Os resultados obtidos pela *Shell* Orion foram analisados considerando o funcionamento do módulo pela análise dos *clusters*, por meio de índices de validação. Também é apresentada as vantagens que os algoritmos A²CA e *Ant Based Clustering* possuem em relação ao SACA. Para todos os testes foi utilizada a base de dados referente ao monitoramento de indicadores ambientais, com os dados da bacia hidrográfica do rio Urussanga.

Como já foi citado, nessa pesquisa os algoritmos implementados geram resultados diferentes a cada execução, portanto os algoritmos foram executados 10 vezes e os seus resultados analisados.

4.3.1 Teste de Qualidade do Agrupamento Formado pelos Algoritmos SACA, *Ant Based Clustering* e A²CA

A base de dados utilizada para avaliação dos algoritmos SACA, *Ant Based Clustering* e A²CA, refere-se a dados de monitoramento da bacia do rio Urussanga, composta por 425 registros coletados ao longo de 37 pontos de monitoramentos espalhados por essa bacia. Essa base de dados pertence a área de engenharia ambiental referente ao monitoramento de bacias hidrográficas da região carbonífera.

Os atributos da base de dados de entrada selecionados foram os índices de pH e a concentração de ferro presentes nas amostras obtidas ao longo da bacia do rio Urussanga, o

objetivo dos testes é analisar a qualidade dos *clusters* formados pelos algoritmos por meio dos índices de validação, a quantidade de *cluster* que cada algoritmo forma e o tempo que cada um necessita para formar os *clusters*, além da demonstração da variação dos resultados por meio do desvio padrão[8], média[9] e coeficiente de variação[10].

4.3.1.1 Resultados obtidos pelo algoritmo SACA

Na avaliação dos resultados gerados pelo algoritmo SACA, os parâmetros utilizados foram os seguintes, sendo que na execução dos três algoritmos os atributos de entrada foram os mesmos, a fim de se comparar os resultados:

a) **número de formigas**: 50;
b) **total de iterações** : 2500;
c) **memória:** 50;
d) **alfa**: 0.3;
e) **kp**: 0.1;
f) **kd**: 0.3;
g) **campo de visão:** 5;
h) **atributo de entrada:** pH e fe.

Para definir os parâmetros foram necessários vários testes até encontrar a parametrização ideal para a base de dados. Dessa forma, na tabela 7 pode-se observar os resultados que o algoritmo SACA gerou durante os testes.

[8] Define o quanto os valores dos quais se extraiu a média são próximos ou distantes da própria média (SNEDECOR; COCHRAN, 1980, tradução nossa).
[9] Utilizada para determinar o valor que aponta para onde mais se concentram os dados de uma lista de valores (SNEDECOR; COCHRAN, 1980, tradução nossa).
[10] Possibilita estabelecer faixa de valores que orientam os pesquisadores sobre a validade de seus experimentos (SNEDECOR; COCHRAN, 1980, tradução nossa).

Tabela 7 – Resultado de cada teste realizado com o algoritmo SACA.

(continua)

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
1	6	21	00m: 16s: 422ms	1,3518	*Cluster* 1 = 0,0294 *Cluster* 2 = 0,1631 *Cluster* 3 = 0,1966 *Cluster* 4 = 0,0912 *Cluster* 5 = 0,0800 *Cluster* 6 = 0,1355
2	8	32	00m: 51s: 382ms	1,3496	*Cluster* 1 = 0,0800 *Cluster* 2 = 0,1011 *Cluster* 3 = 0,0760 *Cluster* 4 = 0,1121 *Cluster* 5 = 0,0719 *Cluster* 6 = 0,2216 *Cluster* 7 = 0,0550 *Cluster* 8 = 0,0345
3	7	18	00m: 11s: 232ms	1,3498	*Cluster* 1 = 0,2404 *Cluster* 2 = 0,0494 *Cluster* 3 = 0,0728 *Cluster* 4 = 0,1018 *Cluster* 5 = 0,1084 *Cluster* 6 = 0,0950 *Cluster* 7 = 0,0708
4	6	20	00m: 18s: 341ms	1,3505	*Cluster* 1 = 0,1470 *Cluster* 2 = 0,1990 *Cluster* 3 = 0,0824 *Cluster* 4 = 0,7420 *Cluster* 5 = 0,1198 *Cluster* 6 = 0,0730
5	5	26	00m: 60s: 459ms	1,3449	*Cluster* 1 = 0,1396 *Cluster* 2 = 0,1658 *Cluster* 3 = 0,0718 *Cluster* 4 = 0,1354 *Cluster* 5 = 0,1023
6	5	19	00m: 28s: 632ms	1,3649	*Cluster* 1 = 0,1505 *Cluster* 2 = 0,0545 *Cluster* 3 = 0,2645 *Cluster* 4 = 0,1219 *Cluster* 5 = 0,0348
7	6	22	00m: 56s: 341ms	1,5061	*Cluster* 1 = 0,1094 *Cluster* 2 = 0,2077 *Cluster* 3 = 0,0413 *Cluster* 4 = 0,1221 *Cluster* 5 = 0,0911 *Cluster* 6 = 0,0692
8	7	20	00m: 08s: 532ms	1,3519	*Cluster* 1 = 0,1405 *Cluster* 2 = 0,1161 *Cluster* 3 = 0,1972 *Cluster* 4 = 0,0218 *Cluster* 5 = 0,0714 *Cluster* 6 = 0,1847 *Cluster* 7 = 0,0152
9	6	20	01m: 01s: 231ms	1,3515	*Cluster* 1 = 0,2117 *Cluster* 2 = 0,0799 *Cluster* 3 = 0,0634 *Cluster* 4 = 0,1004 *Cluster* 5 = 0,0779 *Cluster* 6 = 0,1187

Tabela 7 – Resultado de cada teste realizado com o algoritmo SACA.

(conclusão)

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
10	4	21	00m: 57s: 789ms	1,3498	*Cluster* 1 = 0,1903 *Cluster* 2 = 0,1753 *Cluster* 3 = 0,1134 *Cluster* 4 = 0,0699

Fonte: Do autor.

Nos testes pode-se observar que a partir de um determinado momento o algoritmo SACA não forma *cluster*, ele apenas movimenta até o fim das iterações os objetos que não conseguiu agrupar. Isso se justifica pelo fato do algoritmo SACA não se adapta de acordo com a formação dos grupos, ou seja, não existe parâmetro que se altera conforme o andamento do algoritmo, assim, grupos formados dificilmente são desfeitos e reagrupados em outros grupos. Essa falta de adaptação também prejudica o tempo de processamento do algoritmo, pois os objetos agrupados dificilmente são pegos e a formiga perde um tempo até achar um objeto livre que possa ser carregado.

A variação no número de *cluster* está ligada as ações aleatórias do algoritmo, como o algoritmo dificilmente desfaz os grupos formados, as posições aleatórias no inicio do algoritmo podem influenciar na formação dos *clusters*, pois os objetos similares podem ser colocados, aleatoriamente, próximos entre si na grade e no início do algoritmo eles já formam um grupo. O número de objetos não classificados explica-se devido ao fato das formigas estarem carregando esses objetos quando o algoritmo terminou a sua execução.

Na tabela 8 pode-se observar o desvio padrão, média e coeficiente de variação dos resultados obtidos pelo algoritmo SACA.

Tabela 8 – Variação dos resultados do algoritmo SACA.

	Clusters	Pontos não Classificados	Índice de *Dunn*
Desvio Padrão	1,9544	3,9357	0,0465
Média	6,1	21,9	1,367
Coeficiente de Variação	32,0393%	17,9712%	3,4081%

Fonte: Do autor.

Pode-se observar na tabela 8 que houve uma grande variação nos números de *clusters* gerados pelo SACA, comprovando que o mesmo gera resultados diferentes a cada execução. Na tabela 7 pode-se observar o índice de validação *C-index*, que avalia internamente cada *cluster*, os valores destes foram próximo de zero indicando que mesmo tendo variação nos números de *clusters*, estes são formados com objetos similares.

4.3.1.2 Resultados obtidos pelo algoritmo *Ant Based Clustering*

Na avaliação dos resultados gerados pelo algoritmo *Ant Based Clustering*, os parâmetros utilizados foram os seguintes:

a) **número de formigas**: 10;

b) **total de iterações**: 100000;

c) **memória:** 10;

d) **atributo de entrada:** pH e fe.

Os parâmetros *Número de Formiga* e *Memória* foram os mesmos sugeridos pela autora do algoritmo, apenas o parâmetro *Total de Iterações* foi ajustado de acordo com os testes realizados. Na tabela 9 pode-se observar os resultados gerados pelo algoritmo nas 10 execuções:

Tabela 9 – Resultado de cada teste realizado com o algoritmo *Ant Based Clustering*.

(continua)

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
1	9	13	02m: 12s: 135ms	1,3498	*Cluster* 1 = 0,1488 *Cluster* 2 = 0,1216 *Cluster* 3 = 0,0847 *Cluster* 4 = 0,1205 *Cluster* 5 = 0,1042 *Cluster* 6 = 0,1340 *Cluster* 7 = 0,1381 *Cluster* 8 = 0,0800 *Cluster* 9 = 0,0690
2	5	12	02m: 50s: 798ms	1,3518	*Cluster* 1 = 0,0998 *Cluster* 2 = 0,1712 *Cluster* 3 = 0,2796 *Cluster* 4 = 0,1108 *Cluster* 5 = 0,0897
3	3	9	05m: 42s: 972ms	1,8806	*Cluster* 1 = 0,4033 *Cluster* 2 = 0,1473 *Cluster* 3 = 0,0976
4	8	11	02m: 23s: 564ms	1,3495	*Cluster* 1 = 0,1076 *Cluster* 2 = 0,1372 *Cluster* 3 = 0,0633 *Cluster* 4 = 0,2800 *Cluster* 5 = 0,0940 *Cluster* 6 = 0,1515 *Cluster* 7 = 0,1777 *Cluster* 8 = 0,1820
5	8	10	02m: 34s: 673ms	1,3498	*Cluster* 1 = 0,0911 *Cluster* 2 = 0,2236 *Cluster* 3 = 0,1027 *Cluster* 4 = 0,0270 *Cluster* 5 = 0,1846 *Cluster* 6 = 0,1222 *Cluster* 7 = 0,0901 *Cluster* 8 = 0,2122
6	6	10	02m: 34s: 341ms	1,3495	*Cluster* 1 = 0,1616 *Cluster* 2 = 0,2572 *Cluster* 3 = 0,1118 *Cluster* 4 = 0,1155 *Cluster* 5 = 0,0797 *Cluster* 6 = 0,0386
7	5	10	02m: 56s: 562ms	1,3498	*Cluster* 1 = 0,1324 *Cluster* 2 = 0,1753 *Cluster* 3 = 0,1474 *Cluster* 4 = 0,1040 *Cluster* 5 = 0,2117
8	5	12	01m: 28s: 113ms	1,3498	*Cluster* 1 = 0,2209 *Cluster* 2 = 0,0470 *Cluster* 3 = 0,1313 *Cluster* 4 = 0,1275 *Cluster* 5 = 0,1772
9	7	10	01m: 34s: 931ms	1,3495	*Cluster* 1 = 0,1796 *Cluster* 2 = 0,1541 *Cluster* 3 = 0,2223 *Cluster* 4 = 0,1206 *Cluster* 5 = 0,1436 *Cluster* 6 = 0,0816 *Cluster* 7 = 0,0210

Tabela 9 – Resultado de cada teste realizado com o algoritmo *Ant Based Clustering*.

(conclusão)

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
10	7	14	01m: 51s: 239ms	1,3515	*Cluster* 1 = 0,1899 *Cluster* 2 = 0,0260 *Cluster* 3 = 0,1469 *Cluster* 4 = 0,2005 *Cluster* 5 = 0,1059 *Cluster* 6 = 0,0755 *Cluster* 7 = 0,1028

Fonte: Do autor.

Nos testes pode-se observar que o algoritmo necessita de muitas iterações para começar a formar *clusters*. A utilização de um baixo *Campo de Visão* e a alteração das equações k_p e k_d do algoritmo possibilita a formação de grupos bem compactos, porém o tempo de formação dos grupos é maior comparando com o algoritmo SACA.

A adaptação do Alfa α tem efeitos positivos para o algoritmo, pois com essa adaptação possibilita que elementos de um grupo possam ser analisados novamente e movidos para outros grupos, onde a vizinhança é mais similar a ele conforme vão passando as iterações.

No momento que o número de iteração do algoritmo ultrapassa o valor de inicio e o algoritmo começa a utilizar a outra função vizinhança já apresentada, acontece um espalhamento dos grupos, e quando termina essa etapa os grupos são formados novamente, porém em lugares diferentes.

Entre as vantagens apresentadas pelo algoritmo *Ant Based Clustering* em relação ao SACA, pode-se destacar a facilidade de parametrização, pois o mesmo necessita somente da informação dos parâmetros *Número de Formigas*, *Total de Iterações* e *Memória*. A adaptação de Alfa α apresenta uma vantagem considerável, pois elimina o problema de determinar esse parâmetro que é crucial na boa formação de grupos.

Na tabela 10 pode-se observar o desvio padrão, média e coeficiente de variação dos resultados obtidos pelo algoritmo.

Tabela 10 – Variação dos resultados do algoritmo *Ant Based Clustering*.

	Clusters	Pontos não Classificados	Índice de *Dunn*
Desvio Padrão	1,7349	1,5132	0,1591
Média	6,3	11,1	1,4031
Coeficiente de Variação	27,5381%	13,6324%	11,3391%

Fonte: Do autor.

Analisando a tabela 10, pode-se verificar que o algoritmo *Ant Based Clustering* tem uma variação de números de *clusters* e pontos não classificados um pouco menor que o SACA. Na tabela 9, comparando com o SACA, o tempo de execução do algoritmo aumentou, isso se justifica pelo fato do algoritmo *Ant Based Clustering* precisar de muito mais iterações que o algoritmo SACA. Já o índice de validação *C-Index* não foi melhorado no *Ant Based Clustering*, porém o objetivo desse algoritmo é diminuir a quantidade de parâmetros de entrada.

4.3.1.3 Resultados obtidos pelo algoritmo A²CA

Na avaliação dos resultados gerados pelo algoritmo A²CA, os parâmetros utilizados foram os seguintes:

a) **número de formigas**: 30;
b) **total de iterações** : 8000;
c) **memória:** 20;
d) **alfa**: 0.3;
e) **kp**: 0.9;
f) **kd**: 0.3;
g) **atributo de entrada:** pH e fe.

Para definição dos parâmetros foram realizados vários testes a fim de se determinar os ideais para esse algoritmo. Na tabela 11 pode-se observar os resultados obtidos nos testes pelo algoritmo A²CA.

Tabela 11 – Resultado de cada teste realizado com o algoritmo A²CA.

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
1	6	28	08m: 20s: 341ms	1,3495	*Cluster* 1 = 0,2192 *Cluster* 2 = 0,0667 *Cluster* 3 = 0,0229 *Cluster* 4 = 0,0892 *Cluster* 5 = 0,0947 *Cluster* 6 = 0,8192
2	5	29	02m: 10s: 893ms	1,3498	*Cluster* 1 = 0,1270 *Cluster* 2 = 0,1027 *Cluster* 3 = 0,1555 *Cluster* 4 = 0,0710 *Cluster* 5 = 0,0946
3	4	22	01m: 29s: 741ms	1,3518	*Cluster* 1 = 0,1298 *Cluster* 2 = 0,0557 *Cluster* 3 = 0,0386 *Cluster* 4 = 0,2284
4	5	20	03m: 12s: 435ms	1,3498	*Cluster* 1 = 0,0644 *Cluster* 2 = 0,2131 *Cluster* 3 = 0,0841 *Cluster* 4 = 0,0245 *Cluster* 5 = 0,0795
5	6	18	04m: 32s: 345ms	1,3495	*Cluster* 1 = 0,1023 *Cluster* 2 = 0,1266 *Cluster* 3 = 0,1097 *Cluster* 4 = 0,0650 *Cluster* 5 = 0,0629 *Cluster* 6 = 0,1986
6	4	22	03m: 10s: 531ms	1,3495	*Cluster* 1 = 0,0863 *Cluster* 2 = 0,0517 *Cluster* 3 = 0,1141 *Cluster* 4 = 0,2235
7	5	18	06m: 59s: 313ms	1,3498	*Cluster* 1 = 0,0168 *Cluster* 2 = 0,1059 *Cluster* 3 = 0,0987 *Cluster* 4 = 0,0512 *Cluster* 5 = 0,2109
8	4	25	02m: 08s: 740ms	1,3518	*Cluster* 1 = 0,1005 *Cluster* 2 = 0,2577 *Cluster* 3 = 0,0942 *Cluster* 4 = 0,0186
9	5	25	02m: 10s: 356ms	1,3498	*Cluster* 1 = 0,2203 *Cluster* 2 = 0,0570 *Cluster* 3 = 0,0735 *Cluster* 4 = 0,0596 *Cluster* 5 = 0,0977
10	6	22	02m: 28s: 492ms	1,3498	*Cluster* 1 = 0,0233 *Cluster* 2 = 0,2470 *Cluster* 3 = 0,0475 *Cluster* 4 = 0,0268 *Cluster* 5 = 0,1023 *Cluster* 6 = 0,0596

Fonte: Do autor.

Esse algoritmo, apesar de necessitar dos mesmos parâmetros que o algoritmo SACA, desconsiderando o *Campo de Visão*, apresenta resultados bons em relação a o índice *C-index* que avalia a qualidade interna dos *clusters*, pois os valores apresentados são mais

próximos de zero. Isso graças a heurística de feromônio e a sua adaptação do campo de visão para cada formiga. Esse algoritmo, como o SACA, tende a formar muitos grupos no início, porém graças as suas adaptações pequenos grupos são desfeitos e seus elementos passam a pertencer a grupos maiores e que contenham objetos similares a ele.

Quanto maior o número de iteração do algoritmo mais compacto são os grupos formados por ele. Os objetos que estão nas limitações dos grupos tendem a serem aproximados do centro ou então são realocados para outro grupo onde é mais aceito. Todo esse processo justifica a boa formação de *clusters* do algoritmo.

Esse algoritmo apresentou tempo de processamento maior que o SACA devido ao número de iterações que foi utilizado para os testes. A quantidade de *cluster* variou menos que o SACA, isso por que o A²CA utiliza uma comunicação global entre as formigas, que é o uso do feromônio que indica que determinada região tem um grande ou pequeno número de objetos agrupados e também pela adaptação do campo de visão para cada formiga, onde elas passam a diferenciar pequenos e grandes grupos.

A principal vantagem desse algoritmo em reação as SACA é e boa qualidade da formação de grupos e a capacidade de eliminar os grupos pequenos através da heurística de feromônio. As adaptações propostas nesse algoritmo eliminam o problema do SACA em parar de formar grupos em determinadas situações e fazendo com que o os grupos formados fiquem melhores com o passar das iterações.

Na tabela 12 pode-se observar o desvio padrão, média e coeficiente de variação dos resultados obtidos pelo algoritmo A²CA.

Tabela 12 – Variação dos resultados do algoritmo A²CA.

	Clusters	**Pontos não Classificados**	**Índice de *Dunn***
Desvio Padrão	0,7745	3,618	0,0008
Média	5	22,9	1,3501
Coeficiente de Variação	15,49	15,7991	0,0592

Fonte: Do autor.

A tabela 12 demonstra que a quantidade de *clusters* gerados pelo A²CA é menor comparado com os dois algoritmos anteriores, os demais resultados também tiveram uma variação menor comprovando dessa forma o objetivo do algoritmo que é diminuir a variação dos resultados do algoritmo SACA. Além disso, observando a tabela 11 o A²CA apresentou os melhores resultados do índice de validação *C-Index*. Levando o desempenho em consideração pode-se concluir que o algoritmo A²CA obteve o melhor desempenho.

CONCLUSÃO

A aquisição e armazenamento digital de dados facilitado pelo avanço da tecnologia geram grandes repositórios de dados. *Data mining* tem a tarefa de extrair os novos conhecimentos desses repositórios e auxiliar na tomada de decisões. Esse processo é de fundamental importância para análise de bases de dados.

Essa pesquisa fundamentou-se no entendimento dos principais conceitos relacionados a tarefa de clusterização, utilizando um método baseado em inteligência de enxame, por meio do algoritmo SACA, que se trata de um estudo inspirado no comportamento coletivo de formigas na tarefa de organização de cemitérios, onde as formigas fazem grupos de corpos com semelhanças entre si, dessa forma possibilita o algoritmo a formar grupos de dados similares espalhados em uma grade.

Dentre as dificuldades encontradas durante a realização dessa pesquisa, podem ser destacadas a falta de ferramentas que contenham o algoritmo SACA implementado, para que a comparação entre o algoritmo SACA implementado na *Shell* Orion e outra ferramenta fosse possível, a implementação de uma lógica para classificar como *clusters* os grupos formados pelo algoritmo SACA, e a própria implementação do algoritmo que tem uma série de observações que devem ser cuidadosamente analisadas, porém, essa última foi superada com o auxílio da modelagem matemática do algoritmo e muito estudo sobre as possíveis situações das formigas no decorrer das iterações do algoritmo.

Os objetivos da pesquisa foram atingidos, mesmo com as dificuldades encontradas. Além do algoritmo proposto, mais duas variações do algoritmo SACA foram disponibilizadas na *Shell* Orion, são elas o algoritmo *Ant Based Clustering* e o algoritmo A²CA. Os módulos desenvolvidos foram testados e a qualidade das partições geradas pelos algoritmos foram analisadas em conjunto com os índices de validação, onde apresentaram resultados satisfatórios que comprovam o funcionamento dos algoritmos implementados.

O SACA possui a vantagem de não necessitar da informação inicial do número de grupos da base de dados, porém, a sua parametrização é complicada para um usuário não especialista. As adaptações do algoritmo SACA se propõem a diminuir essa quantidade de parâmetros e possibilitar a um usuário com pouca experiência ter bons resultados utilizando o algoritmo.

O algoritmo *Ant Based Clustering*, apresentou uma demora considerável na formação dos *clusters*, devido ao fato de necessitar de muito mais iterações do que os algoritmos SACA e A²CA. Esse algoritmo apresentou grupos bem compactos, porém o

resultado dos índices de validação não foram tão bons comparados aos demais. A vantagem desse algoritmo é que somente três parâmetros são necessários: número de formigas, total de iterações e tamanho de memória. Mas o fato de ele não utilizar os parâmetros k_p e k_d faz com que os grupos não sejam tão homogêneos como esperado. A sua adaptação do parâmetro Alfa α é a alteração que pode-se considerar de maior valor, pois elimina o problema do SACA em informar esse parâmetro, que é crucial para o bom funcionamento do algoritmo.

O algoritmo A²CA se diferenciou do SACA no fato de não ser tão sensível com os parâmetros informados devido a heurística de feromônio que o mesmo possui. Essa heurística possibilitou que os pequenos grupos que são formados durante o processo do algoritmo fossem desmanchados e seus elementos colocados onde existe concentração maior de feromônio, ou seja, em lugares onde os objetos similares estão em maior número. Ainda assim, a parametrização do mesmo não é simples, tanto é que os seus valores para teste foram encontrados devido a insistência em achar o melhor resultado. A adaptação do campo de visão proposta nesse algoritmo também é muito importante para a boa formação dos grupos. Pode-se concluir que esse algoritmo foi o que apresentou melhor resultado na qualidade interna dos *clusters* formados.

O SACA, se corretamente parametrizado, retorna grupos com índices de validações satisfatórios. É importante lembrar que ele é um método em estudo e ainda não está completamente definido, portanto ainda não é possível competir com algoritmos clássicos de clusterização.

Considerando a pesquisa realizada, algumas sugestões de trabalhos futuros são descritas a seguir, com o objetivo de dar continuidade ao desenvolvimento do projeto da *Shell Orion Data Mining Engine*:

a) encontrar outro critério de determinação dos vários parâmetros do SACA estudando uma interação entre os mesmos;
b) estudar e implementar uma heurística para auxiliar na determinação dos parâmetros do algoritmo;
c) implementar outras variações do SACA a fim de comparar os resultados entre eles e verificar a vantagem de cada um;
d) desenvolver uma nova lógica para classificar em *cluster* os grupos formados pelo algoritmo SACA e, dessa forma, também diminuir os pontos que não são classificados;

e) desenvolver um estudo sobre a possibilidade de utilizar as modificações propostas nos algoritmos *Ant Based Clustering* e A²CA para implementar um algoritmo onde utilize as melhores mudanças sugeridas nesses .

REFERÊNCIAS

ADRIANS, Pieter; ZANTINGE, Dolf. **Data mining**. England: Addison-Wesley, 1996.

AMORIM, Thiago. **Conceitos, técnicas, ferramentas e aplicações de Mineração de Dados para gerar conhecimento a partir de bases de dados**. 2006. 50 f.2006 (Graduação) - Curso de Ciência da Computação, Ufpe, Recife, 2006.

BARRETO, Alexandre Serra. **Considerações prévias à utilização empírica do Data-Mining.** In. SQL Magazine (versão digital). 2004a Disponível no link: http://www.sqlmagazine.com.br/Colunistas/AlexandreBarreto/02_DataMining.asp. Acesso em: 14 mai 2011.

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, pp. 181–189, 1994.

BERRY, J.A, LINOFF, Gordon S. **Mastering Data Mining**. New York: John Wiley & Sons; 2000.

BERRY, Michael J.; LINOFF, Gordon. **Data mining techniques:** for marketing, sales, and customer relationship management. 2. ed. Indianapolis: Wiley Publishing, 2004.

BEZDEK, James et al. **Fuzzy models and algorithms for pattern recognition and image processing**. New York: Springer, 2005.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. 2ª, edição. Rio de Janeiro; Elsevier, 2007.

BONABEU, E.; THERAULAZ, G.; DORIGO, M. **Swarm Intelligence:** from natuaral to artificial systems. New York, USA: Oxford University Press, 1999.

BOTELHO, Glenda Michele. **Seleção de características apoiada por mineração visual de dados**. 2011. 84f. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional)- Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, São Carlos, 2011. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-29032011-145542/> Acesso em: 28 mar 2012.

BORYCZKA, U. **Finding groups in data**: Cluster analysis with ants. Applied Soft Computing, v. 9, p. 61-70, 2009.

BROCK G.; PIHUR, V.; DATTA, S.; DATTA, S. (2008), **clValid**: Validation of Clustering Results. Disponível em: <http://www.louisville.edu/~g0broc01/research/> Acessado em: 28 mai 2010.

BUSCHINGER, A.; SCHULZ, A., **Leptothorax athabasca sp.n. (Hymenoptera: Formicidae)** from Alberta, Canada, an ant with an apparently restricted range., Myrmecologische Nachrichten (11), pp. 243-248: 244-247, 2008.

CARLANTONIO, Lando Mendonça di. **Novas metodologias para clusterização de dados**.2001. 148p. Dissertação (Mestrado) - Departamento de Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2001. Disponível em:<http://wwwp.coc.ufrj.br.teses/mestrado/inter/2002/teses/di%20CARLANTONIO_LM_02_t_M_int.pdf/> Acesso em: 03 mar 2012.

CASAGRANDE, Diego Paz. **O Módulo da Técnica de Associação pelo Algoritmo Apriori no desenvolvimento da Shell de Data Mining Orion**. 2005. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2005.

CASSETTARI JUNIOR, José Márcio. **O Método de Lógica Fuzzy pelo Algoritmo Gustafson-Kessel na Tarefa de Clusterização da Shell Orion Data Mining Engine**. 2008. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina. 2008.

COELHO, L.S., **Fundamentos, Potencialidades e aplicações de algoritmos Evolutivos**. SBMAC, São Carlos, SP:, 2003

CROTTI JUNIOR, Ademar. **O Método de Lógica Fuzzy pelos algoritmos Robust C-Prototypes e Unsupervised Robust C-Prototypes para a Tarefa de Clusterização na Shell Orion Data Mining Engine**. 2010. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina. 2010.

DENEUBOURG, J.L.,GOSS, S., FRANKS, N., SENDOVA, F. A., DETRAIN, C., CHRÉTIEN L., **The Dynamics of Collective Sorting Robot-Like Ants and Ant-Like Robots**. From Animals to Animats: Proc. of the 1st Int. Conf. on Simulationof Adaptive Behaviour. 1991.

DIAS, Carlos Rodrigo. **Algoritmos Evolutivos para o Problema de Clusterização de Grafos Orientados: Desenvolvimento e Análise Experimental**. Rio de Janeiro, 2004. Dissertação (Mestrado em Computação Aplicada e Automação) – Universidade Federal Fluminense - Programa de Pós-Graduação em Computação Aplicada e Automação. Área de concentração: Otimização e Inteligência Artificial. 2004.

DIAS, Madalena Maria. **Um modelo de formalização do processo de desenvolvimento de sistemas de descoberta de conhecimento em banco de dados.** – Florianópolis, 2001. Tese (Doutorado em Engenharia de Produção) - Universidade Federal de Santa Catarina – Programa de Pós-graduação em Engenharia de Produção, 2001. 212 p.

DORIGO, M.; DI CARO, G. **The Ant Colony Optimization Meta-Heuristic**. In: CORNE, D.; DORIGO, M.; GLOVER, F. (Ed.). New Ideas in Optimization. London: McGraw- Hill, 1999. p.11–32.

DORIGO, M.; STÜTZLE, T. **Ant colony optimization**. Cambridge: MIT Press, 2004

FAYYAD, Usama M.; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. **From Data Mining to Knowledge Discovery:** An overview**.** In: FAYYAD et al. Advances in Knowledge Discovery and Data Mining. G. Cambridge-Mass:AAAI/MIT Press, 1996.

FRANKS, N. R., SEDOVA, A.F., **Brood sorting by ants**: Distributing the workload over the work-surface. In: Behavioral Ecology and Sociobiology, 30:109–123, 1992.

FURLAN, José Davi, **Modelagem de objetos através da UML**, 1998, Makron Books.

GAN, Goujan; MA, Chaoqun; WU, Jianhong. **Data clustering**: theory, algorithms and applications. Philadelphia: SIAM, 2007.

GAVA, Éverton Marangoni. **O Método de Densidade pelo Algoritmo *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) na Tarefa de Clusterização da *Shell* Orion *Data Mining Engine***. 2011. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina. 2011.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel Lopes. **Data mining:** uma guia prático. Rio de Janeiro: Elsevier, 2005.
GTA - Grupo Técnico de Assessoramento (DNPM, CPRM, SIECESC, FATMA, MPF). **Terceiro Relatório de Monitoramento dos Indicadores Ambientais**. Versão 03. Revisão Final. Criciúma, 2009.

GUEDES, Gilleanes Thorwald Araujo. **UML**: uma abordagem prática. 3. ed. São Paulo: Novatec, 2008.

HAN, Jiawei; KAMBER, Micheline. **Data Mining: Concepts and Techniques**. The Morgan Kaufmann Series in Data Management Systens, Jim Gray, Series Editor; 2000.

HAN, J.; KAMBER, M., **Data mining: Concepts and Techniques**. San Francisco: Morgan Kaufmann, 2001.

HAN, J., KAMBER, M., **Data Mining Concepts and Techniques**. Morgan Kaufman Publishers, San Francisco, USA, 2006.

HANDL, J., **Ant-based methods for tasks of clustering and topographic mapping**: extensions, analysis and comparison with alternative techniques. Masters Thesis, Universität, Erlangen-Nürnberg, Erlangen, Germany, 2003.

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

JAIN, Anil K.; DUBES, Richard C. **Algorithms for clustering data**. Englewood Cliffs: Prentice Hall, 1988.

JAIN, Anil K.; MURTY, M. N.; FLYNN, P. J. **Data clustering:** a review. ACM Computing Surveys (CSUR), New York, p.264-323. sep. 1999

LAROSE, Daniel T. **Discovering Knowledge in Data:** An Introduction to Data Mining. New Jersey: John Wiley & Sons, 2005.

JOHNSON, E. **Emergência**: a vida integrada de formigas, cérebros, cidades e softwares. Rio de Janeiro, BR: Jorge Zahar, 2003. 232p.

LAURO, André Luís. **Agrupamento de dados utilizando algoritmo de colônia de formigas**. 2008. Dissertação de Mestredo – Pós-graduação e Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, 2008.

LOPES, H. S. **Fundamentos da Computação Evolucionária e Aplicações**. Bandeirantes, Paraná, 2006.

LUMER E.D., FAIETA B., **Diversity and adaptation in populations of clustering ants**. In Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, pages 501–508, 1994.

MARTINS, Denis Piazza. **O Algoritmo de Particionamento K-means na Tarefa de Clusterização da Shell Orion Data Mining Engine**. 2007. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2007.

MILLIGAN, Glenn W.; COOPER, Martha C. **An examination of procedures for determining the Number of clusters in a data set**. Psychometrika, Colombus, p. 159-179. Jun. 1985.

MOTTA, Custódio G. L. da. **Sistema Inteligente para Avaliação de Riscos em Vias de Transporte Terrestre**. 2004. Programa de Pós-graduação de Engenharia, Universidade Federal do Rio de Janeiro. Disponível em: <http://www.coc.ufrj.br/teses/mestrado/inter/2004/Teses/MOTTA_CGL_04_t_M_int.pdf> Acesso em: 26 mai 2012.

MONDARDO, Ricardo Lineburger. **O algoritmo C4.5 na tarefa de classificação da Shell Orion Data Mining Engine**. 2009. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina. 2009.

NAVEGA, Sergio. **Princípios Essenciais do Data mining**. In: Anais do Infoimagem 2002, Cenadem, Novembro. Intelliwise Research and Training. Agosto de 2002.

OCHI, Luiz Satoru; DIAS, Carlos Rodrigo; SOARES, Stênio S. Furtado. **Clusterização em Mineração de Dados**. ERI RJ/ES, Niterói, n. 4, p. 1-46, Nov. 2004.

OMRAN, M.; ENGELBRECHT, A. P.; SALMAN, A. **Particle swarm optimization method for image clustering**. International Journal of Pattern Recognition and Artificial Intelligence, [S.l.], v.19, n.3, 2005.

PATERLINI, Adriano Arantes. **Imersão de espaços métricos em espaços multidimensionais para indexação de dados usando detecção de agrupamentos**. 2011. 90f. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional)- Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, São Carlos, 2011. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-25042011 - 155810/> Acesso em: 23 mai 2012.

Graduação em Ciências Ambientais, Universidade do Extremo Sul Catarinense, Criciúma, 2007. Disponível em: <http://www.bib.unesc.net/biblioteca/sumario/000035/0000359F.pdf /> Acesso em: 23 mai 2012.

PELEGRIN, Diana Colombo. **A Tarefa de Classificação e o Algoritmo ID3 para Indução de Árvores de Decisão na Shell de Data Mining Orion**. 2005. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2005.

PEREGO, Daniel. **O Método de Lógica Fuzzy pelo algoritmo GATH-GEVA na tarefa de clusterização da Shell Orion Data Mining Engine**. 2009. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina. 2009.

RAIMUNDO, Lidiane Rosso. **O Algoritmo CART na Tarefa de Classificação da Shell Orion Data Mining Engine**. 2007. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2007.

REZENDE, Solange Oliveira. **Sistemas inteligentes:** fundamentos e aplicações. Barueri: Manole, 2005.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência artificial**. Rio de Janeiro: Elsevier, 2004.

SCOTTI, Ana Paula. **O Método de Redes Neurais com Função de Ativação de Base Radial para a Tarefa de Classificação na Shell Orion Data Mining Engine**. 2010. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina. 2010.

SCOSS, Anne Marie. **A Clusterização e Classificação no Processo de Data Mining para Análise do Desempenho Docente no Ensino de Graduação**. 2006. Monografia – Pós-graduação em Gerenciamento em Banco de Dados, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina. 2006.

SHELOKAR, P. S.; JAYARAMAN, V. K.; KULKARNI, B. D. **An ant colony approach for clustering**. Analytica Chimica Acta, [S.l.], v.509, n.2, p.187–195, 2004.

SHERAFAT, V., CASTRO L.N., HRUSCHKA, E.R. **TermitAnt**: An Ant Clustering algorithm improved by ideas from termite colonies. ICONIP: pp.1088-1093, 2004.

SIVANANDAM, S. N.; SUMATHI, S. **Introduction to data mining and its applications**. Berlim: Springer, 2006.

SNEDECOR, G.W.; COCHRAN, W.G. **Statistical methods**. 7th ed. Ames: The Iowa State University Press, 1980.

TAN, Pan-ning; STEINBACH, Michael; KUMAR, Vipin. Introdução ao *Data Mining*: Mineração de dados. Rio de Janeiro: Ciência Moderna, 2009.

VILLWOCK, R.; STEINER, M. T. A. **Análise do Desempenho de um Algoritmo de Agrupamento Modificado Baseado em Colônia de Formigas**. In: XLI Simpósio Brasileiro de Pesquisa Operacional, Porto Seguro, 2009. XLI Simpósio Brasileiro de Pesquisa Operacional. Porto Seguro: SOBRAPO, 2009.

VIZINE, A.L, CASTRO, L.N., HRUSCHKA, E.R., GUDWIN, R.R.. **Towards Improving Clustering Ant**: An Adaptative Ant Clustering Algorithm..29:143-154, 2005.

WILSON, E. O. **Pheidole in the New World**: A dominant, hyperdiverse ant genus. Harvard University Press, 921 pp, 2003.

WITTEN, Ian H.; FRANK, Eibe; HALL, Mark A. **Data mining practical machine learning tools and techniques**. 3. ed. Burlington: Morgan Kaufmann, 2011.

APÊNDICE A – ARTIGO

Algoritmo *Standard Ant Clustering Algorithm* na Tarefa de Clusterização da *Shell* Orion *Data Mining Engine*

Samuel Lodetti Ghellere[1], Merisandra Côrtes de Mattos Garcia[2]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma– SC

[2]Professora do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma– SC

samuel_ghellere@hotmail.com, mem@unesc.net

Resumo. *Tecnologias atuais de armazenamento permitem gerar grandes bases de dados. Estas podem possuir informações desconhecidas. Dessa forma, são necessárias tecnologias voltadas a explorar essas bases. Data mining é uma dessas tecnologias, utiliza algoritmos para extrair conhecimento destas bases. Este artigo demonstra a modelagem matemática e o desenvolvimento dos algoritmos SACA, Ant Based Clustering e A²CA pertencentes à tarefa de clusterização. Esta tarefa objetiva-se em agrupar uma base de dados em grupos de dados semelhantes, sendo que os algoritmos implementados nessa pesquisa simulam o comportamento de organização de cemitérios observados espécies de formiga.*

1. Introdução

O avanço da tecnologia facilita o armazenamento de dados e permite gerar grandes bases de dados. Devido ao tamanho dessas bases de dados a análise das informações tonou-se complexa para a capacidade humana. Em vista disso, técnicas capazes de procurar e extrair informações significativas dos dados foram criadas. Essa procura de relações entre os dados ficou conhecida como *Knowledge Discovery in Databases* (KDD), sendo o *data mining* a principal etapa desse processo.

Para a execução do *data mining* são necessários métodos que as implementem, esses métodos são disponibilizados em ferramentas computacionais específicas, sendo que existe em desenvolvimento a *Shell* Orion *Data Mining Engine*, que consiste em um projeto acadêmico que está sendo desenvolvido pelo Grupo de Pesquisa em Inteligência Computacional Aplicada do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense. A *Shell* Orion possui implementadas, atualmente, as tarefas de classificação, associação e clusterização.

A tarefa de clusterização tem como objetivo agrupar dados de um conjunto de elementos de forma que os grupos formados, denominados *clusters*, apresentem a maior similaridade possível com os dados do mesmo *cluster*. Algoritmos de clusterização baseados em inteligência de exame são inspirados no comportamento coletivo de insetos sociais, e possuem, como ponto positivo, flexibilidade, robustez e auto-organização.

Nesse artigo apresenta-se o desenvolvimento dos algoritmos baseados em inteligência de enxame SACA, *Ant Based Clustering* e A²CA, no módulo de clusterização da *Shell* Orion.

2. *Data Mining*

Data mining é a exploração e análise, por meio automático ou semi-automático, de grandes quantidades de dados a fim de descobrir padrões [Goldschmidt e Passos 2005].
Existem diversas tarefas de *data mining*, entre elas estão as tarefas de:

a) **associação**: baseia-se na ação de encontrar um grupo de itens afins;
b) **classificação**: busca por uma função que permita integrar corretamente cada registro de um conjunto de informações a um único rótulo de um conjunto de classes [Goldschmidt e Passos 2005];
c) **clusterização**: conhecida também por agrupamento, é utilizada para separar os registros de uma base de dados em *clusters* (subconjuntos).

3. A Tarefa de Clusterização em *Data Mining*

Esta tarefa tem como principal objetivo dividir um conjunto de dados em grupos, de forma que os elementos de cada *cluster* compartilhem propriedades semelhantes diferenciando-os dos outros subconjuntos [Goldschmidt e Passos 2005].

A clusterização frequentemente é usada como a primeira tarefa no processo de extração de padrões, podendo o seu resultado servir de entrada para alguma outra tarefa. Mediante isso, a clusterização pode facilitar e melhorar o desempenho de outros algoritmos de *data mining* no processo de descoberta de conhecimento [Berry e Linoff 2004].

3.1 Inteligência de Enxame

Denomina-se inteligência de enxame a tentativa de desenvolvimento de algoritmos inspirados no comportamento coletivo de colônia de insetos sociais e outras sociedades de animais. Esses algoritmos são caracterizados pela interação com um grande número de agentes que percebem e modificam seu ambiente localmente [Bonabeu, Theraulaz e Dorigo 1999].

Pode-se exemplificar a inteligência de enxame por meio dos processos naturais de construção de ninhos pelos cupins ou sociedade de abelhas, atividade forrageadora das formigas, divisão de trabalho e alocação de tarefas nas colônias de formiga, transporte cooperativo de comida, seleção de fontes com melhor néctar pelas abelhas, entre outros.

3.1.1 *Standard Ant Clustering Algorithm* (SACA)

O algoritmo SACA foi apresentado por Lumer e Faieta (1994). Segundo Lumer e Faieta (1994), no algoritmo SACA as formigas e os objetos (conjunto de dados) são espalhados em uma grade bidimensional. A cada iteração uma formiga é selecionada aleatoriamente e pode pegar ou deixar um objeto no seu local atual uma vez que, respectivamente: à um objeto naquele local, ou que a formiga esteja carregando um objeto e o lugar esta vazio. Supondo que o agente encontrou um objeto, a probabilidade de a formiga pegar esse objeto diminui com a similaridade dos demais objetos posicionados ao seu redor.

Além da função da vizinhança que permitiu que as formigas identificassem objetos ao seu redor, Lumer e Faieta (1994) introduziram ainda no algoritmo SACA os seguintes conceitos: memória de curta duração e população heterogênea de agentes.

3.1.2 *Ant Based Clustering*

Handl (2003) verificou a existência de dois grandes problemas no algoritmo SACA proposto por Lumer e Faieta (1994). Um destes problemas era devido ao resultado apresentado pelo algoritmo: ele não gera partições, mas sim um distribuição dos elementos em uma

representação gráfica. Outro problema identificado pela autora foi a dificuldade de se ajustar os diversos parâmetros do algoritmo para diferentes tipos de dados.

Visando aperfeiçoar o algoritmo, a autora introduziu as seguintes alterações: adaptação da função vizinhança, raio de percepção crescente, separação espacial, vizinhança ponderada, modificação na probabilidade de pegar e deixar um item e adaptação de α .

3.1.3 Adaptive Ant-Clustering Algorithm (A²CA)

Segundo Vizine et al (2005), uma das dificuldades de se aplicar o algoritmo SACA em problemas complexos é devido ao fato do algoritmo gerar um número excessivo de grupos que não correspondem ao número efetivo.

A fim de superar a dificuldade citada, Vizine et al (2005) propuseram o algoritmo adaptativo, *Adaptive Ant-Clustering Algorithm* (A²CA). Foram introduzidas no algoritmo original, SACA de Lumer e Faieta (1994), três alterações principais: uma rotina de adaptação do valor do parâmetro k_p, um campo de visão que se adapta aos grupos formado durante o processo do algoritmo, e utilização de uma heurística de feromônio.

4. O Algoritmo SACA na Tarefa de Clusterização da *Shell* Orion *Data Mining Engine*

As modelagens dos algoritmos SACA, *Ant Based Clustering* e A²CA iniciou-se com a construção dos diagramas de caso de uso, atividades e sequência utilizando os padrões UML. Posteriormente foi desenvolvida a demonstração matemática dos funcionamentos dos algoritmos com a finalidade de facilitar o entendimento e a sua implementação.

Os algoritmos necessitam dos seguintes parâmetros:

a) **número de formigas**: parâmetro que determina quantas formigas irão trabalhar para agrupar os objetos na grade, esse parâmetro é informado nos algoritmos SACA, A²CA e *Ant Based Clustering*;
b) **total de iterações**: quantidades de ciclos do algoritmo, informado nos algoritmos SACA, A²CA e *Ant Based Clustering*;
c) **memória**: quantidade de posições que a formiga consegue lembrar, informado nos algoritmos SACA, A²CA e *Ant Based Clustering*;
d) **alfa** α **:** parâmetro que determina a dissimilaridade na formação dos grupos, informado nos algoritmos SACA, A²CA;
e) **kp e kd**: constantes utilizadas no calculo das probabilidades, informadas nos algoritmo SACA e A²CA;
f) **campo de visão**: número de células que uma formiga consegue perceber ao seu redor, informado no algoritmo SACA;
g) **número de passos**: esse valor é aleatório em todos os algoritmos e representa a quantidade de células que uma formiga consegue se movimentar.

Após a parametrização os algoritmos constroem a matriz de dissimilaridade, que é montada calculando a distancia euclidiana n-dimensional. Essa matriz é quadrada e de tamanho *n* x *n*, que é definido pela quantidade de objetos a ser clusterizado. A figura 1 apresenta uma matriz de dissimilaridade [Han e Kamber 2006]:

$$\begin{bmatrix} 0 & & & & \\ dist(2,1) & 0 & & & \\ dist(3,1) & dist(3,2) & 0 & & \\ : & : & : & 0 & \\ dist(n,1) & dist(n,2) & .... & .... & 0 \end{bmatrix}$$

Figura 1. Matriz de dissimilaridade

A matriz de dissimilaridade é montada calculando a distancia euclidiana n-dimensional conforme a formula:

$$\delta(i,j) = \sqrt{\left|x_{i1} - x_{j1}\right|^2 + \left|x_{i2} + x_{j2}\right|^2 + ... + \left|x_{in} - x_{jn}\right|^2}$$

Com todas as distâncias calculadas é realizada a normalização para que os valores fiquem entre 0 e 1. Para isso usou-se a normalização MIN-MAX:

$$z_i = \frac{v_i - \min_{vi}}{\max_{vi} - \min_{vi}}(n\_\max - n\_\min) + n\_\min$$

Cada linha dessa matriz representa um objeto que será carregado pelas formigas durante o processo do algoritmo e sempre que a função vizinhança for calculada essa matriz devera ser consultada.

Com os parâmetros de entrada definidos, a primeira fase dos algoritmos é iniciada, eles espalham os dados na grade bidimensional que corresponde a uma matriz *n* x *n.* Com todos os objetos espalhados na grade, os algoritmos, aleatoriamente, carrega cada formiga com um objeto, posicionando-a no mesmo lugar do objeto que ela irá carregar. Dessa forma a segunda fase, que é a fase de agrupamento é iniciada.

Nessa fase uma formiga é escolhida aleatoriamente para iniciar o processo. Verifica-se se ela esta carregando um objeto e calcula-se a probabilidade de deixar o objeto carregado na vizinhança atual da formiga, caso for verdadeiro a formiga deixa o objeto e procura um outro livre na grade, dessa forma, calcula-se a probabilidade de pegar esse objeto. Quando uma formiga não consegue deixar o objeto na sua posição atual, ela se movimenta aleatoriamente na grade considerando o número de passos que a mesma foi parametrizada e outra formiga é selecionada para continuar o processo de agrupamento. Quando uma formiga não consegue pegar um objeto ela seleciona um outro objeto aleatoriamente até consegui pegar algum objeto para prosseguir na execução do algoritmo.

A função que verifica a vizinhança da formiga é determinada como:

$$f(i)\begin{cases} \frac{1}{\sigma^2}\sum_j \left(1 - \frac{\delta(i,j)}{\alpha}\right) & se \quad f(i) > 0 \\ 0 \quad em \quad outros \quad casos \end{cases}$$

Essa função é utilizada nos três algoritmos sendo que no *Ant Based Clustering* ela sofre modificações. Em determinado momento ela deixa de considerar o valor de σ e passa a utilizar no lugar do mesmo, o número de objetos encontrados na vizinhança e também possue uma restrição adicional que penaliza altas dissimilaridades entre os objetos. Essa função passa a ser determinada como:

$$f^*(i) = \begin{cases} \frac{1}{Nocc}\sum_j (1 - \frac{\delta(i,j)}{\alpha}) & se \quad f(i) > 0 \wedge \left(1 - \frac{\delta(i,j)}{\alpha}\right) > 0 \\ 0 \quad em \quad outros \quad casos \end{cases}$$

As probabilidades de pegar e deixar um objeto também variam entre os algoritmos. Para o SACA as equações são as seguintes, respectivamente:

$$p_d(i) = \begin{cases} 1 & se \quad f(i) \geq k_d \\ 2 * f(i) & em \quad outros \quad casos \end{cases}$$

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

No algoritmo *Ant Based Clustering*, as constantes k_p e k_d são descartadas utilizando apenas o valor da função vizinhança para calcular as duas probabilidades. As equações das probabilidades de pegar e deixar um objeto são determinadas da seguinte forma, respectivamente:

$$p_p(i) = \begin{cases} 1 & se \quad f(i) \leq 1 \\ \frac{1}{f(i)^2} & em \quad outros \quad casos \end{cases}$$

$$p_d(i) = \begin{cases} 1 & se \quad f(i) \geq 1 \\ f(i)^4 & em \quad outros \quad casos \end{cases}$$

Já o algoritmo A²CA possue uma heurística de feromônio onde é acrescentado o valor na célula onde está sendo deixado um objeto. Esse valor é considerado nas equações das probabilidades. Essas equações são determinadas das seguintes formas no A²CA:

$$p_d = \frac{1}{f(i)\phi(i)}\left(\frac{f(i)}{f(i) + k_d}\right)^2$$

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

Onde o valor de feromônio e representado por $\phi(i)$.

O parâmetro α no algoritmo *Ant Based Clustering* possue uma adaptação, esse valor é modificado sempre que a formiga falhar ao tentar deixar um objeto, essa adaptação é dada por:

$$\alpha \longleftarrow \begin{cases} \alpha + 0.01 & se \quad r_{falhas} > 0.99 \\ \alpha - 0.01 & se \quad r_{falhas} \leq 0.99 \end{cases}$$

Onde o *rfalhas* consiste no número de vezes que a formiga falhou em deixar o objeto dividido por for 100.

Já o campo de visão inicia com o valor 3 e no final das iterações essa valor passa a ser

5.

No algoritmo A²CA o campo de visão é baseado na função vizinhança, quando o valor desta é maior que um valor determinado θ, o campo de visão da formiga é acrescentado 2 unidades até um valor máximo de 7e a função vizinhança é recalculada. Já o parâmetro k_p possui um resfriamento que acontece a cada 10000 ciclos de formigas, esse resfriamento é dado por:

$$\begin{cases} k_p \leftarrow \quad k_p * 0.98, \\ k_{p\min} = 0.001. \end{cases}$$

Onde $k_{p\min}$ é o máximo de resfriamento do parâmetro k_p.

4.1 Implementação

Os algoritmos dessa pesquisa foram desenvolvidos no módulo de clusterização da *Shell Orion Data Mining*, por meio da linguagem de programação Java e do ambiente de programação integrado *NetBeans* 7.0.1.

A validação dos resultados do agrupamento gerado pelos algoritmos de clusterização foi realizada pelo uso de índices estatísticos. Foi utilizado o índice de *Dunn* proposto por J.C. Dunn em 1974 e o *C-Index* proposto por L. J. Hubert e J. R. Levin em 1976.

O índice de *Dunn* tem como objetivo avaliar as partições para identificar o quanto os *clusters* encontrados são compactados e bem separados [Jain e Dubes 1988].

$$Dunn = \min_{l<i<k}\left(\min_{i+l\le j\le k}\left(\frac{dist(c_i, c_j)}{\max \quad diam \quad (c_l)}\right)\right)$$

O índice *C-Index* apresenta valores para cada *cluster* encontrado separadamente, seus resultados estão no intervalo de [0,1], onde quanto mais próximo de zero for o resultado melhor é considerada a clusterização [Milligan e Cooper 1985].

$$C - Index = \frac{d_w(C_i) - \min(n_w)}{\max(n_w) - \min(n_w)}$$

$$d_w(C_i) = \sum_{x_i, x_{,} \in C_i} dist(x_i, x_j)$$

4.2 Resultados Obtidos

A base de dados utilizada para avaliação dos algoritmos SACA, *Ant Based Clustering* e A²CA, refere-se a dados de monitoramento da bacia do rio Urussanga, composta por 425 registros coletados ao longo de 37 pontos de monitoramentos espalhados por essa bacia.

Os atributos da base de dados de entrada selecionados foram os índices de pH e a concentração de ferro, o objetivo dos testes é analisar a qualidade dos *clusters* formados pelos algoritmos por meio dos índices de validação, a quantidade de *cluster* que cada algoritmo forma e o tempo que cada um necessita para formar os *clusters*. Todos os três algoritmos foram executados 10 vezes.

4.2.1 Resultados Obtidos pelo Algoritmo SACA

Na avaliação dos resultados gerados pelo algoritmo SACA, os parâmetros utilizados foram os seguintes:

a) **número de formigas**: 50;
b) **total de iterações** : 2500;
c) **memória:** 50;
d) **alfa**: 0.3;
e) **kp**: 0.1;
f) **kd**: 0.3;
g) **campo de visão:** 5;
h) **atributo de entrada:** pH e fe.

Na tabela 1 pode-se observa um dos resultados obtidos nos testes realizados como o algoritmo SACA.

Tabela 1. Resultado de um dos testes realizados com o algoritmo SACA.

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
1	6	21	00m: 16s: 422ms	1,3518	*Cluster* 1 = 0,0294 *Cluster* 2 = 0,1631 *Cluster* 3 = 0,1966 *Cluster* 4 = 0,0912 *Cluster* 5 = 0,0800 *Cluster* 6 = 0,1355

Conforme tabela 1 *o C-index* apresentou resultados próximos de zero, indicando que os *clusters* formados estão compactados, porém o número de *clusters* poderia ser menor de acordo com a base de dados utilizada.

A tabela 2 apresenta a variação dos resultados obtidos nos testes.

Tabela 2. Variação dos resultados do algoritmo SACA.

	Clusters	Pontos não Classificados	Índice de *Dunn*
Desvio Padrão	1,9544	3,9357	0,0465
Média	6,1	21,9	1,367
Coeficiente de Variação	32,0393%	17,9712%	3,4081%

Pode-se observar na tabela 2 que houve uma grande variação nos números de *clusters* gerados pelo SACA, comprovando que o mesmo gera resultados diferentes a cada execução.

4.2.2 Resultados Obtidos pelo Algoritmo *Ant Based Clustering*

Na avaliação dos resultados gerados pelo algoritmo *Ant Based Clustering*, os parâmetros utilizados foram os seguintes:

a) **número de formigas**: 10;
b) **total de iterações**: 100000;
c) **memória**:10;
d) **atributo de entrada:** pH e fe.

Na tabela 3 pode-se observar um dos resultados gerados pelo algoritmo nas 10 execuções:

Tabela 3. Resultado de um dos testes realizados com o algoritmo *Ant Based Clustering*.

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
1	5	12	02m: 50s: 798ms	1,3518	*Cluster* 1 = 0,0998 *Cluster* 2 = 0,1712 *Cluster* 3 = 0,2796 *Cluster* 4 = 0,1108 *Cluster* 5 = 0,0897

Nos testes pode-se observar que o algoritmo necessita de muitas iterações para começar a formar *clusters*. A utilização de um baixo *Campo de Visão* e a alteração das equações k_p e k_d do algoritmo possibilita a formação de grupos bem compactos, porém o tempo de formação dos grupos é maior comparando com o algoritmo SACA.

A adaptação do Alfa α tem efeitos positivos para o algoritmo, pois com essa adaptação possibilita que elementos de um grupo possam ser analisados novamente e movidos para outros grupos, onde a vizinhança é mais similar a ele conforme vão passando as iterações.

Na tabela 4 pode-se observar o desvio padrão, média e coeficiente de variação dos resultados obtidos pelo algoritmo.

Tabela 4. Variação dos resultados do algoritmo *Ant Based Clustering*.

	Clusters	Pontos não Classificados	Índice de *Dunn*
Desvio Padrão	1,7349	1,5132	0,1591
Média	6,3	11,1	1,4031
Coeficiente de Variação	27,5381%	13,6324%	11,3391%

Analisando a tabela 4, pode-se verificar que o algoritmo *Ant Based Clustering* tem uma variação de números de *clusters* e pontos não classificados um pouco menor que o SACA. Na tabela 3, comparando com o SACA, o tempo de execução do algoritmo aumentou, isso se justifica pelo fato do algoritmo *Ant Based Clustering* precisar de muito mais iterações que o algoritmo SACA. Já o índice de validação *C-Index* não foi melhorado no *Ant Based Clustering*, porém o objetivo desse algoritmo é diminuir a quantidade de parâmetros de entrada.

4.2.3 Resultados Obtidos pelo Algoritmo A²CA

Na avaliação dos resultados gerados pelo algoritmo A²CA, os parâmetros utilizados foram os seguintes:

a) **número de formigas**: 30;
b) **total de iterações**: 8000
c) **memória**: 20;
d) **alfa**: 0.3;
e) **kp**: 0.9;
f) **kd**: 0.3;
g) **atributo de entrada:** pH e fe.

Na tabela 5 pode-se observar um dos resultados obtidos nos testes pelo algoritmo A²CA.

Tabela 5. Resultado de um dos testes realizados com o algoritmo A²CA.

Teste	*Clusters*	Pontos não classificados	Tempo	Índice de *Dunn*	*C-Index*
1	5	20	03m: 12s: 435ms	1,3498	*Cluster* 1 = 0,0644 *Cluster* 2 = 0,2131 *Cluster* 3 = 0,0841 *Cluster* 4 = 0,0245 *Cluster* 5 = 0,0795

Esse algoritmo, apesar de necessitar dos mesmos parâmetros que o algoritmo SACA, desconsiderando o *Campo de Visão*, apresenta resultados bons em relação a o índice *C-index*, pois os valores apresentados são mais próximos de zero. Isso graças a heurística de feromônio e a sua adaptação do *Campo de Visão* para cada formiga. Esse algoritmo, como o SACA, tende a formar muitos grupos no início, porém graças as suas adaptações pequenos grupos são desfeitos e seus elementos passam a pertencer a grupos maiores e que contenham objetos similares a ele.

Na tabela 6 pode-se observar o desvio padrão, média e coeficiente de variação dos resultados obtidos pelo algoritmo A²CA.

Tabela 6. Variação dos resultados do algoritmo A²CA

	Clusters	Pontos não Classificados	Índice de *Dunn*
Desvio Padrão	0,7745	3,618	0,0008
Média	5	22,9	1,3501
Coeficiente de Variação	15,49	15,7991	0,0592

A tabela 6 demonstra que a quantidade de *clusters* gerados pelo A²CA é menor comparado com os dois algoritmos anteriores, os demais resultados também tiveram uma variação menor comprovando dessa forma o objetivo do algoritmo que é diminuir a variação dos resultados do algoritmo SACA.

5. Considerações Finais

Este artigo apresentou os algoritmos SACA, *Ant Based Clustering* e A²CA que utiliza o conceito de clusterização baseada em inteligência de enxame para a tarefa de clusterização da *Shell Orion Data Mining Engine.*

O algoritmo A²CA se diferenciou do SACA no fato de não ser tão sensível com os parâmetros informados devido a sua heurística de feromônio. O algoritmo *Ant Based Clustering*, apresentou uma demora considerável na formação dos *clusters*, devido ao fato de necessitar de mais iterações do que os algoritmos SACA e A²CA. Pode-se concluir que o algoritmo A²CA foi o que apresentou melhor resultado na qualidade interna dos *clusters* formados.

Referências

BERRY, J.A, LINOFF, Gordon S. **Mastering Data Mining**. New York: John Wiley & Sons; 2000.

BONABEU, E.; THERAULAZ, G.; DORIGO, M. **Swarm Intelligence:** from natuaral to artificial systems. New York, USA: Oxford University Press, 1999.

GOLDSCHMIDT, Ronaldo; PASSOS, Emmanuel Lopes. **Data mining:** uma guia prático. Rio de Janeiro: Elsevier, 2005.

HAN, J., KAMBER, M., **Data Mining Concepts and Techniques**. Morgan Kaufman Publishers, San Francisco, USA, 2006.

HANDL, J., **Ant-based methods for tasks of clustering and topographic mapping**: extensions, analysis and comparison with alternative techniques. Masters Thesis, Universität, Erlangen-Nürnberg, Erlangen, Germany, 2003.

JAIN, Anil K.; DUBES, Richard C. **Algorithms for clustering data**. Englewood Cliffs: Prentice Hall, 1988.

LUMER E.D., FAIETA B., **Diversity and adaptation in populations of clustering ants**. In Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, pages 501–508, 1994.

MILLIGAN, Glenn W.; COOPER, Martha C. **An examination of procedures for determining the Number of clusters in a data set**. Psychometrika, Colombus, p. 159-179. Jun. 1985.