UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO SIKORA DE MELO

UMA ABORDAGEM HÍBRIDA BIO-INSPIRADA APLICADA À MELHORA NA QUALIDADE DO RECONHECIMENTO DE PADRÕES

FORTALEZA, CEARÁ
2011

GUSTAVO SIKORA DE MELO

UMA ABORDAGEM HÍBRIDA BIO-INSPIRADA APLICADA À MELHORA NA QUALIDADE DO RECONHECIMENTO DE PADRÕES

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientação: Prof. Dr. Jackson Sávio de Vasconcelos Silva

FORTALEZA, CEARÁ
2011

GUSTAVO SIKORA DE MELO

UMA ABORDAGEM HÍBRIDA BIO-INSPIRADA APLICADA À MELHORA NA QUALIDADE DO RECONHECIMENTO DE PADRÕES

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Aprovada em __/__/ 2011

BANCA EXAMINADORA

_____

Prof. Dr. Jackson Sávio De Vasconcelos Silva (Orientador)
Universidade Estadual do Ceará

_____

Prof. Dr. Antônio Clécio Fontelles Thomaz
Universidade Estadual do Ceará

_____

Prof. Dr. Gustavo Augusto Lima de Campos
Universidade Estadual do Ceará

_____

Prof. Dr. Reinaldo Castro Souza
Pontifícia Universidade Católica do Rio de Janeiro

AGRADECIMENTOS

A Deus, por tudo. Antes, agora e sempre.

À minha noiva Hannah, pela constante compreensão, apoio, dedicação, motivação e amor.

Aos meus pais pelo apoio, por acreditarem mais uma vez nos meus estudos e, ao meu irmão, por toda a ajuda, conselhos e motivação.

A todos os meus amigos, pelo apoio, paciência e momentos de diversão.

Aos colegas e professores do mestrado, pelo apoio ao longo do curso, pelas idéias e discussões que sempre acrescentaram algo de valioso na construção do conhecimento.

Ao professor Jackson Silva, pela orientação, paciência e incentivo ao longo de todo o mestrado.

Ao professor Gustavo Campos, por toda a ajuda e inspiração desde a graduação.

Ao professor Antônio Clécio, por sempre acreditar, incentivar e motivar.

À FUNCAP pelo apoio financeiro com a manutenção da bolsa de auxílio.

“Não é porque certas coisas são difíceis que nós não ousamos.
É justamente porque não ousamos que tais coisas são difíceis!”
Sêneca

RESUMO

Esta dissertação propõe uma nova abordagem híbrida, bio-inspirada, aplicada à melhora na qualidade do reconhecimento de padrões. Busca-se assim agregar, a capacidade de clusterização das colônias de formigas, com a capacidade de classificação supervisionada das redes neurais artificiais, projetadas automaticamente com algoritmos genéticos. Assim, no intuito de aprimorar a qualidade dos *clusters* formados pela colônia de formiga, e assim resolver o problema dos dados que ficam presos ou perdidos nestes, é introduzido um agente artificial capaz de identificá-los e classificá-los com o auxilio das redes neurais. Vários experimentos são realizados para validar a proposta, mostrando que a qualidade é, de fato, aprimorada.

Palavras-chave: Reconhecimento de padrões. Colônia de formigas. Clusterização. Redes neurais artificiais. Algoritmos genéticos. Computação bio-inspirada.

ABSTRACT

This dissertation proposes a new hybrid bio-inspired approach to improve the quality of pattern recognition. In order to accomplish that goal, the clustering capacity of ant colonies is combined with the capacity of supervised classification of an evolving artificial neural network. Hence, willing to improve the quality of clusters formed by the ant colony algorithm, and by doing so, solving the problem concerning trapped or lost data among the formed clusters, an artificial agent capable of identify those lost or trapped data, and classify them with the aid of neural networks is introduced. Several experiments are to done to validates this approach, showing that the quality is indeed, improved.

Keywords: Pattern recognition. Ant colony. Clustering. Artificial neural networks. Genetic algorithms. Bio-inspired computing.

LISTA DE ABREVIATURAS E SIGLAS

A4C - Adaptive Artificial Ants Clustering Algorithm

AC - Autômato Celular

ACGAANN - Ant Colony Genetic Algorithm Artificial Neural Network

AG - Algoritmo Genético

AR - Agente Rebocador

ASM - Ant Sleeping Model

CF - Colônia de Formigas

CN - Classificador Neural

CNA - Classificador Neural Automático

GNGN - Growing Neural Gas Network

LF - Lumer e Faieta

MB - Modelo Básico

RNA - Rede Neural Artificial

SA^{4}C - A^{4}C estático

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Reconhecer padrões é uma atividade cotidiana do ser humano que muitas vezes é realizada sem a nossa percepção. Diariamente nosso cérebro analisa uma série de dados provenientes do meio que nos cerca através de nossos sensores, ou seja, nossos sentidos: visão, olfato, audição, tato e paladar. Ao escutar uma música, por exemplo, nosso cérebro processa, inicialmente, o fato de que é um som, em seguida, realiza rapidamente uma série de comparações com o que já escutamos antes e, pela melodia, reconhece que o som é uma música. Uma série de outras comparações é realizada rapidamente e somos capazes de reconhecer quais instrumentos estão sendo tocados, qual o estilo da música, se já a escutamos antes, se é semelhante à outra música e, para quem é músico, é possível ainda reconhecer padrões de acordes de um determinado instrumento, possibilitando reproduzir a música por conta própria.

Como se pode perceber, nosso cérebro é uma máquina de processamento muito potente, capaz de receber diversas informações, processá-las a uma velocidade impressionante e ainda fornecer uma análise dos dados que as compõem, de modo que podemos reconhecer o tipo e as características da informação recebida. No entanto, embora tenhamos esta incrível capacidade, existem conjuntos de dados muito grandes e com muitas informações, sobre os quais nossa máquina de processamento biológico não consegue extrair informação útil, como reconhecer algum padrão presente nestes dados. Podemos citar como exemplos, o reconhecimento de padrões em impressões digitais, em imagens, em base de dados de clientes, pacientes, imóveis ou em vários outros tipos de dados.

Desta forma, tem-se estudado diversas técnicas e algoritmos que aproveitem o grande poder de processamento matemático computacional, para desenvolver sistemas capazes de reconhecer padrões em uma massa de dados extensa e rica em detalhes, que fogem da capacidade cerebral humana de processamento.

No intuito de desenvolver tais algoritmos, o ato de reconhecer padrões pode conter as seguintes etapas: eliminar dados não representativos ou deturpados, escolher os atributos que melhor representem os dados remanescentes e então classificar, ou seja, indicar qual a categoria de cada dado de entrada. Esta divisão visa auxiliar e facilitar o processamento computacional, uma vez que, na realidade biológica, nosso cérebro realiza estas etapas

simultaneamente. Voltando ao exemplo anterior de quando escutamos uma música, inicialmente nos concentramos especificamente naquele som que desejamos ouvir, eliminando outros ruídos do ambiente. Posteriormente, concentrando-se somente na música de interesse, podemos selecionar quais instrumentos desejamos dar maior atenção, qual aspecto será levado em consideração (letra, tempo, ritmo, etc.) e baseando-se neles, podemos classificar qual o tipo de música, se já a escutamos, além de estabelecer outras possíveis classificações. Ao final, teremos reconhecido os padrões da música.

Muitas destas técnicas e algoritmos são ramos da inteligência artificial, ciência que pesquisa formas de simular em ambiente computacional a capacidade de sistemas biológicos, como o cérebro humano, de resolver problemas, utilizando, muitas vezes, o comportamento biológico como inspiração para a formulação de soluções.

Assim, vários algoritmos bio-inspirados, tais como as redes neurais artificiais que visam simular a rede de neurônios do nosso cérebro, os algoritmos genéticos, que simulam a seleção natural de indivíduos, a colônia de formigas e de abelhas, que por sua vez simulam a chamada *swarm inteligence* ou inteligência de enxames, dentre outros, têm sido propostos no intuito de fornecer uma maneira inteligente, capaz de reconhecer padrões, realizando, por exemplo, o agrupamento de dados e a classificação destes, a partir de base de dados complexas, oferecendo assim, utilidade aos dados disponíveis.

1.2 Motivação

Embora existam vários algoritmos e sistemas bio-inspirados empenhados no reconhecimento de padrões, a literatura mostra que grande parte dos avanços é realizada no sentido de melhorar o desempenho ou a qualidade com algoritmos específicos. Poucos são os casos em que se busca uma nova inspiração biológica, ou ainda, uma nova abordagem que forme um sistema integrado de soluções bio-inspiradas, capaz de utilizar os benefícios que cada uma oferece.

É neste contexto que o presente trabalho encontra motivação, uma vez que propõe agregar vários algoritmos bio-inspirados apresentados na literatura, incorporando modificações e novas abordagens artificiais e bio-inspiradas, desenvolvidas ao longo da pesquisa. No caso especifico de algoritmos que realizam clusterização, ou seja, que agrupam

dados semelhantes de maneira não supervisionada, poucas abordagens visam tratar os dados que ficam presos ou perdidos durante a formação dos *clusters*.

Deste modo, pretende-se explorar a capacidade individual de alguns destes algoritmos em determinadas etapas da solução do problema, formando um sistema robusto, inteligente e que possa tratar e classificar os *clusters* de forma automatizada, aumentando sua qualidade e, aprimorando assim, o reconhecimento de padrões.

1.3 Objetivos

A presente pesquisa busca a criação de um sistema bio-inspirado que incorpore a capacidade de agrupar dados, presente nas colônias de formigas, com a de realizar classificação e seleção de atributos das redes neurais artificiais, projetadas automaticamente com algoritmos genéticos. Deste modo, dada uma base de dados da qual se deseje extrair informações, padrões poderão ser reconhecidos no sentido de auxiliar o usuário a melhor compreendê-los e a tomar decisões a partir do conhecimento ali presente.

1.3.1 Objetivo geral

De forma geral, a pesquisa tem por objetivo apresentar uma nova abordagem de integração de sistemas bio-inspirados em colônias de formigas, redes neurais artificiais e algoritmos genéticos, para aprimorar o reconhecimento de padrões.

1.3.2 Objetivo específico

Especificamente, deseja-se com a abordagem proposta, introduzir uma nova forma de melhorar a qualidade dos *clusters* formados pelo algoritmo de colônia de formigas, realizando

para tal fim, a classificação supervisionada destes *clusters* por meio de redes neurais artificiais, projetadas automaticamente com algoritmos genéticos. Deste modo, os dados presos ou perdidos poderão ser classificados e alocados em outro *cluster* mais assemelhado, ou seja, em um grupo de dados que melhor os represente.

1.4 Estrutura da dissertação

O presente trabalho encontra-se organizado da seguinte forma: no segundo capítulo serão apresentados e fundamentados os conceitos básicos para a boa compreensão do tema proposto, expondo assim, o reconhecimento de padrões e os algoritmos bio-inspirados em colônia de formigas e em redes neurais artificiais evolutivas com algoritmos genéticos. No terceiro capítulo serão apresentadas pesquisas presentes na literatura, relacionadas ao reconhecimento de padrões utilizando colônia de formigas, à classificação através de redes neurais evolutivas com algoritmos genéticos e, por fim, à uma abordagem que mescla as anteriores. No quarto capítulo será descrita a abordagem proposta nesta dissertação, no sentido de detalhar as modificações realizadas nas anteriores, já apresentadas na literatura. No capítulo cinco serão apresentados os testes realizados com a implementação da abordagem descrita no capítulo quatro, além da avaliação dos resultados em comparação aos resultados da literatura. Em seguida será apresentada a conclusão do trabalho, bem como propostas futuras de continuação da pesquisa. Por fim, serão apresentadas as referências bibliográficas empregadas no desenvolvimento da dissertação.

2 FUNDAMENTAÇÃO TEÓRICA

Conforme exposto no capitulo introdutório, o objeto central desta dissertação é a melhora na qualidade do reconhecimento de padrões, através da junção de vários algoritmos bio-inspirados, os quais podemos citar: a colônia de formigas (CF), para efetuar o reconhecimento de padrões através da clusterização, as redes neurais artificiais (RNA), para classificar os *clusters* criados pela CF e os algoritmos genéticos (AG), para automatizar a criação das RNA. Assim, a seguir serão expostos os fundamentos sobre estes temas.

2.1 Reconhecimento de Padrões e Clusterização

Conforme introduzido no capitulo anterior, reconhecer padrões é uma atividade que, embora complexa, é realizada por nosso cérebro todos os dias sem que, muitas vezes, sequer percebamos. O reconhecimento de padrões, como destaca Jesan (2004), é um conceito simples de compreender e é realizado no dia a dia do ser humano, porém não é uma tarefa trivial traduzi-la para o ambiente virtual, especialmente de forma inteligente. No entanto, embora difícil, Jain *et al*. (1999) ressaltam sua importância e necessidade, pois é através desta que se torna possível e viável o trabalho com grandes volumes de dados, permitindo que, a partir daí, se formule hipóteses ou se tome decisões com base nas informações contidas nestes dados. Esta é, portanto, tarefa de grande relevância, principalmente no atual cenário global, onde os dados são abundantes e, muitas vezes, as informações neles contidas, são desconhecidas.

No contexto computacional, o reconhecimento de padrões pode ser organizado de acordo com o tipo de aprendizado utilizado para classificar os dados de entrada, ou seja, para definir qual é o padrão apresentado ou a que padrão os dados pertencem. Desta forma, podemos dividir o aprendizado em supervisionado e não supervisionado. O primeiro é aquele em que já existem dados previamente classificados, cujos padrões já são conhecidos e, assim, serão utilizados como um conjunto de treinamento, servindo para guiar o aprendizado, de

modo que novos dados de entrada, ainda não rotulados, possam ser identificados como pertencentes a um determinado padrão, para só então serem rotulados.

A classificação através do aprendizado não supervisionado, por sua vez, é caracterizada pela ausência de um conjunto de dados previamente rotulados, ou seja, os dados ainda não possuem nenhum padrão conhecido. É por meio deste aprendizado que se espera descobrir os padrões contidos no próprio conjunto de dados, sendo para tanto, necessário encontrar grupos com alguma propriedade ou característica semelhante. Assim, segundo Jain *et al*. (1999), pode-se definir clusterização como a classificação não supervisionada de padrões em um conjunto de dados, formando agrupamentos, também chamados de *clusters*.

É importante destacar que, ambas as formas de aprendizado são úteis e bastante utilizadas na literatura para a análise de dados e, consequentemente para o problema do reconhecimento de padrões. Esta análise dos dados pode ser classificada em exploratória, quando se deseja formular hipóteses e tomar decisões, ou confirmatória, quando se deseja apenas validar modelos já definidos dos dados. Assim, ainda segundo os autores, a clusterização é normalmente associada à análise exploratória, uma vez que, em muitos problemas envolvendo tomada de decisão, mineração de dados e o próprio reconhecimento de padrões, existe pouca ou até mesmo nenhuma informação disponível sobre os dados utilizados, para que se possa afirmar algo sobre estes, ou utilizá-los com coerência. Neste contexto, a clusterização é bastante apropriada e amplamente utilizada, tornando possível definir, ainda que de maneira preliminar, a relação existente entre os dados disponíveis e, a partir daí, tirar conclusões ou tomar decisões.

Por meio da clusterização, os dados analisados são geralmente representados como um vetor de atributos ou pontos em um espaço multidimensional, sendo então organizados em grupos (*clusters)* através de uma medida de similaridade, onde os membros, neste caso, os dados de um determinado grupo, devem ser mais semelhantes entre si do que os pertencentes a outros grupos, e desta forma cada grupo representa um padrão. A figura 1 mostra um exemplo da clusterização de dados onde em (a) são representados os dados não rotulados como pontos no espaço e em (b), o padrão associado aos dados.

Para que sejam identificados os grupos, a clusterização normalmente apresenta alguns passos, mostrados na figura 2 e descritos abaixo conforme Jain *et al.* (1999):

1. **Representação do padrão:** Este passo é referente à definição da quantidade, do tipo e de como os atributos representarão cada padrão. Assim, cada dado é composto por atributos que podem ser quantitativos ou qualitativos, e que, por sua vez, podem representar um objeto real ou abstrato, atribuindo um padrão ao dado. A título

exemplificativo, podemos citar um animal de quatro patas, dócil, pesando 10 kg e com 40 cm de altura, que poderia ser representado como (4,dócil,10,40) e classificado como um cachorro.

É importante ressaltar que esta representação depende da base de dados e do contexto em que será utilizada, sendo, portanto, responsabilidade do usuário fornecer, transformar ou ainda destacar quais são as informações mais importantes acerca dos dados. Pode-se desta forma incluir, opcionalmente, nesta etapa, a seleção e/ou a extração de atributos, visando uma melhora na representação dos dados e consequentemente nos padrões. Este passo é bastante significativo, pois mesmo uma pequena transformação na representação dos dados pode resultar em uma melhora no processo de clusterização.

Com este intuito, cada dado ***A***, pertencente a um conjunto de dados ***C***, composto de ***n*** atributos, é tipicamente definido como: $\boldsymbol{A} = \{a_1,a_2,a_3,...,a_n\}$, onde cada elemento $\boldsymbol{a_i}$ de ***A*** é um atributo que compões o dado. Assim, o conjunto de dados ***C*** pode ser definido como: $\boldsymbol{C} = \{A_1,A_2,A_3,...,A_m\}$, onde ***m*** é a quantidade de dados do conjunto, formando assim uma matriz de dados $\boldsymbol{m \times n}$.

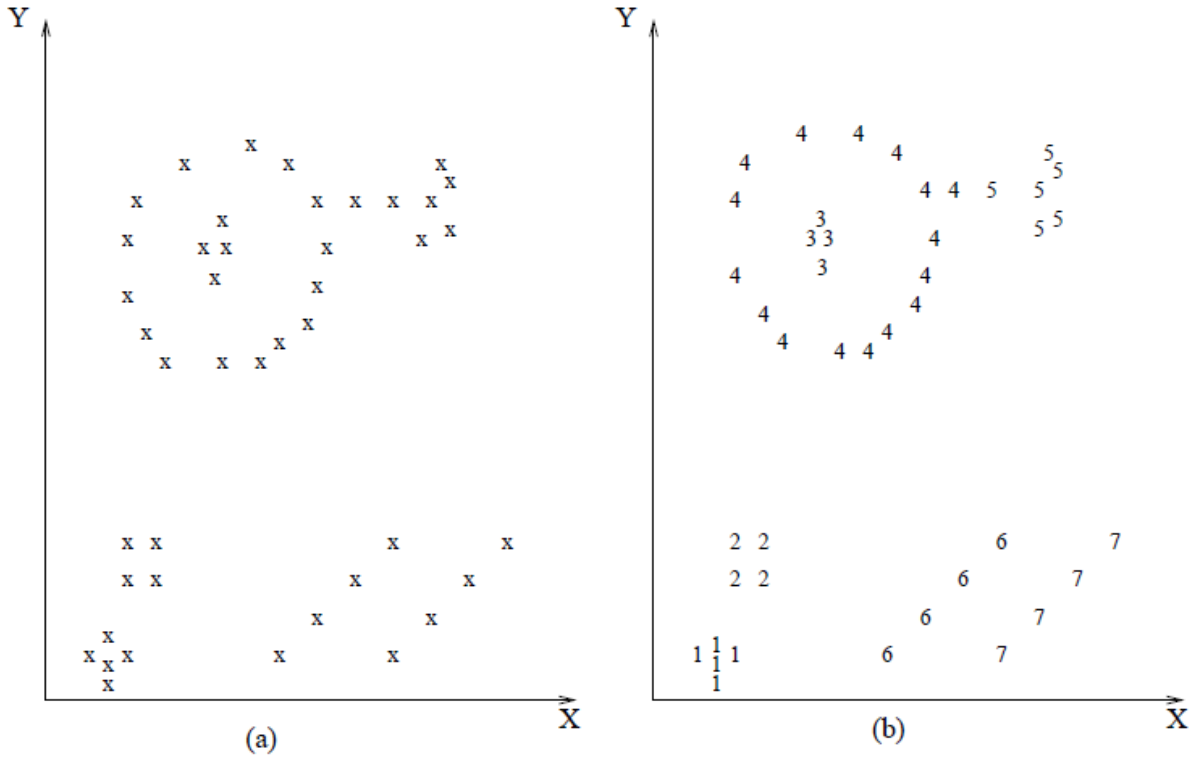


FIGURA 1 - Exemplo de clusterização de dados. Jain *et al*. (1999).

2. **Definição de uma medida de similaridade entre os dados:** A medida de similaridade é utilizada em pares de dados ou padrões através de uma função de distância, sendo fundamental para a distinção entre os dados e, consequentemente,

entre os grupos a serem formados. Existem várias funções de distância que podem ser utilizadas nesta etapa, sendo comum utilizar alguma medida de dissimilaridade entre os dados, a partir de seus atributos, como a distância de Hamming, de Minkowski e a mais comumente utilizada, distância Euclidiana que pode ser definida como:

$$d(A_i, A_j) = \left\| A_i - A_j \right\|_2 = \sqrt{\sum_{k=1}^{n} (a_{i,k} - a_{j,k})^2} \qquad 1$$

3. **Clusterização ou Agrupamento:** Este é o passo fundamental em que de fato os grupos, ou padrões, são definidos, dando inclusive o nome a este tipo de aprendizado. Em geral, o processo de clusterização é subdividido em hierárquico e não hierárquico ou particional. O primeiro refere-se a quando os padrões são aninhados e comparados entre si para definir se serão agrupados ou divididos, de acordo com uma medida de similaridade. Já o segundo, refere-se a quando não se deseja aninhar padrões, mas sim criar grupos independentes, otimizando algum critério local, inerente ao grupo formado, ou global. Ainda dentro destas subdivisões, existem várias técnicas que podem compor um algoritmo de clusterização de acordo com algumas particularidades.

   Uma dessas particularidades, diz respeito a como associar os padrões aos dados. Esta associação pode ser **rígida** (*hard*), quando cada dado recebe de maneira bem definida um padrão inicial, podendo ao longo do processo mudar de grupo, mas ao final terá também somente um padrão associado. Neste caso, é utilizado o conceito de rotular uma classe ao dado, de forma que cada dado $\boldsymbol{A_i}$ recebe um rótulo $\boldsymbol{r_i}$. Assim, dado o conjunto de dados $\boldsymbol{C}$, o conjunto de rótulos $\boldsymbol{R}$ é definido como: $R = \{r_1, r_2, r_3, ..., r_k\}$ onde k é o número de grupos formados (*clusters*). Por outro lado, pode-se utilizar uma associação nebulosa (*fuzzy*), quando cada dado recebe uma parcela de participação em cada grupo formado.

   Outra particularidade diz respeito à técnica **aglomerativa**, quando se deseja que, inicialmente cada dado seja considerado um grupo, de forma que os grupos vão sendo formados por aglomeração dos dados, diferentemente da técnica **divisiva**, quando inicialmente todos os dados formam um único padrão a ser separado até que seja atingido um critério de parada.

   Além destes, podem ser considerados vários outros aspectos, tais como a quantidade de atributos que serão utilizados para descrever os dados (mono-atributo ou poli-

atributo), se o tipo de particionamento deve ser determinístico ou estocástico ou ainda se o conjunto de dados deve ser incremental ou não. Desta forma, existem vários algoritmos e várias abordagens que visam realizar esta etapa, o que permite uma grande flexibilidade em sua implementação.

4. **Apresentação do resultado:** Esta etapa refere-se à maneira como o resultado da clusterização será representado. Este poderá ser exibido de forma intuitiva e utilizável por uma pessoa, de forma que seja fácil compreender a relação entres os grupos formados, como através de gráficos, ou ser tratado de forma automática por uma máquina, a fim de efetuar outros processamentos com o resultado. Ao final deste passo, deseja-se obter uma abstração dos dados para que efetivamente se tenha uma informação útil, que possa ser facilmente interpretada e utilizada, ao invés de ter apenas uma nova organização dos dados em forma de padrões.
5. **Validação:** A última etapa envolve alguma forma de avaliar o resultado, indicando se os agrupamentos formados são coerentes e de boa qualidade. Este passo, de modo geral, recorre à verificação de algum critério ótimo a ser satisfeito pelo algoritmo, o que, no entanto, geralmente é bastante subjetivo. Processos de validação mais específicos podem ser estudados em Jain *et al.* (1999), em Dubes (1993) e em Jain e Dubes (1988).

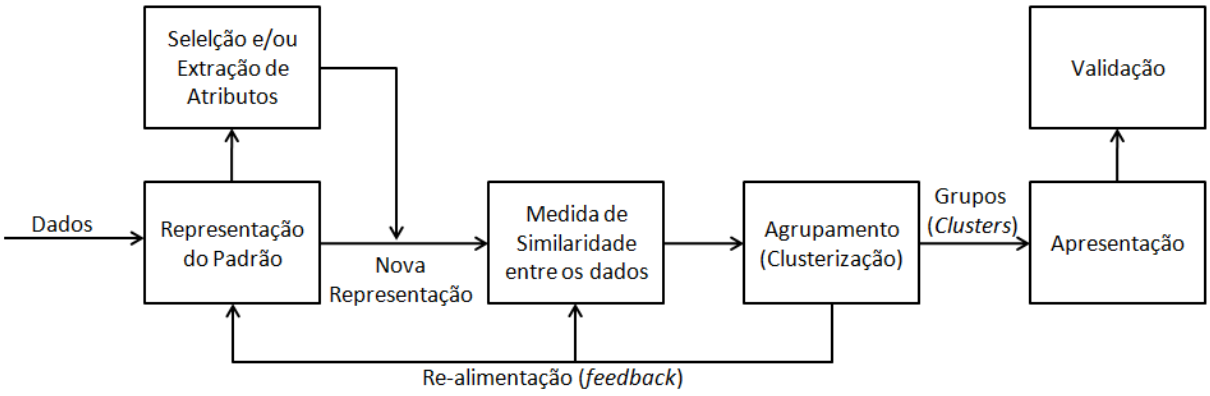


FIGURA 2 - Etapas do processo de clusterização.

Ao se utilizar estes passos e combinar as várias técnicas existentes para a etapa de agrupamento, surgem vários algoritmos que visam realizar a clusterização para abordar os mais diversos problemas. Isto se deve também ao fato de que, as técnicas de clusterização dependem muito dos dados e, por isso, não são universais. Em Jain *et al*. (1999) pode-se encontrar uma grande variedade de algoritmos que realizam a clusterização, bem como sua

utilização em diversos problemas reais. A figura 3 mostra uma ramificação de diversas abordagens para realizar a clusterização.

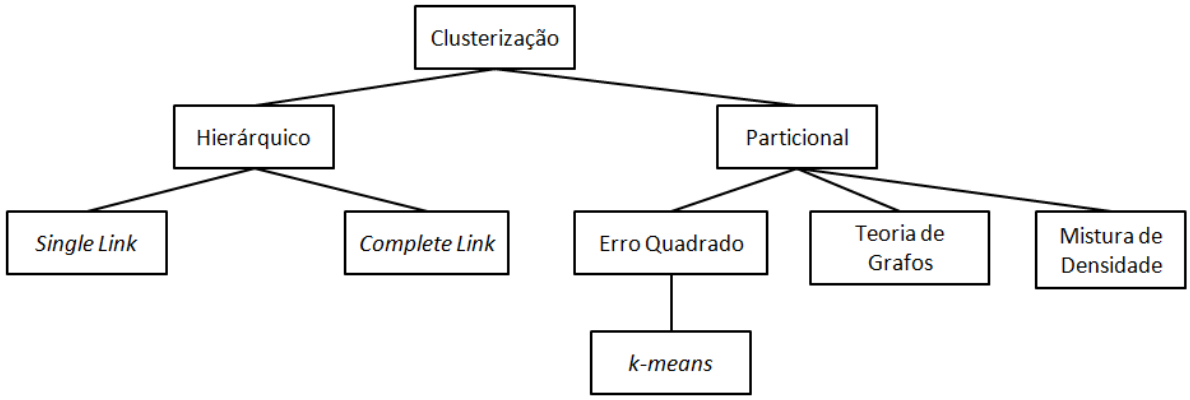


FIGURA 3 - Abordagens para clusterização. Adaptado de Jain *et al*. (1999).

Em decorrência das características do problema de clusterização, tais como sua complexidade, sua dependência em relação aos dados e da grande quantidade destes, muitas abordagens inteligentes do ponto de vista da inteligência artificial vem sendo propostas a fim de solucionar este problema, sendo muitas delas inspiradas biologicamente.

2.2 Algoritmos Bio-Inspirados

Embora a tecnologia avance a cada dia, os sistemas naturais a nossa volta possuem tamanha variedade e complexidade que, mesmo estas tecnologias não são capazes de substituir ou sequer imitar grande parte destes sistemas, como destaca Bongard (2009). No âmbito computacional esta relação também está presente, de forma que muitos problemas demandam algoritmos computacionais e equipamentos cada vez mais robustos para fornecer soluções adequadas, muito embora sistemas naturais complexos possuam tal capacidade, mesmo quando compostos por elementos mais simples.

Foi devido a estas observações do mundo natural, em especial às que dizem respeito a indivíduos, sociedades e mecanismos vivos, que surgiu a computação bio-inspirada e, com ela, os algoritmos que visam *imitar* o comportamento biológico para a solução de diversos problemas complexos. Assim, dentro da inteligência artificial (IA), muitos algoritmos buscam inspiração na biologia, dentre eles as redes neurais artificiais (RNA), que tiveram inspiração

no cérebro humano, os algoritmos genéticos (AG), os quais tiveram como inspiração a genética, além dos algoritmos baseados em insetos sociais (formigas, besouros, abelhas, etc.).

Para o problema de clusterização citado anteriormente, Jain *et al*. (1999) destacam que os que se baseiam em RNA e AG possuem fortes atrativos como a grande capacidade de lidar com processamento paralelo, com grande quantidade de dados e ainda com uma certa independência do conjunto de dados, podendo embutir uma seleção de atributos.

Os algoritmos baseados em insetos sociais são também bastante estudados e utilizados para este problema, pois apresentam alta inteligência de enxame, isto é, embora individualmente não sejam capazes de resolver tarefas complexas, quando atuam em conjunto, como numa colônia ou enxame, passam a ter esta capacidade, como bem destacam Bonabeau *et al.* (1999). Esta capacidade de atingir objetivos complexos, como o agrupamento de dados, através de entidades simples é, segundo os autores supracitados, um componente fundamental para caracterizar a coletividade dos insetos sociais como uma forma de sistema auto-organizável.

Um exemplo que merece destaque são as colônias de formigas, que trabalham juntas para limpar, nutrir e construir imensos formigueiros, como se fossem um único organismo vivo, capaz de se auto-organizar mesmo sem a existência de uma inteligência ou controle central.

Outro exemplo importante são as redes neurais artificiais, onde a unidade mais simples, o neurônio, não apresenta um grande poder de processamento quando isolado, porém quando conectado a outros neurônios, forma uma rede capaz de se auto-adaptar e aprender. Esta característica é muito importante e desejada para o problema de classificação, seja ela supervisionada ou não supervisionada.

2.3 Colônia de Formigas

Ao observar o comportamento de formigas reais, muitos pesquisadores foram inspirados por várias de suas características, tais como a busca por comida, a construção de formigueiros, bem como a limpeza, manutenção, ventilação e defesa destes. Observaram assim que estas tarefas são realizadas por indivíduos pequenos, não dotados de inteligência e que, por si só, não seriam capazes de realizar nenhuma destas ações. Deste modo, o comportamento de

trabalho cooperativo e grupal, chamou a atenção de estudiosos que decidiram investigar que mecanismos levavam as formigas a agirem com tamanha coordenação.

Devido à grande quantidade de indivíduos e coordenação descentralizada de tarefas, as colônias de formigas (CF) possuem elevado paralelismo, auto-organização e tolerância a falhas, que segundo Aranha e Iba (2006), representam características fundamentais para muitos sistemas computacionais modernos, como, por exemplo, para o problema de clusterização.

Assim, observando como seria possível esta auto-organização, muitas formas de comunicação entre as formigas foram sendo estudadas, como apresentam Bonabeau *et al*. (1999), entre elas a comunicação direta, realizada através do contato antena-antena, por meio da troca de fluídos, pelo contato mandibular, por contato químico na forma de odor entre as integrantes do mesmo formigueiro, dentro outros. Observou-se também a existência de uma forma indireta, sem contato físico, que, por sua vez, ocorre quando um indivíduo modifica o ambiente e os demais respondem a esta modificação. Este tipo de contato indireto foi observado e denominado estigmergia pelo pesquisador francês Grassé (1959).

Com base neste conceito, muitos comportamentos complexos apresentados pelas formigas e que eram, aparentemente, sem nexo, puderam ser traduzidos na forma de algoritmos. O comportamento de busca por comida, por exemplo, como citam Handl (2003) e Bonabeau *et al.* (1999), pôde ser visto como uma tarefa em que as formigas envolvidas depositam substâncias químicas, chamadas de feromônio, o que oferece um reforço positivo para que outras sigam o mesmo caminho. Tal comportamento serviu de base para a criação de algoritmos que tinham por objetivo resolver problemas de otimização, levando à criação de uma metaheurística baseada em CF chamada *Ant-Colony optimization*.

Outro comportamento observado em formigas reais, como presente nas espécies *Pheidole pallidula* e *Messor sancta*, é a capacidade de empilhamento de cadáveres, conforme mostrado na figura 4, em que existem 1500 corpos espalhados em uma pequena área e os resultados ao longo de três, seis e trinta e seis horas. Esta habilidade demonstra outro comportamento presente, também, na manutenção de um formigueiro, em que em um ambiente, as formigas tendem a carregar objetos, empilhando-os em um outro local de acordo com as características de outros objetos presentes neste local de depósito. Este comportamento foi visto como a capacidade que formigas reais possuem para agrupar objetos e serviu como inspiração para a criação de algoritmos que envolvem a clusterização de dados.

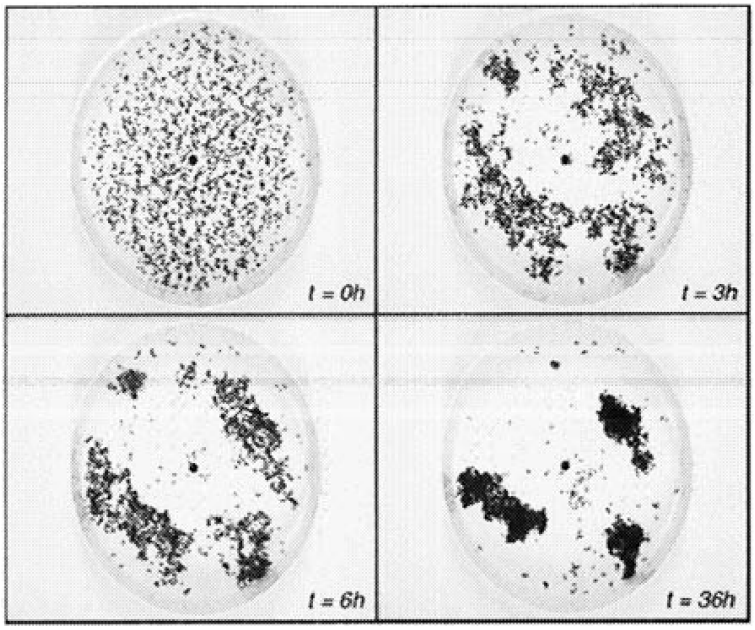


FIGURA 4 - Tarefa de clusterização desempenhada por formigas reais da espécie *Messor sancta*. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999

2.3.1 Modelo Básico e Modelo de Lumer e Faieta

Baseando-se neste último comportamento, Deneubourg *et al.* (1990) introduziram o primeiro algoritmo baseado em colônia de formigas para a tarefa de clusterização, denominado posteriormente de modelo básico (MB), aplicando-o para que simples robôs desempenhassem a tarefa de pegar um objeto e transportá-lo para um local adequado.

Desta forma, a idéia principal do modelo básico é a de que em uma grade sejam espalhados dados e dispostos agentes os quais se movimentem apenas aleatoriamente, de modo que, quando um agente encontra um objeto, que é um dado, ele deve pegá-lo e movê-lo para outro lugar na grade. Este movimento deve ser feito de tal forma que o agente somente deve depositar o objeto que está levando quando encontrar em sua vizinhança, vários objetos semelhantes àquele que carrega.

Os autores definiram então medidas que controlam a probabilidade com a qual os agentes irão pegar, $\boldsymbol{p_p}$, e depositar, $\boldsymbol{p_d}$, os dados:

$$p_p = \left(\frac{k^+}{k^+ + f} \right)^2 \qquad 2$$

$$p_d = \left(\frac{f}{k^- + f} \right)^2 \quad (3)$$

Nestas equações, *f* representa a fração de dados que estão na vizinhança do agente e que são de um mesmo tipo, e as constantes k^+ e k^- determinam a influência de *f*. Neste modelo, a função *f* é determinada através de uma memória de curto prazo de cada agente, o qual guarda o conteúdo das últimas células que visitou. Assim, percebe-se que, quando *f = 0*, ou seja, quando não há objetos semelhantes na vizinhança, a probabilidade de o agente pegar um objeto é de 100%, enquanto a probabilidade de depositar um dado nesta vizinhança é 0%. Quando f tende a 1, significa que a vizinhança do agente está com muitos objetos semelhantes e, neste caso, a probabilidade de pegar um objeto é muito pequena, enquanto a de depositar, é bem grande.

Inspirados no modelo básico, Lumer e Faieta (1994) mudaram o foco dos robôs para a análise de dados e, assim, realizaram uma série de modificações para que o algoritmo pudesse trabalhar com dados numéricos, promovendo também uma melhora na qualidade dos agrupamentos formados. Este novo modelo ficou conhecido como LF, ou modelo de Lumer e Faieta.

Neste modelo, tanto a grade quanto a disposição de dados e agentes é a mesma do MB, mas agora os dados são representados como vetores numéricos e, para determinar a semelhança entre estes, os autores definiram uma medida de dissimilaridade entre pares de dados. Com este intuito, utilizaram a distância Euclidiana para medir a distância, ou dissimilaridade, entre os objetos, mas permitem também que outras distâncias possam ser utilizadas.

Com esta nova medida, dado um objeto o_i e outro o_j, define-se a dissimilaridade entre eles como sendo $d(o_i,o_j)$ e a função $f(o_i)$ agora é utilizada para calcular a média das semelhanças entre um objeto o_i e todos os seus vizinhos. A partir destas novas funções, a probabilidade de um agente pegar um objeto, p_p, continua igual a do MB, mas a de depositar um objeto, p_d, bem como a função *f* são agora definidas:

$$p_d = 2f(o_i) \textbf{ se } f(o_i) \leq k^- \textbf{ ou } 1 \textbf{ se } f(o_i) > k^- \quad (4)$$

$$f(o_i) = \max\left\{0, \frac{1}{r^2} \sum_{o_j \in N_{rxr}(o_i)} \left[1 - \frac{d(o_i,o_j)}{\alpha}\right]\right\} \quad (5)$$

Nesta última equação, r^2 representa o tamanho da vizinhança local de cada agente e α um fator que define a escala de dissimilaridade entre os pares de objetos.

Neste modelo, os agentes possuem ainda uma memória de curto prazo em que guardam a informação do local onde depositaram o objeto anterior, sendo direcionados para este local quando passam a carregar um novo objeto.

Em ambos os modelos, Cheng *et al.* (2004) e Xu *et al.* (2007), destacam que por separarem as formigas, que são os agentes, dos objetos, que são os dados, o custo de armazenamento e de processamento de ambos os modelos se torna muito elevado. Outro ponto apontado pelos autores é que devido a natureza de movimentação naqueles modelos, as formigas realizam uma grande quantidade de movimentos improdutivos antes de efetivamente recolher e depositar um objeto, o que eleva muito o tempo computacional gasto pelo algoritmo, tornando-se ainda mais lento se exigida a formação de *clusters* com maior qualidade.

2.3.2 *Ant Sleeping Model*

Buscando uma nova forma de tratar os problemas dos modelos anteriores, Cheng *et al.* (2004) propuseram um novo modelo, que logo foi mais bem detalhado por Xu *et al.* (2007), em que buscaram inspiração na representação dos agentes por meio de autômatos celulares (AC) de John von Neumann e no comportamento descrito pela teoria das necessidades hierárquicas do psicólogo Abraham Harold Maslow, como mostra a figura 5.

A utilização de AC para representar as formigas, se deve à grande capacidade desta representação para simular fenômenos complexos através de poucas regras e operações simples, da mesma forma como se comportam os indivíduos em um sistema auto-organizável. A outra inspiração se deve ao fato de que na teoria do referido psicólogo, os seres humanos após realizarem seus anseios fisiológicos primários, buscam por segurança, e quando se aplica esta teoria para outros seres vivos mais frágeis, como as formigas, estas de fato buscam locais mais seguros, seja se referindo ao local de construção de formigueiros, seja na divisão das câmaras dentro destes, ou até mesmo pelo fato de se agruparem com outras formigas mais semelhantes.

FIGURA 5 - Pirâmide da teoria das necessidades hierárquicas de Abraham Harold Maslow.

Com base nestes fatores os autores criaram um novo modelo chamado *Ant Sleeping Model* (ASM), que visa explicar o comportamento de busca por um habitat seguro, onde as formigas possam dormir. A seguir, o modelo é definido formalmente:

1. **Autômato Celular:** O autômato celular é quem representará as formigas e o espaço em que vivem, bem como definirá as regras de transição, ou seja, as regras de clusterização. Assim, o autômato A é composto por uma 4-tupla: A = (S_d, Q, N, R) onde S é o espaço celular e d a dimensão deste espaço; Q é o conjunto de estados das células deste espaço; N a vizinhança celular; R o conjunto de regras de transição de estados.
2. **Conjunto de estados celulares:** Quando existe um agente em um célula, seu rótulo é denominado i e neste caso, $i \neq 0$, sendo a classe ao qual pertence denominada c_i. Quando a célula está vazia, isto é, não há nenhum agente, $i = 0$ e $c_i = 0$. Se n é a quantidade de agentes, $Q = \{ (i,c_i) \mid 0 \leq i \leq n, 0 \leq c_i \leq n \}$.

3. **O espaço Celular é a grade G:** A grade G é representada como uma matriz bi-dimensional, onde “vivem” os agentes. Neste modelo, a grade é esférica, na forma de um tórus, de forma que as bordas da matriz são conectadas, permitindo assim,

uma movimentação livre dos agentes. Assim, a posição (*x*,*y*) na grade G, é representada como G(*x*,*y*) ϵ Q, ou seja, se um *agente* $_i$ está na posição (*x*,*y*), então G(*x*,*y*) = (*i*,c_i), caso não haja um agente na posição (*x*,*y*), então G(*x*,*y*) = (0,0). O tamanho da grade é definido como G = [0...w(n)-1]x[0...h(n)-1] onde n é o número de agentes e:

$$w(n) = h(n) = 2\left(\lfloor \sqrt{n} \rfloor + 1\right) \quad (6)$$

4. **A célula é o agente:** No ASM, cada agente representa um objeto, de modo que cada formiga é o próprio dado a ser clusterizado. Assim, a quantidade de agentes é fixa e igual à quantidade de dados, variando apenas a classe a qual o dado/agente pertence. Esta representação acaba sendo mais natural ao problema de clusterização, pois o próprio dado é agrupado ao invés de ser classificado, assim como as formigas reais se agrupam por semelhança e em busca de segurança. Desta forma, cada agente contém todos os atributos do dado que ele representa, além de incluir informações acerca de sua vizinhança, além de outras próprias, como seu estado, sua classe atual, dentre outras. Formalmente, um *agente* $_i$ é representado por sua posição (x_i,y_i), com a nomenclatura G(*agente* $_i$) = G(x_i,y_i) = (*i*, c_i).

5. **A vizinhança celular é a vizinhança do agente:** Para cada agente do ASM, existem dois conjuntos de vizinhos: N(*agente* $_i$) e L(*agente* $_i$). O conjunto N representa toda a vizinhança do *agente* $_i$, inclusive as células da grade que estiverem vazias. Já o conjunto L representa apenas as posições vazias da vizinhança do *agente* $_i$. Para determinar o tamanho da vizinhança dos agentes, pode-se utilizar duas abordagens: a primeira utiliza a visão do agente apenas nos sentidos norte, sul, leste e oeste, sendo denominada modelo de 4-Vizinhos de John von Neumann; a segunda utiliza, além das quatro anteriores, as diagonais nordeste, sudeste, noroeste e sudoeste, sendo denominada 8-Vizinhos de Moore. Além desta vizinhança padrão, pode-se utilizar uma vizinhança expandida, de forma que sejam delimitados novos valores para o limite de visão vertical, s_y, e horizontal, s_x, de cada agente. Com base nestes valores, o tamanho da vizinhança é dado por:

$$\text{Tam\_Vizinhança} = (2s_x + 1) \times (2s_y + 1) \quad (7)$$

6. **A medida de similaridade:** Para determinar a proximidade, ou seja, a semelhança entre os dados, deve-se utilizar uma medida de distância, que pode ser a Euclidiana da equação 1, para que se possa determinar a distância de um agente para outro. Assim, d_{ij} = *d*(*agente* $_i$, *agente* $_j$) = *d*(*data* $_i$, *data* $_j$) e com base nestes valores, cria-se

uma matriz de dissimilaridade com todas as distâncias inter-agentes.

7. **A medida de aptidão *f*:** Esta medida visa mensurar o quão seguro o agente se sente na sua posição atual. Assim, deve ser calculada como segue:

$$f(agente_i) = \max\left\{0, \frac{1}{(2s_x+1)\times(2s_y+1)} \sum_{agente_j \in N(agente_i)} \left(1 - \frac{d(agente_i, agente_j)}{\alpha_i}\right)\right\} \quad 8$$

$$\alpha_i = \frac{1}{n-1}\sum_{j=1}^{n} d(agente_i, agente_j) \quad 9$$

$$\alpha = \frac{1}{n\times(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n} d(agente_i, agente_j) \quad 10$$

Nestas fórmulas, α_i refere-se à média das distâncias entre o *agente* $_i$ e todos os outros agentes, e α a média geral das distâncias entre todos os agentes, ambos representam um indicativo de quando um agente deve se distanciar ou se aproximar.

8. **A probabilidade de ativação:** No ASM o agente possui apenas dois estados, ativo ou dormindo. Para determinar se um agente está ativo e apto a se movimentar, é preciso definir uma probabilidade de ativação deste agente, de modo que se o valor for próximo a 1, o agente tem muitas chances de ser ativado, ao passo que, se o valor for próximo a 0, é esperado que ele fique dormindo com mais frequência. Assim, a probabilidade de ativação p_a é dada por:

$$p_a(agente_i) = \frac{\beta^{\lambda}}{\beta^{\lambda} + f(agente_i)^{\lambda}} \quad 11$$

Onde, β representa o limiar de ativação do agente e λ a pressão sofrida pelo agente devido aos demais agentes. Desta forma, se f é muito maior que β, p_a tende a 0 e assim, o agente tende a ficar em dormindo. Caso contrário, se f é muito menor do que β, p_a tende a 1, e assim o agente tem maior probabilidade de ser ativado.

9. **As regras de clusterização R:** A regra de clusterização desempenha o papel de atualizar a classe dos agentes, de modo que:
   a. Se o *agente* $_i$ estiver dormindo, sua classe é igual a da maioria de seus vizinhos.
   b. Se o *agente* $_i$ estiver ativo, sua classe é igual a sua classe de origem.
10. **O modelo *Ant Sleeping Model:*** Por fim, o ASM pode ser resumido como uma 5-tupla da seguinte forma: ASM = (G, Q, N, R, D) onde G é a grade, Q o conjunto de estados das células, N a vizinhança dos agentes, R o conjunto de regras de transição e D_{nxn} a matriz de dissimilaridade entre os agentes.

Uma vez definido o modelo, pode-se descrever o passo a passo de um algoritmo nele baseado:

1. Inicializar a grade e posicionar os agentes aleatoriamente sobre esta, de modo que cada posição contenha apenas um agente. Neste estágio inicial, todos os agentes estão ativos e aptos a se mover segundo alguma estratégia de movimento, que pode ser, por exemplo, aleatória (escolhe-se uma posição livre na vizinhança e muda sua posição para esta).
2. Quando um agente se move, deve-se recalcular sua aptidão f de acordo com sua nova vizinhança, e obter sua nova probabilidade de ativação p_a para decidir se este deve continuar ou não se movendo. Se p_a for pequena, o agente entra no estado de repouso e dorme, caso contrário permanece ativado e se movendo. Desta forma, o agente sempre busca, de acordo com sua aptidão, um local seguro para dormir.

A execução destes passos leva ao longo do tempo, a uma redução dos movimentos até que os agentes mais semelhantes estejam seguros, obtendo-se assim, a clusterização.

2.4 Redes Neurais Artificiais e Algoritmos Genéticos

As redes neurais artificiais, ou RNAs, foram desenvolvidas por McCulloch e Pitts, na década de 40, como um modelo computacional do cérebro humano que visa simular as unidades de aprendizado, os neurônios, e as conexões cerebrais representadas pelas sinapses. No modelo computacional foi então criado o neurônio artificial e para as sinapses, foram introduzidos pesos numéricos entre as conexões dos neurônios que podem ser modificados através do aprendizado. Na figura 6 é mostrado o neurônio biológico e seu equivalente artificial na figura 7.

Ao serem combinados diversos neurônios artificiais, forma-se uma RNA. Ela apresenta uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída. Pela primeira camada chegam as informações de entrada, os estímulos para a rede. Na camada intermediária, também chamada de oculta, encontram-se vários neurônios, podendo inclusive existir mais de uma camada, que recebem “sinais” vindos da entrada, após estas serem multiplicadas pelos pesos e somadas. Por fim tem-se a camada de saída que também pode apresentar vários neurônios, ou seja, várias saídas.

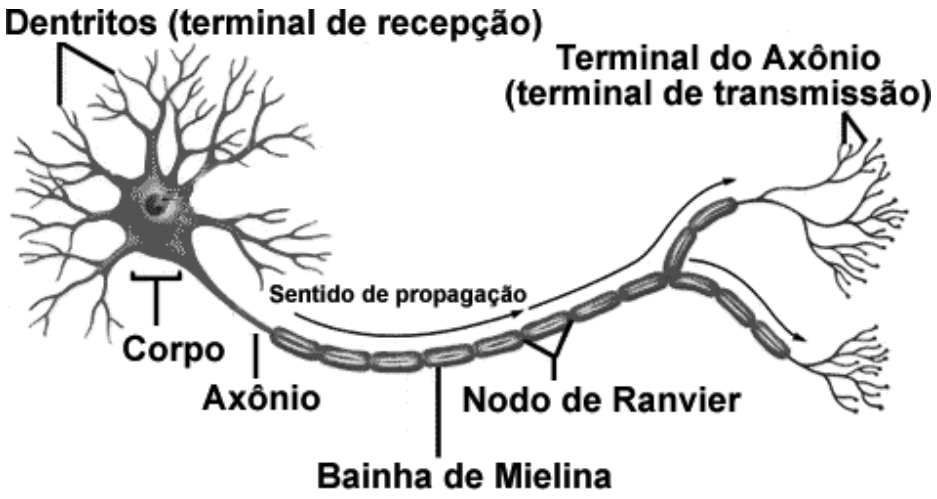


FIGURA 6- Neurônio biológico. Adaptado do livro Conceitos de Biologia vol.1. Amabis e Martho (2001)

2.4.1 Características Gerais

Em uma RNA, cada neurônio realiza um processamento simples de ativação: primeiramente o valor da entrada é multiplicado pelo peso de sua conexão com outro neurônio da camada intermediaria ao qual está conectado e somado com o valor das demais multiplicações para este mesmo neurônio de destino, como mostrado na figura 7.

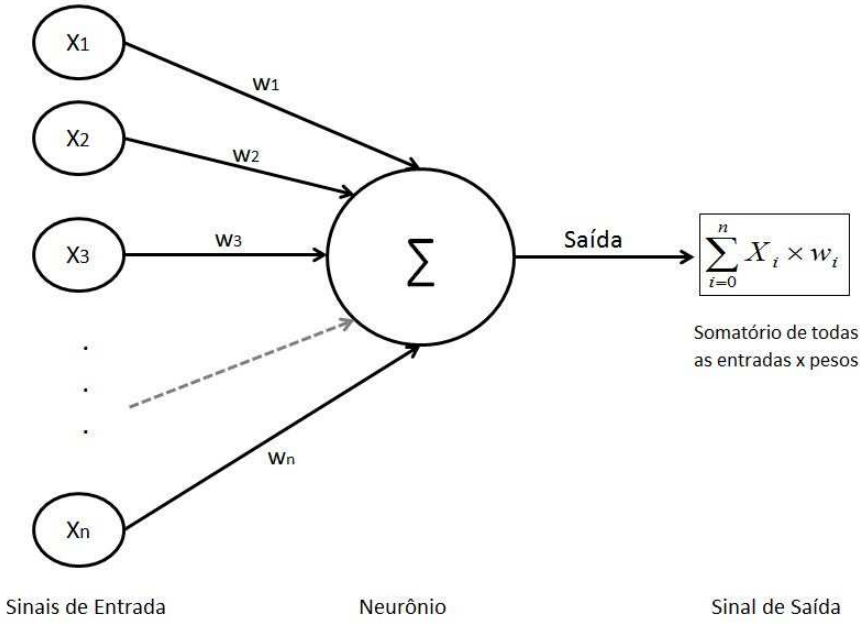


FIGURA 7 - O neurônio artificial.

No exemplo da figura 8, a RNA é composta de dois neurônios na camada de entrada (x_1 e x_2), duas camadas intermediárias, sendo a primeira com três neurônios e a segunda com dois, e na camada de saída apenas um neurônio. A figura 8 mostra ainda parte do processamento da rede, onde o valor da entrada x_1 é multiplicado pelo peso $W(x_1)_2$ e o valor da entrada x_2 multiplicado pelo peso $W(x_2)_2$ e o neurônio 2 da camada intermediaria recebe como valor de entrada o resultado da soma de cada uma dessas multiplicações: $x_1.w_{(X1)2} + x_2.w_{(X2)2}$.

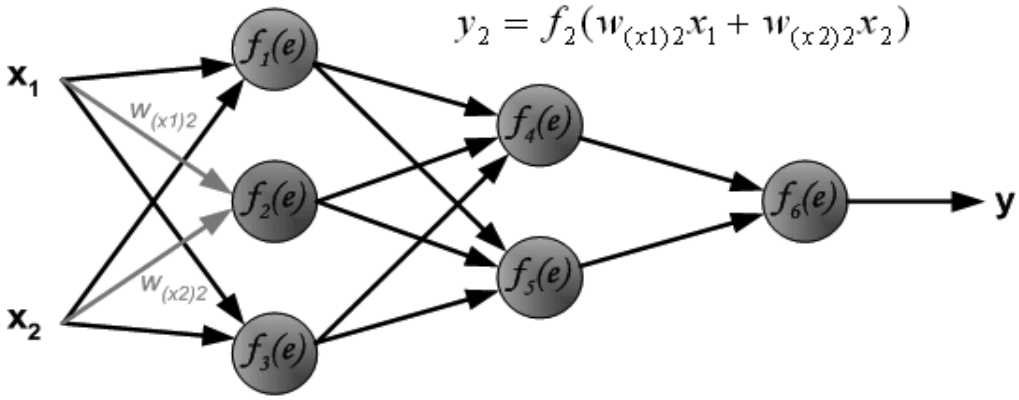


FIGURA 8 - Exemplo de Rede Neural Artificial. Adaptado do livro *Sieci neuronowe*. Tadeusiewcz (1992).

Em seguida, aplica-se uma função de ativação que deve simular a não linearidade do modelo biológico, sendo comumente empregada a função sigmóide mostrada na equação 12 e que permite a transição gradual entre dois estados:

$$f(x) = \frac{1}{1+e^{-x}} \qquad 12$$

As RNA são basicamente classificadas em redes *feed-forward*, ou alimentadas à frente, e redes recorrentes, mas podem apresentar várias arquiteturas, ou seja, várias formas de dispor suas camadas, as quantidades de neurônios em cada uma delas e a disposição das conexões. Nas redes *feed-forward* o sinal é propagado da entrada para a saída, enquanto nas recorrentes a saída de um neurônio pode ser entrada de outro da camada anterior, apresentando assim, ciclos.

Outra característica fundamental das RNA é o tipo de aprendizado adotado pela rede. Este pode ser supervisionado ou não supervisionado. O primeiro realiza o treinamento com o auxílio de um professor que utiliza um conjunto de treinamento contendo entradas com saídas conhecidas, o qual é apresentado à rede de forma que a partir da entrada dada, a rede realiza seu processamento para obter a saída desejada.

O aprendizado não supervisionado é caracterizado pela ausência de uma saída conhecida e pela ausência de um professor. Assim, o padrão de entrada permite que a rede escolha o padrão de saída a partir das regras de aprendizado, e não por uma saída já conhecida. Os algoritmos de aprendizado não supervisionado mais utilizados são o algoritmo de Hopfield e os mapas de Kohonen.

2.4.2 O *Backpropagation*

Dentre os algoritmos de aprendizagem supervisionada, o *backpropagation*, criado por Werbos (1974) e popularizado por Rumelhart *et al.* (1986), é o mais utilizado, sendo também o algoritmo de aprendizagem da RNA utilizada neste trabalho.

Este algoritmo possui dois passos computacionais: processar o sinal da entrada para a saída e processar a correção do erro de trás pra frente, daí o nome *backpropagation*, isto é, a propagação de trás para frente, da saída para a entrada.

No primeiro passo uma entrada é fornecida à rede e seu efeito é propagado sem modificação dos pesos, até a saída. No segundo passo o erro calculado na saída é propagado até a entrada, camada por camada, para que os pesos sejam ajustados conforme uma regra de correção.

Em geral os passos de treinamento com o *backpropagation* são:

1. Apresentar as entradas do conjunto de treinamento.

2. Fazer a entrada percorrer toda a rede da camada de entrada, passando pelas intermediárias até a saída, realizando as operações de soma, multiplicação e ativação a cada transição de camada.

3. Calcular o erro na saída e verificar se é um erro aceitável. Caso não seja, o erro é retro-propagado ajustando os pesos das conexões de trás para frente, visando obter na próxima saída um erro mínimo ou aceitável.

2.4.3 Algoritmos Genéticos e Redes Neurais Evolutivas

Os algoritmos genéticos, apresentados primeiramente por Holland (1975), têm como objetivo desenvolver sistemas artificiais que simulem o processo natural de seleção de indivíduos dentro de uma população, através de operadores genéticos biológicos tais como *crossover* e mutação. Desta forma, utiliza o conceito natural de seleção de Darwin dos mais aptos, para resolver problemas de otimização na forma de busca local, codificando a população em forma de cromossomos, tendo por base a genética natural de Mendel.

A população em AG são cromossomos que caracterizam o individuo. Estes são codificados geralmente em *strings* binárias {0,1}, de modo que se possa operar sobre este cromossomo através de mutação e reprodução com outros indivíduos.

Em AG, uma população inicial é gerada aleatoriamente e essa população é submetida a um ambiente para que seja verificada a existência de indivíduos que possuam características desejáveis para a solução de um problema. Através da seleção natural dos melhores indivíduos, é esperado que os mais aptos se reproduzam gerando outros cada vez mais aptos ao ambiente, ou seja, ao problema que se deseja resolver. A aptidão, ou *fitness*, é obtida avaliando-se cada individuo de acordo com a função custo a ser otimizada, variando de problema para problema. Abaixo na figura 9 um exemplo de algoritmo genético:

Algoritmo 1: Algoritmo Genético

```
função algoritmo_genetico (populacao , calc_fitness ) retornar um_individuo
        entradas: populacao  //conjunto de indivíduos
                  calc_fitness  //função que calcula a aptidão do indivíduo
1       repetir
2           nova_populacao ← conjunto vazio;
3           para i de 1 até tamanho(populacao) faça
4               x ← seleção_aleatória(populacao,calc_fitness);
5               y ← seleção_aleatória(populacao,calc_fitness);
6               filho ← reproduz(x ,y );
7               se (pequena probabilidade aleatória) então filho ← mutação(filho );
8               adicionar filho  a nova_populacao;
9           populacao ← nova_populacao;
10      até algum indivíduo estar adaptado o suficiente ou ter decorrido tempo suficiente
11      retornar o melhor indivíduo em populacao  de acordo com calc_fitness
```

FIGURA 9 - Algoritmo Genético. Adaptado do livro: *Artificial Intelligence: A Modern Approach*. Russel e Norvig (1995).

Os algoritmos genéticos têm sido utilizados em redes neurais para vários fins: treinar a rede, escolher os atributos mais relevantes ou ainda encontrar uma boa arquitetura. No treinamento, a motivação para o uso de AG, é o fato de o treinamento apresentar algumas limitações no que diz respeito ao algoritmo de retropropagação, o qual converge para mínimos locais, podendo estes não ser satisfatórios e o fato de redes muito grandes tenderem a ser computacionalmente lentas devido ao ajuste de tantos pesos. Como os AG efetuam buscas mais amplas, tem sido avaliado como um método alternativo para o ajuste de pesos e assim o treinamento de RNA.

Em Yao (1993), é apresentado um esquema simples de evolução de pesos, considerando que a rede já tenha sido geneticamente codificada:

- Decodifique cada indivíduo da geração atual num conjunto de pesos e construa a rede neural correspondente.
- Calcule o erro quadrático médio entre as saídas da rede e as saídas desejadas para cada rede neural e defina o negativo do erro como o *fitness* do indivíduo que gerou a rede (outras funções de *fitness* também são possíveis).
- Reproduza um determinado número de indivíduos a partir da população atual com probabilidade proporcional ao *fitness* ou outro critério qualquer de seleção.
- Aplique operadores genéticos, tais como *crossover* e mutação, e obtenha a nova geração.

Na seleção dos atributos mais relevantes, os AG ao auxiliarem no treinamento ou na arquitetura da rede, acabam por definir pesos ou arquiteturas que impactam na seleção ou não de atributos de forma que a rede ganhe desempenho em seu treinamento.

O problema de definição ótima de uma arquitetura de rede neural segundo Miller *et al*. (1989), pode ser visto como uma busca por uma arquitetura que apresente melhor desempenho (de acordo com algum critério) em uma tarefa específica.

Como mencionado anteriormente, a arquitetura das redes neurais artificiais são em geral definidas para cada tipo de problema através de uma concepção intuitiva, por tentativa e erro, até que se obtenha uma boa resposta ao problema. É sabido também que a arquitetura é crucial para seu desempenho, sendo um dos pontos que mais demoram na concepção de uma RNA. Assim, existem estudos que visam delinear as melhores práticas para a concepção de redes neurais artificiais. Caudill (1990), diz que a escolha do tamanho da camada intermediária deve ter como critério o bom senso: uma camada muito grande memoriza, mas não generaliza muito bem. Já uma camada muito pequena apresenta um treinamento muito

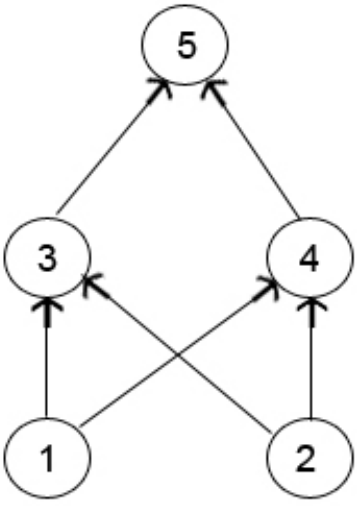
demorado. Weigend *et al.* (1990), afirmam que as redes que melhor generalizam são as menores, propondo começar com redes grandes e então seguir efetuando podas para obter a menor rede possível, dependendo do custo de cada conexão ou peso.

Outro aspecto importante relacionado aos algoritmos evolutivos para definição de arquitetura de redes neurais diz respeito à codificação de arquiteturas em indivíduos para o processo de evolução (definição do genótipo).

Os esquemas de codificação de RNA em algoritmos genéticos podem ser classificados em duas categorias principais segundo Yao e Liu (1997):

- **Codificação direta ou esquema de especificação forte:** todas as conexões e nós de uma arquitetura são especificados no genótipo (por exemplo, através de representação binária). Sua maior vantagem é a simplicidade e as desvantagens são que os cromossomos são muito grandes e ocorre a geração de muitas arquiteturas incorretas, necessitando verificação de viabilidade.

A codificação direta pode ser exemplificada conforme a figura 10 abaixo, onde é montada a codificação baseando-se na presença ou não de conexão entre os neurônios. Assim, a figura 10 a) mostra uma RNA com os neurônios 1 e 2 na camada de entrada, ambos conectados a dois neurônios (3 e 4) na camada intermediaria e que por sua vez possuem conexão para o neurônio 5 na camada de saída. A figura 10 b) mostra a tabela de codificação direta da RNA e a figura 10 c) a concatenação das linhas da tabela b) para formar o cromossomo que representa a RNA.



a)

Unidade de Origem:	1	2	3	4	5
Unidede Destino: 1	0	0	0	0	0
2	0	0	0	0	0
3	1	1	0	0	0
4	1	1	0	0	0
5	0	0	1	1	0

b)

Cromossomo: 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 1 1 0

c)

FIGURA 10 - Exemplo de RNA codificada diretamente

- **Codificação indireta ou esquema de especificação fraca (Codificação Gramatical/Sensível ao contexto):** apenas as características mais importantes de uma arquitetura, como o número de camadas escondidas e número de neurônios escondidos são codificadas. Sua vantagem é que se tenta evitar os problemas da codificação direta e sua desvantagem é que é mais complicada de implementar.

Com relação à evolução de arquiteturas, Yao (1993), apresenta também um esquema simplificado:

- Decodifique cada indivíduo da geração atual em uma arquitetura de rede neural.
- Treine cada rede neural decodificada com um algoritmo de aprendizado fixo e pré-definido.
- Calcule a aptidão de cada indivíduo baseado nos resultados do treinamento.
- Reproduza um número de indivíduos a partir da geração atual de acordo com algum critério de seleção.
- Aplique operadores genéticos, tais como *crossover* e mutação, e obtenha a nova geração.

Desta forma, pode-se ver que existem vários métodos de representação e utilização de algoritmos genéticos para criar redes neurais artificiais evolutivas, os quais permitem que a concepção de uma RNA não seja feita por tentativa e erro. É importante observar ainda com relação as RNA, como mencionam Jain *et al*. (1999), sua grande aptidão como um classificador através de um aprendizado supervisionado.

Observou-se neste capitulo a fundamentação teórica referente aos algoritmos bio-inspirados citados e o problema acerca do reconhecimento de padrões, em especial através da clusterização. Serão apresentados no capítulo a seguir, trabalhos relacionados que utilizam os algoritmos aqui explicitados.

3 TRABALHOS RELACIONADOS

Neste capítulo serão listadas algumas abordagens encontradas na literatura e relacionadas aos algoritmos bio-inspirados tratados no capítulo anterior. Primeiramente serão destacados alguns trabalhos referentes à clusterização por meio de algoritmos inspirados em colônias de formigas, em especial o baseado no *Ant Sleeping Model*, em seguida trabalhos relacionados à classificação supervisionada por meio de redes neurais artificiais evolutivas com algoritmos genéticos e por fim um trabalho que realiza a clusterização com CF e classificação supervisionada com RNA.

3.1 Clusterização com Colônia de Formigas

Após os primeiros estudos de Deneubourg *et al.* (1990) sobre clusterização com colônia de formigas e a definição do *basic model*, várias pesquisas surgiram e aprimoraram este modelo, em especial através dos pesquisadores Lumer e Faieta (1994), e Kuntz *et al.* (1997). Devido aos resultados promissores destes modelos iniciais, muitos pesquisadores passaram a utilizar algoritmos de clusterização inspirados nas CF, incrementando bastante a literatura sobre este tema.

Com base nestes modelos iniciais, ou modificações destes, vários problemas reais passaram a ser abordados com estes algoritmos. A clusterização de dados com CF, por exemplo, foi estudada por Ramos *et al*. (2002) e por Handl e Meyer (2002), para o problema de mineração de textos e web, produzindo resultados promissores, inclusive sendo introduzidas abstrações no movimento das formigas agentes para que fosse obtido maior desempenho e também a introdução de adaptabilidade automática ao parâmetro α que representa um fator de pressão dos dados e agentes.

Outro exemplo é a utilização em reconhecimento de padrões em imagens digitais, como realizado com bons resultados por Ramos e Almeida (2004), onde destacam a grande capacidade de realizar agrupamento devido às características auto-organizáveis das CF, bem como à grande vantagem de sua versatilidade para abordar problemas semelhantes com poucas modificações. Estes trabalhos tiveram como forte base a inspiração dos modelos

inicialmente propostos, principalmente modificando o modelo de Lumer e Faieta (LF), em que a inspiração primordial é o comportamento chamado *foraging*, isto é, o comportamento de busca por comida, entre as formigas, sendo visto como o trabalho destas em procurar, carregar e depositar matéria em um espaço delimitado.

Outras pesquisas presentes na literatura, também inspiradas nas colônias de formigas para realizar a clusterização, introduziram diversas modificações até mesmo na inspiração biológica, aonde se utiliza outro comportamento biológico, ou abstrações deste, das formigas, ou até mesmo hibridizando a clusterização com outros algoritmos. Exemplos destas inovações podem ser vistos em problemas que envolvem a mineração de dados *web* como abordado por Monmarché *et al*. (2003) através do algoritmo chamado *AntClust* que basicamente tratam as formigas como dados que realizam suas interações por contato direto entre si, sem estarem dispostas em uma grade, apenas controlando o contato par a par, gerando por uma medida de similaridade, colônias distintas, e, portanto, grupos de padrões bem definidos. Ainda no que se refere à mineração *web*, mais especificamente no contexto de comercio eletrônico, Abraham e Ramos (2003) utilizaram o conceito de stigmergia, através da concentração de feromônio, sem abstrair ou modificar as observações biológicas de formigas reais, realizando apenas alguns ajustes do modelo de LF, através do algoritmo ACLUSTER de Ramos *et al.* (2002), visando eliminar a grande quantidade de movimentos improdutivos. Incorporaram também uma análise de comportamento dos agrupamentos formados, por meio de programação genética linear, a fim de formar um algoritmo híbrido.

Ainda no que diz respeito ao uso de algoritmos genéticos, juntamente com a clusterização por meio de algoritmos baseados em colônia de formigas, Aranha e Iba (2006) propuseram um meio de escolher automaticamente através de AG, os parâmetros de inicialização em um algoritmo de CF baseado no modelo de Lumer e Faieta. Utilizaram como indivíduos, várias configurações de parâmetros iniciais tais como a constante que determina quando as formigas pegam ou depositam dados, a constante de controle do tamanho da vizinhança, a visibilidade de cada formiga, o tamanho do ambiente e a quantidade de formigas. Como resultados, obtiveram uma melhora na performance da clusterização se comparado com outros algoritmos da literatura em que os parâmetros são configurados manualmente.

Um trabalho que envolve puramente a tarefa de clusterização é apresentado por Monmarché *et al.* (1999), onde se explora uma forma híbrida de descobrir *clusters* em base de dados numéricas e melhorar a qualidade dos grupos formados. Desta forma, os autores denominaram de *AntClass* o algoritmo que utiliza duas formas de clusterização com colônia

de formigas, mesclado com o algoritmo *k-means*, aproveitando assim as características estocásticas do primeiro e as determinísticas do segundo. É interessante notar que esta abordagem híbrida, por adotar estas duas características, forma *clusters* de maior qualidade do que os algoritmos abordados de maneira individual, uma vez que nos modelos tradicionais, as formigas realizam muitos movimentos antes de atingir o estado desejado. É por este motivo que se utiliza o *k-means*, visando assim acelerar a convergência dos grupos em formação pela CF. Uma vez que este passo é realizado, aplica-se novamente a clusterização com a CF, porém os dados a serem agrupados agora são os grupos formados pela CF e pelo *k-means* na etapa anterior. Assim, nesta etapa a formiga passa a carregar grupos inteiros, como se o grupo fosse o próprio dado, e então o algoritmo assume uma característica hierárquica, realizando a clusterização de grupos pré-formados e não mais de dados isolados. Ao final deste novo agrupamento, aplica-se novamente o *k*-means, visando retirar erros que possam ter surgido, aprimorando assim a qualidade final dos grupos formados. Os passos envolvidos no *AntClass* podem ser resumidos como segue:

1. Realizar a clusterização dos dados através de um algoritmo de CF.
2. Aplicar *k-means* sobre a partição inicial criada no passo 1, uma vez que no passo anterior é dada uma estimativa do número correto de grupos *k*, valor este necessário para o este algoritmo.
3. Aplicar novamente a clusterização com CF, porém sobre os grupos formados após o passo 2, de forma hierárquica.
4. Aplicar novamente *k-means* sobre a partição criada após o passo 3, visando remover “ruídos” dos passos anteriores.

Os resultados obtidos com o *AntClass* são destacados como promissores em comparação com outros métodos isolados como o próprio *k-means.*

No intuito de obter maior qualidade na formação de agrupamentos, assim como o caso do algoritmo anterior, o *AntClass*, outras pesquisas surgiram e buscaram aprimorar diretamente o algoritmo de clusterização com colônia de formigas, buscando novas inspirações e até abstraindo de certa forma os conceitos tradicionais da formiga real, e desta forma, não necessariamente empregando um modelo hibrido.

Um trabalho muito interessante envolvendo este tipo de abordagem é a desenvolvida por Chen *et al.* (2004) e aprimorada por Xu *et al.* (2007), em que os autores propõe um novo modelo de CF, chamado *Ant Sleeping Model* (ASM), que visa eliminar pontos fracos dos modelos básico e de LF, pois nestes modelos o comportamento das formigas tende a um grande número de movimentos improdutivos e também devido a grande influência dos

parâmetros de inicialização do algoritmo que fica a cargo do usuário, gerando resultados muito distintos para cada problema e para cada configuração. Assim, os autores propuseram este novo modelo em que as formigas são representadas como autômatos celulares de John von Neumann, e não mais como empilhadoras de objetos, mas como agentes que representam os próprios dados e que buscam locais seguros em que possam dormir, tendo como base para isto a teoria das necessidades hierárquicas do psicólogo Abraham Harold Maslow. Complementando este modelo, os autores desenvolveram um algoritmo adaptativo que visa trazer independência dos parâmetros de configuração, de forma que a clusterização possa ser realizada em diversas bases de dados e independentemente do conhecimento do usuário sobre esta base. Este algoritmo adaptativo recebeu o nome de *Adaptive Artificial Ants Clustering Algorithm* (A^4C), onde os parâmetros são inicializados e alterados de forma adaptativa ao longo da clusterização, reagindo de acordo com a base de dados utilizada ao longo da execução do algoritmo. Abaixo, a figura 11 mostra o algoritmo A^4C em que os parâmetros são inicializados e automaticamente adaptados ao longo de sua execução.

Algoritmo 2: A^4C - *Adaptive Artificial Ants Clustering Algorithm*

```
1  Inicializar os parâmetros α, β, θ, λ, k_α, k_λ, t, t_max, s_x, s_y
2  para cada agente faça
3  |    coloque o agente em um local randômico na grade e inicialize as informações do agente
4  fim do para
5  enquanto (t <= tmax) faça
6  |    para cada agente faça
7  |    |    calcule a fitness f(agente) e a probabilidade de ativação p(agente)
8  |    |    r ← random([0,1))
9  |    |    se (r <= p(agente) então
10 |    |    |    ativar o agente e movê-lo para um local não ocupado em sua vizinhança
11 |    |    senão
12 |    |    |    permanecer dormindo no local em que está
13 |    |    fim do se
14 |    |    atualizar a classe do agente de acordo com a regra de classificação
15 |    fim do para
16 |         atualizar adaptativamente os parâmetros α e λ e incrementar t ← t + 1
17 fim do enquanto
18 exibir os agentes em cada grupo e sua localização na grade (clusterizados)
```

FIGURA 11 - Algoritmo adaptativo de clusterização com colônia de formigas, A4C. Adaptado de Xu *et al.* (2007).

Os autores descrevem formas de atualizar adaptativamente os parâmetros α (equação 13 e 14) e λ (presente na fórmula 15) como segue:

$$\alpha(t) = \alpha(t - \Delta t) - k_\alpha(\overline{f_t} - \overline{f_{t-\Delta t}}) \quad 13$$

$$\overline{f_t} = \frac{1}{n}\sum_{i=1}^{n} f_t(agente_i) \quad 14$$

Aqui, a atualização do parâmetro α necessita de um intervalo de atualização Δt, da aptidão média, f_t, de todos os agentes, na *t-ésima* iteração e uma constante k_α.

$$\lambda(t) = 2 + \frac{k_\lambda}{\overline{f_t}} \lg \frac{t_{\max}}{t} \qquad 15$$

A atualização de λ depende também da aptidão média f_t, da quantidade máxima de iterações, t_{max}, da iteração atual t e de uma constante k_λ.

Nesta abordagem, uma vez que a formiga é o próprio dado, a quantidade de formigas é igual a quantidade de dados. Este fato pode parecer computacionalmente custoso, como destaca Handl (2003) acerca do tamanho da população não ser muito menor do que a quantidade de dados, mas conforme destacam os autores desta abordagem, o custo final acaba por ser bastante reduzido se comparado com o modelo de LF por exemplo. Assim, supondo que haja ***n*** dados a serem clusterizados e ***m*** formigas, os autores afirmam que a complexidade de ambos os algoritmos seria $O(s^2mT)$, onde s^2 representa a vizinhança de cada formiga e T a quantidade de iterações exigidas pelo algoritmo. Uma vez que o A^4C exige menos iterações, este se torna então mais rápido que o modelo anterior.

3.2 Classificação Supervisionada com Redes Neurais Evolutivas

Desde o surgimento das redes neurais artificiais, o problema de classificação tem sido um dos mais pesquisados e abordados com RNA. Conforme a pesquisa de Zhang (2000), os trabalhos em que se aplicam RNA para classificação tanto supervisionada como não supervisionada tem ganhado bastante espaço na literatura, destacando-se como uma ferramenta promissora para esta tarefa. O autor destaca ainda motivos para o sucesso desta técnica, indicando fatores como o fato de ser guiada pelos dados de forma auto-adaptativa, sem que haja necessidade de suposições especificas sobre os dados. Destaca também a capacidade da rede neural em aproximar funções com uma boa precisão, o que auxilia no descobrimento de relações entre os dados, e também por ser um modelo não linear, tornando-a flexível e adaptável a vários problemas reais. Desta forma o autor faz um levantamento de várias técnicas que utilizam redes neurais artificiais para realizar a classificação, comparando-as também com classificadores tradicionais como os baseados na teoria de classificação

bayesiana. O trabalho também aborda questões como a relação entre aprendizado e generalização, métodos para redução no erro de treinamento, a importância da seleção de atributos e exemplos para um bom treinamento e também os custos associados a má classificação.

Embora seja um levantamento importante, não são abordadas alternativas evolutivas para RNA que realizam classificação supervisionada. De fato, embora algoritmos genéticos tenham sido usados em vários trabalhos para a criação de redes neurais artificiais evolutivas, poucos são os trabalhos que o fazem especificamente para o problema de classificação. Destacam-se aqui os trabalhos de Paz e Kamath (2002) e de Melo *et al.* (2009).

O primeiro é diretamente ligado a classificação supervisionada com RNA evolutiva, tendo como título a aplicação de redes neurais evolutivas para a classificação de galáxias. Nesta pesquisa, os autores destacam a problemática existente em classificar de forma manual, várias e imensas imagens espaciais obtidas à rádio, o que é bastante desgastante, demorado e impreciso. Desta forma, descrevem seis métodos de redes neurais artificiais evolutivas através de AG, com o objetivo de reconhecer galáxias duplamente dobradas (*bent double galaxies*), resolvendo assim alguns problemas comuns envolvidos na classificação por meio de RNA como por exemplo o treinamento da rede, a definição de sua topologia e a escolha dos melhores atributos.

Os autores destacam vários trabalhos que realizam a classificação de galáxias com redes neurais artificiais, mas reforçam o fato de que determinar a arquitetura de uma RNA é um processo de tentativa e erro, assim como o treinamento da rede deve ser guiado pelos dados e também o fato de que os atributos mais relevantes não são conhecidos em um primeiro momento. Assim, dão ênfase na escolha de algoritmos genéticos para automatizar a RNA de forma evolutiva, abordando cada um destes aspectos.

Após definir os parâmetros básicos dos algoritmos como a quantidade de indivíduos da população, o tipo de codificação, de seleção e tipo de cruzamento, o tipo de rede, as funções de ativação, o algoritmo de treinamento e atualização de pesos e a função de avaliação dos indivíduos, os autores realizaram experimentos da seguinte forma:

1. **Treinamento da RNA com AG, apenas com definição automática dos pesos:** Neste experimento a arquitetura da RNA é fixa e os indivíduos são vetores contendo todos os pesos da rede. Estes vetores são equivalentes aos cromossomos que representam cada individuo da população, e podem vir a ser bastante grande devido a comum característica de camadas adjacentes serem em geral totalmente conectadas. Nestes casos o custo computacional é muito grande e ocorre também

problemas com relação a permutação de pesos ao longo da execução do algoritmo. Sugere-se então que este método seja utilizado para redes pequenas e que os pesos de entrada e saída das camadas intermediárias sejam dispostos próximos uns aos outros. Após a avaliação dos indivíduos e evolução desta população, os pesos do melhor indivíduo são utilizados no treinamento da RNA.

2. **Treinamento da RNA com AG, realizando o *backpropagation* com os pesos dos indivíduos:** Este experimento é muito semelhante ao anterior, sendo modificado apenas o fato de que os indivíduos, ou vetores de pesos da rede, são utilizados no treinamento da rede com o *backpropagaion* e depois são avaliados, gerando ao final o melhor conjunto de pesos, ou seja, o melhor individuo.
3. **Seleção dos atributos que serão utilizados no treinamento da RNA:** Este experimento visa remover atributos que possam tornar a rede maior do que o necessário ou ainda reduzir a precisão de classificação desta. Desta forma, os indivíduos são codificados de forma que o cromossomo contém um bit para cada atributo que estiver presente no individuo. Após a codificação dos indivíduos, ou seja, a definição de quais atributos formam o individuo, estes são utilizados para treinar a RNA e assim fornecer uma avaliação que determinará a aptidão de cada um deles, sendo ao final escolhido o melhor individuo, isto é, o melhor conjunto de atributos.
4. **Definição da topologia da RNA com relação a quantidade de neurônios da camada intermediária, os parâmetros de treinamento do *backpropagation* e o intervalo inicial de pesos:** Nesta abordagem, a topologia de conexão da rede é fixa e os indivíduos representarão a quantidade de neurônios a serem alocados em cada camada intermediária, bem como é possível determinar esta quantidade de camadas. No trabalho os autores definiram apenas uma camada intermediária e assim buscam a quantidade de neurônios nesta camada. Outra codificação desta abordagem diz respeito a indivíduos que representem, através de quatro bits, a taxa de aprendizado do treinamento através do *backpropagation* e outros indivíduos codificando entre [0,1] os possíveis valores dos pesos. Uma vez criados os indivíduos, estes fornecem as informações necessárias para a criação e treinamento da rede. Ao final de todas as épocas em que os indivíduos são convertidos em RNA e operados geneticamente, resultam os melhores indivíduos, dentre os quais se escolhe a melhor arquitetura da rede.

5. **Definição da topologia da RNA em busca de uma matriz de conectividade:** Para este caso, o que é fixado inicialmente são os parâmetros de treinamento do backpropagation e a quantidade de camadas e de neurônios em cada uma delas. O que se deseja então é a criação de indivíduos que representem as conexões entre os neurônios. Isto é feito criando-se uma matriz de conectividade a qual é transcrita para um cromossomo de forma direta (codificação direta), concatenando as linhas desta matriz. O tamanho total do cromossomo é dado então pela quantidade de neurônios em cada camada: *(intermediária + saída) x entrada + intermediária x saída*. Uma vez definidos os indivíduos, cada um é convertido em uma RNA com a topologia descrita em seu cromossomo, sendo cada um avaliado conforme o desempenho final da RNA. Assim, após todas as épocas e conseqüente evolução dos indivíduos, escolhe-se aquele mais apto, ou seja, cujas conexões da rede ofereceram um melhor aprendizado.
6. **Definição da topologia da RNA com representação gramatical dos indivíduos:** Semelhante a anterior, utiliza-se o conceito da matriz de conectividade, deixando fixos os parâmetros de treinamento do backpropagation e a quantidade de camadas e de neurônios em cada uma delas. No entanto, esta abordagem utiliza a codificação gramatical da matriz de conectividade, não mais codificando diretamente o cromossomo como concatenação das linhas desta matriz, mas sim através de regras gramaticais que indicam as regras de transições e a formação das conexões da rede. Assim, após criação dos indivíduos como gramáticas, estas geram as redes que são treinadas e após este processo são avaliadas as melhores gramáticas, sendo escolhida aquela que apresenta a melhor conexão de rede, ou seja, quais as conexões que formam a rede mais bem treinada.

Após aplicar as seis abordagens descritas, os autores avaliaram como promissor os resultados obtidos com este classificador neural evolutivo se comparado com a mesma base de dados e outros métodos de classificação neural manual. Destacam ainda que embora promissor, existe uma desvantagem no quesito tempo computacional, uma vez que é necessário realizar vários ciclos de treinamento de várias RNA, que neste caso são os indivíduos. Apesar desta desvantagem, há alternativas como a programação paralela ou o aprimoramento da avaliação da aptidão dos indivíduos.

O trabalho de Melo *et al.* (2009), assim como o dos autores anteriores, utiliza algoritmos genéticos para a definição automática de alguns parâmetros de redes neurais artificiais, criando para tal um algoritmo neural evolutivo. Neste caso específico, abordam o

problema de previsão em séries temporais, mas a própria natureza da RNA e o treinamento supervisionado realizado fazem com que a previsão seja uma extrapolação da classificação.

A proposta destes autores difere das anteriores, pois busca através de AG, codificar a quantidade de neurônios em cada uma das camadas da RNA, utilizando para tal fim, a codificação direta dos neurônios, semelhante a codificação de uma matriz de conectividade, porém referindo-se a quantidade de neurônios e não a conexão da rede. Destacam como objetivo, reduzir o trabalho de tentativa e erro na concepção da arquitetura da RNA, no que diz respeito aos seus neurônios.

Desta forma, Melo *et al.* (2009) modificaram a codificação direta de conectividade para comportar a representação do número de neurônios em cada camada. Assim, cada indivíduo da população representa uma determinada quantidade de neurônios em cada uma das camadas. Especificamente no trabalho mencionado, utilizam apenas uma camada intermediária e varia-se a quantidade de neurônios da camada de entrada, desta única intermediária e da saída. É muito interessante notar que esta variação da quantidade de neurônios na camada de entrada, pode representar uma seleção de atributos ou de exemplos, dependendo de como os dados são apresentados a rede, sendo desta forma outra característica muito importante desta metodologia, pois com apenas uma codificação, os indivíduos desta população representam parte da arquitetura da rede como também realizam a referida seleção. Ainda neste caso especifico, utilizam o treinamento supervisionado através do *backpropagation* cujos parâmetros são definidos pelo usuário. Uma vez criada a população, cada individuo leva à criação de uma RNA que é treinada, utilizando como aptidão o erro quadrático médio após o treinamento. Assim, os indivíduos evoluem em busca de novas RNA baixas taxas de erro.

Como resultado, os autores destacam a abordagem como promissora, pois além dos fatos acima mencionados, obtiveram bons resultados de previsão se comparados com redes definidas manualmente por tentativa e erro para o mesmo problema. Como desvantagem, assim como Paz e Kamath (2002), destacam o grande tempo computacional exigido pelo algoritmo, uma vez que toda uma população de RNA é testada e operada geneticamente a cada iteração.

3.3 Clusterização e Classificação Supervisionada de Clusters com Colônia de Formigas e Redes Neurais Artificiais Evolutivas

Diante do exposto nas seções anteriores, é possível observar uma grande quantidade de algoritmos baseados em colônia de formigas para realizar a clusterização e poucos trabalhos empenhados em realizar a classificação por meio de redes neurais artificiais evolutivas. Nota-se assim a dificuldade em se encontrar na literatura, abordagens hibridas de clusterização com CF e classificação supervisionada com RNA evolutiva por meio de AG.

Nesta linha de pesquisa, embora não exatamente com estas características, se encontra o interessante trabalho de Oca *et al.* (2005) no qual utilizam uma abordagem hibrida de clusterização com colônia de formigas e classificação com redes neurais baseadas em modelos de expansão de gases, ou *growing neural gas networks* (GNGN), com o objetivo de criar *clusters* de boa qualidade. Aqui, a etapa de classificação com RNA não é evolutiva e nem supervisionada, mas a idéia hibrida do modelo é muito interessante e merece destaque, embora sem muito aprofundamento, visto que na literatura não foi encontrada uma abordagem com as características específicas citadas.

A idéia dos autores nesta abordagem é associar a capacidade de detectar padrões em base de dados através do algoritmo de clusterização baseado em colônia de formigas, com a capacidade de aprender a classificar padrões de forma não supervisionada através de redes neurais que utilizam o conceito de mapas auto-organizáveis, como é o caso da *growing neural gas network.*

Desta forma, propuseram um algoritmo hibrido em que cada formiga, baseada no modelo de LF, contém uma rede neural expansiva de gases embutida em sua estrutura. Com esta abordagem, na medida em que as formigas encontram objetos, isto é, dados, em sua vizinhança, elas treinam suas próprias redes neurais para que aprendam tanto a topologia dos dados no espaço de atributos bem como sua distribuição em uma grade bi-dimensional, que é o ambiente das formigas. Assim, os grupos formados pelas formigas se tornam classes a serem utilizadas para a classificação pelas redes neurais. Como a quantidade de *clusters* a ser formados não é conhecida, e estes sofrem muitas mudanças ao longo do algoritmo, os autores optaram pela utilização da referida GNGN que se adapta a estas modificações e incertezas.

Com esta abordagem, sempre que uma formiga encontra e pega um dado da grade, ela o classifica com sua própria rede neural que, por estar constantemente mapeando a distribuição espacial dos dados na grade, também indica à formiga qual a melhor direção a seguir para que

possa encontrar um bom local de depósito deste dado o qual está carregando. Após ser atingido um critério de parada, uma nova rede é criada, sendo esta última uma rede coletiva, criada a partir das redes individuais de cada formiga. Assim, ao final do algoritmo, uma rede estará pronta para classificar novos dados que não foram previamente clusterizados e classificados. Este algoritmo pode ser resumido pelos passos abaixo, conforme destacam os próprios autores:

1. **Inicialização:**
    a. Espalhar de forma aleatória os dados em uma grade em forma de tórus.
    b. Criar e espalhar de forma aleatória os agentes na grade, inicializando a sua direção de movimentação também de forma aleatória.
        i. Inicializar a GNGN embutida em cada agente.
2. **Treinamento e Clusterização:**
    a. Realizar movimentos aleatórios.
    b. Se o agente encontra dados em sua vizinhança, treina sua própria rede neural com estes dados.
    c. Continua buscando por dados, carregando e os depositando até que um critério de parada seja atingido com as seguintes observações:
        i. Se um objeto está sendo carregado, classifique-o com a rede neural.
        ii. Indique, de acordo com o neurônio vencedor, o local que o agente deve seguir.
3. **Classificação:**
    a. Utilizar os grupos formados como classes para a classificação.
    b. De todas as redes individuais, determinar quais neurônios farão parte da rede coletiva.
    c. Testar dados não clusterizados/classificados anteriormente, com a rede coletiva.

Como resultados desta abordagem, os autores destacam resultados promissores e também a capacidade que o modelo apresenta para lidar com bases de dados muito grandes, já que basta utilizar o algoritmo completo para uma porção dos dados, deixando o restante a cargo do classificador coletivo, o qual não precisará modificar os parâmetros de treinamento.

4 APRIMORAMENTO NA QUALIDADE DE *CLUSTERS* POR MEIO DE UM ALGORITMO HÍBRIDO BIO-INSPIRADO

Um problema comum a vários algoritmos inspirados em colônia de formigas que realizam a clusterização de dados é que, ao final do processo, os *clusters* formados muitas vezes contêm dados que não deveriam pertencer a um determinado grupo, impactando assim a qualidade final dos *clusters*. Este fato pode ocorrer por vários motivos como, por exemplo, a existência de "ruído" nos dados, a existência de dados com características pertencentes a mais de um grupo e também ao fato de que em muitos algoritmos, alguns dados ficam presos em grupos já formados ou perdidos, isto é, espacialmente distantes do grupo ao qual deveria pertencer ou ainda isolados, por exemplo, por grupos que formem uma barreira a qual acaba por impedir o dado de chegar ao seu grupo.

Neste contexto, de dados que ficam presos ou perdidos, surgem alguns questionamentos:

1. Como identificar se um *dado* está preso ou perdido se ao final da clusterização ele estará em um grupo?
2. Caso o dado seja identificado como preso ou perdido, como classificá-lo corretamente se não há informação prévia sobre a qual grupo este deve pertencer?

É buscando uma forma de aprimorar a qualidade dos clusters solucionando estas questões que neste capítulo será proposto um algoritmo híbrido bio-inspirado que realiza a clusterização de dados, inspirada em colônia de formigas através do modelo ASM, e que aprende a classificar de maneira supervisionada, inspirando-se em redes neurais evolutivas, os dados de acordo com os grupos formados pela etapa de clusterização. O intuito deste algoritmo híbrido é obter *clusters* de maior qualidade e para tal fim, é proposta a criação de um agente chamado de *agente rebocador*, o qual em conjunto com o classificador neural automático (CNA), desempenha papel fundamental no refinamento dos *clusters* e consequentemente na melhora em sua qualidade.

4.1 Colônia de Formigas e o Agente Rebocador

Para realizar a etapa de clusterização do algoritmo proposto, foi escolhido dentre os trabalhos relacionados ao tema, o algoritmo adaptativo A^4C baseado no *Ant Sleeping Model* de Xu *et al.* (2007). Esta escolha se deve principalmente por dois fatos:

1. **Desempenho:** Quando comparado com outras abordagens que utilizam o MB e o modelo de LF, o baseado no ASM produz *clusters* de maior qualidade com um número significativamente menor de iterações, sendo assim, muito veloz.
2. **Inspiração:** A própria inspiração do ASM em utilizar o dado como sendo a própria formiga-agente em busca de segurança, além de garantir o desempenho citado, fornece uma informação direta muito útil que pode ser utilizada ao final da clusterização, na determinação se o dado está preso ou perdido, ou seja, se o dado considera que está ou não em segurança.

Desta forma, o modelo adotado, o ASM, e o algoritmo que realiza a clusterização, o A^4C, são utilizados conforme foram definidos no capítulo 2, seção 2.3.2, e no capítulo 3, seção 3.1, semelhante ao algoritmo da figura 11, e conforme descritos a seguir:

1. Inicialmente devem ser inicializados os seguintes parâmetros:
    a. α – conforme a equação 10 (página 30).
    b. λ – conforme a equação 15 (página 43).
    c. β, k_α, k_λ, t_{max}, s_x, s_y– conforme necessário.
2. Dada uma base de dados *B* com *n* dados, cria-se uma grade em forma de tórus cujo tamanho é $w(n)_x h(n)$, dados pela equação 6 (página 29).
3. Após a criação da grade, esta é povoada com *n* agentes, onde cada um contém as seguintes informações, que são inicializadas no momento da criação:
    a. Vetor de atributos contendo cada atributo do dado o qual representa.
    b. Valor de sua aptidão.
    c. O estado em que se encontra: 1-Ativo 0-Dormindo
    d. Vetor *N*, cujo tamanho é dado pela equação 7 (página 29), contendo todas as posições de sua vizinhança. Utiliza-se nesta abordagem, a vizinhança expandida de Moore, onde cada agente também possui visão diagonal.
    e. Vetor *L* com todas as posições vazias de sua vizinhança.

   f. O rótulo da classe a qual pertence e que inicialmente é único para cada agente.
   g. Sua atual posição *(x,y)* na grade.

   Ao serem depositados na grade é importante lembrar que cada célula da grade pode receber apenas um agente. Na medida em que os agentes vão sendo depositados, utiliza-se um vetor de posições que auxilia o mapeamento da grade, indicando sempre a posição de cada agente.

4. Uma vez que os agentes foram depositados na grade, cria-se a matriz de dissimilaridade, onde é calculada a distância Euclidiana entre cada par de agentes conforme a equação 1.
5. Uma vez definidos os passos anteriores, deve-se iterar até o limite de vezes desejado, onde, a cada iteração, serão realizadas as seguintes operações para cada agente da grade:
   a. Calcular sua aptidão atual, dada pela equação 8 (página 30), onde o valor de α é dado na inicialização.
   b. Calcular sua probabilidade de ativação, p_a, de acordo com a equação 11 (página 30).
   c. Gerar um número aleatório no intervalo [0,1) e verificar se este número é menor ou igual a p_a.
      i. Caso seja, o agente deve ser ativado, ou seja, seu estado muda para 1, e seu rótulo volta para o estado original em que foi inicializado. Em seguida, verifica se existe posição livre na vizinhança. Caso exista, procura uma posição mais segura que sua posição atual e, se houver, desloca-se para ela.
      ii. Caso contrário, o agente dorme, ou seja, seu estado muda para 0, e seu rótulo é atualizado pelo rótulo majoritário em sua vizinhança.
6. Após todos os agentes terem realizado suas operações, deve-se atualizar os parâmetros α e λ de forma adaptativa, conforme as equações 13, 14 e 15 (páginas 42 e 43), bem como incrementar o ciclo de iterações.

Uma vez terminada todas as iterações, os grupos estarão formados, ou seja, os dados estarão clusterizados, mas neste ponto a qualidade dos *clusters* ainda é afetada pelos dados presos ou perdidos.

Desta forma, esta abordagem propõe que ao final da etapa de clusterização entre em ação um agente artificial responsável pela identificação dos agentes presos ou perdidos, bem como da quantidade de grupos formados e de seus integrantes, além de ser o responsável por disparar o treinamento dos classificadores neurais. Este agente é denominado agente rebocador (AR) e pode ser definido como: $A_r = (V_g, C, T)$ onde:

***V_g*:** Representa a visão limitada que este agente tem da grade G. Esta visão é limitada, pois o AR somente deve enxergar as posições ocupadas por agentes.

***C*:** É um vetor de agentes com tamanho igual a quantidade de agentes da grade G, que foram sinalizados como presos ou perdidos pelo AR em *T*. Assim, este vetor *C* representa a "carga" do agente rebocador, isto é, todos os agentes que estavam em posições inseguras, ou seja, presos ou perdidos.

***T*:** É um conjunto de tarefas a serem desempenhadas pelo AR dentre elas:

i. Para cada *agente* $_i$ em G, verificar a quantidade *S* de 100 vezes em que o *agente* $_i$ é ativado segundo a sua probabilidade de ativação p_a. Caso *S* seja maior ou igual a 50%, o *agente* $_i$ é sinalizado como preso ou perdido.
ii. Retirar da grade G todos os agentes que foram sinalizados pela tarefa *i* e carregá-los no vetor *C*.
iii. Verificar o rótulo de todos os agentes que permaneceram na grade, a fim de determinar a quantidade de grupos "seguros" formados.
iv. Para cada grupo identificado, disparar o treinamento de um CNA, apresentando apenas 70% dos dados de cada grupo para treinamento e o restante para validação.
v. Após o treinamento do CNA para cada grupo, apresentar os agentes rebocados um a um para cada classificador, atribuindo um rótulo ao agente de acordo com o classificador que gerou menor erro de classificação. Após rotular um agente, o AR deve inseri-lo novamente na grade em uma posição adjacente a de um membro do grupo ao qual foi rotulado. Pode-se ainda adotar uma taxa de erro aceitável na classificação de modo que, se nenhum CNA retornar um valor aceitável, o agente é considerado órfão (possivelmente um dado ruidoso). Assim, é alocado em uma posição próxima, mas não adjacente, ao grupo cujo classificador ofereceu o menor erro.

Algumas considerações acerca do agente rebocador devem ser destacadas, como por exemplo, o motivo pelo qual se testa cada agente 100 vezes sobre sua ativação ou não. Esta tarefa é necessária, pois ao final do passo 6 do algoritmo de clusterização, a característica

probabilística de ativar ou não, pode ativar um agente que de fato esteja seguro e mesmo assim neste caso, o agente terá seu rótulo alterado para seu rótulo original e, no entanto, este não deve ser caracterizado como um agente preso ou perdido. A verificação repetida 100 vezes visa assim extrair uma medida mais confiável em relação a condição do agente.

Outra característica importante de se destacar a respeito do AR é que, mesmo removendo da grade os agentes que possam ter sido agrupados corretamente, o AR sempre deixará na grade os que estiverem mais seguros. Assim, aqueles que foram rebocados serão, de qualquer forma, devolvidos a grade após terem sido classificados pelo CNA.

Dando sequência aos passos desta abordagem, tem-se:

7. Inicializar o agente rebocador, apresentando-lhe a grade após o passo 6. Desta forma, obtêm-se os dados necessários para que o AR realize as tarefas i à v.

4.2 Classificador Neural Automático

Para seguir com a abordagem, é necessário a utilização de um classificador neural automático, sendo utilizado o modelo de rede neural evolutivo proposto por Melo *et al.* (2009), em que se aplica algoritmos genéticos para que sejam determinadas as quantidades de neurônios em cada camada da rede, possibilitando também a seleção de atributos dos dados de treinamento da rede.

Sabe-se que o AR solicitará a criação de um CNA para cada grupo formado e que este classificador é uma RNA evolutiva, para a qual o agente rebocador irá apresentar como entrada, os atributos que compõe cada dado de forma individual. Desta forma, ao se apresentar um conjunto de atributos que definem um dado, espera-se como saída da RNA o grupo ao qual o dado pertence. Desta forma, quando a rede estiver treinada, deve ser capaz de classificar os agentes, ou seja, os dados, que foram rebocados.

Baseado no trabalho de Melo *et al.* (2009), que serve de modelo para esta abordagem, a RNA aqui implementada é uma rede multicamadas, de topologia totalmente conectada e que utiliza o backpropagation como algoritmo de aprendizagem supervisionado.

A quantidade de neurônios nas camadas de entrada e intermediária pode variar, sendo nesta abordagem a quantidade mínima de neurônios na camada de entrada igual a metade da quantidade de atributos que definem o dado, e a quantidade máxima igual a quantidade de

atributos. Na proposta utilizou-se apenas uma camada intermediaria, pois conforme Munakata (2007), além do tempo de processamento aumentar, uma segunda camada intermediária pode ser mais bem aproveitada se houver alguma análise dos dados que influencie a inicialização dos pesos entre a camada de entrada e a intermediária. Sendo assim, na camada intermediária a quantidade mínima é igual a quantidade de neurônios da camada de entrada e a quantidade máxima igual a 10 vezes a mínima. É importante lembrar que quanto mais neurônios, mais conexões existirão e mais pesos deverão ser atualizados, sendo por este motivo evitado em geral a utilização de redes com muitas camadas e com uma quantidade muito grande de neurônios em cada uma delas.

A inicialização dos pesos da RNA utilizada é feito de forma aleatória e a função de ativação da rede é a mesma descrita na equação 12. A apresentação do conjunto de treinamento, que deve ser normalizado entre 0 e 1 devido a sigmóide utilizada na ativação, é realizada apresentando-se todos os dados a cada ciclo de treinamento. Ao final de cada ciclo é calculado o erro médio de treinamento e o erro médio de validação é então calculado apresentando à rede os dados para validação.

Foram adotadas as seguintes condições de parada para o treinamento:

i. O erro médio de validação atinge um valor aceitável.
ii. O número máximo de iterações seja atingido.
iii. A rede decora os dados de treinamento, ou seja, o erro de treinamento diminui e o erro de validação aumenta.

A codificação direta da rede para o cromossomo do AG também segue o modelo adotado pelos autores no referido trabalho e desta forma o cromossomo possui comprimento igual a:

$$Comprimento\_Cromossomo = max\_entrada + max\_intermediária + 1 \quad (16)$$

onde, *max_entrada* é a quantidade máxima de neurônios na camada de entrada, *max_intermediaria* a quantidade máxima na camada intermediaria e 1 referente a camada de saída, pois a saída da RNA desejada deve ter apenas um neurônio.

O próximo passo é gerar aleatoriamente o número de neurônios de cada camada, respeitando cada intervalo mínimo e máximo. Uma vez gerada a quantidade de neurônios em cada camada, supondo QENT na entrada e QINT na intermediária, deve-se preencher o cromossomo em três partes como exibido na figura 12 e exposto abaixo:

i. Deve-se preencher o cromossomo da posição 0 até a posição do numero de neurônios na camada de entrada, QENT, com o valor ‘1’, indicando que há tantos neurônios na camada de entrada quanto forem os ‘1s’ presentes na parte inicial do cromossomo.
O restante, ou seja, da posição QENT até *max_entrada* deve ser preenchido com ‘0s, caso o numero de neurônios na entrada não seja igual ao máximo permitido.

ii. Depois, deve-se preencher o cromossomo da posição *max_entrada* até QINT com ‘1s’ e caso QINT não seja igual ao *max_intermediaria*, preencher da posição *max_entrada* + QINT até *max_intermediaria* com ‘0s’.

iii. Por fim, deve-se preencher com ‘1’ a posição do cromossomo referente à saída.

MIN_ENTRADA	3
MAX_ENTRADA	5
MIN_INTERMEDIARIA	3
MAX_INTERMEDIARIA	10
MIN_SAIDA	1
MAX_SAIDA	3

COMPRIMENTO	18

QENT	3
QINT	6
QSAI	2

CROMOSSOMO

posicoes de entrada					posições da intermediaria										posições saida		
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
1	1	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	0

FIGURA 12 - Exemplo de codificação direta da quantidade de neurônios em várias camadas.Melo *et al.* (2009).

Uma vez codificada, a rede é tratada como um individuo dentro de uma população de outras RNA. É possível então, através dos operadores genéticos *crossover* e mutação, efetuar a seleção da RNA mais apta para classificar os grupos formados pela clusterização e assim reconhecer o melhor grupo para os agentes presos ou perdidos, uma vez que o AG é um algoritmo de busca e neste caso, deverá encontrar através das gerações das populações o individuo, ou RNA, mais apta.

Para mensurar então o individuo mais apto, a aptidão de cada individuo será determinada em função do negativo do erro relativo médio (ERM) de validação da RNA da mesma forma que exposto em Melo *et al.* (2009):

$$ERM = \frac{1}{N}\left[\sum_{i=1}^{N}\left|\frac{Vd_i - Vc_i}{Vd_i}\right|\right] \qquad 17$$

onde, N é o número de classificações realizadas durante o treinamento, Vc_i é o valor calculado e Vd_i o valor desejado. Dessa forma, destaca-se como mais apto, o individuo que apresentar o

menor ERM, significando assim que realizou boa generalização e é uma RNA candidata como melhor classificador.

Outro aspecto importante a ser definido é o método de seleção adotado para a escolha dos pares que irão cruzar. Nesta abordagem, utiliza-se uma forma simples onde se deve ordenar por ordem de aptidão todos os indivíduos da população inicial e selecionar apenas a primeira metade, ou seja, os mais aptos. O passo seguinte é cruzar seqüencialmente 2 a 2, o primeiro com o segundo, o terceiro com o quarto e assim por diante até que se obtenha uma nova população formada pela melhor metade da geração anterior com os seus respectivos filhos.

Para efetuar o cruzamento, foi adotado o sistema de ponto de corte único do cromossomo. Assim, é gerado aleatoriamente um número entre um intervalo de corte, IC para a escolha dos trechos de cromossomos que darão origem aos novos indivíduos.

$$\frac{\max\_entrada}{2} \leq IC \leq \left\lceil \frac{\max\_entrada + \max\_itermediaria}{2} \right\rceil \qquad 18$$

Após ter sido gerada a nova população, aplica-se a mutação cuja taxa adotada é o inverso do tamanho do comprimento do cromossomo, de forma que com esta codificação, a mutação troca o valor 0 por 1 e vice versa.

Dando sequência aos passos gerais desta abordagem, tem-se:

8. Para cada grupo distinto apresentado pelo AR, gerar uma população de RNA de acordo com o modelo apresentado.
9. Para cada individuo, ou seja, RNA, da população, o agente rebocador deve apresentar como conjunto de treinamento 70% dos dados que formam o grupo o qual se deseja aprender a classificar. Os 30% restante deve ser utilizado como conjunto de validação. A apresentação dos dados deve ser feita um a um, ou seja, em grupos de atributos, de modo que a RNA poderá utilizar todos, caso a quantidade de neurônios na entrada seja igual a quantidade de atributos, ou realizando uma seleção destes, quando a quantidade de neurônios na entrada for menor.
10. Enquanto o erro médio de validação não for aceitável, a quantidade máxima de iterações não tiver sido atingida e a rede não estiver decorando a classificação, deve-se repetir o passo 9.
11. Quando o individuo sair do passo 10, deve-se atualizar sua aptidão conforme descrito pela equação 17 e em seguida deve-se efetuar os passos 9 e 10 para todos os outros indivíduos.

12. Quando todos os indivíduos, ou seja, RNA, estiverem treinados deve-se selecionar a metade mais apta e realizar as operações de seleção genética descritos no modelo. Estes passos deverão ser realizados de acordo com a quantidade de gerações a serem criadas.
13. Uma vez terminada a seleção automática do melhor classificador neural dentre os testados para cada *cluster*, o AR deve apresentar todos os agentes contidos no seu vetor de carga *C* para todos os CNA. Ao obter a melhor classificação, isto é, o menor erro na saída da RNA classificadora, o agente é rotulado de acordo com o grupo do classificador e assim o AR deverá introduzi-lo novamente na grade, conforme a tarefa T_v.

A seguir, a figura 13 mostra o fluxo passo a passo e a figura 14 o algoritmo proposto:

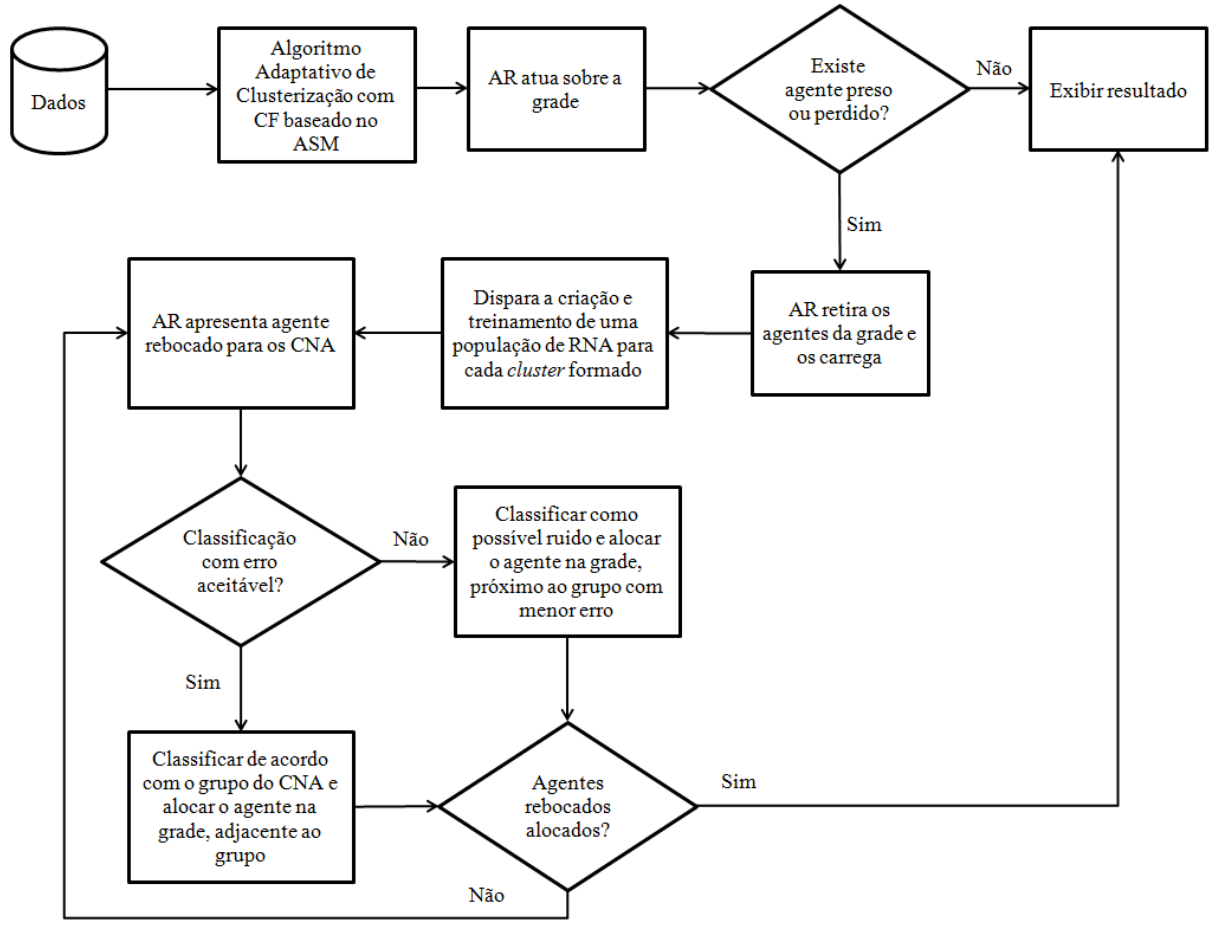


FIGURA 13 - Fluxo passo a passo da abordagem proposta.

Algoritmo 3: Colônia de Formigas + Agente Rebocador + RNA evolutiva

```
1  Inicializar os parâmetros α, λ, β, k_α, k_λ, t_max, s_x, s_y
2  criar a grade
3  para cada agente faça
4  |    coloque o agente em um local randômico na grade e inicialize as informações do agente
5  fim do para
6  criar matriz de dissimilaridade
7  enquanto (t <= tmax) faça
8  |    para cada agente faça
9  |    |    calcule a fitness f(agente) e a probabilidade de ativação p(agente)
10 |    |    r ← random([0,1))
11 |    |    se (r <= p(agente) então
12 |    |    |    ativar o agente // alterar seu estado para 1
13 |    |    |    rotular o agente com a mesma classe em que foi criado
14 |    |    |    se (posições livres na vizinhança > 0) então
15 |    |    |    |    buscar o local mais seguro dentre as posições disponíveis
16 |    |    |    |    se (local mais seguro vizinho > segurança atual) então
17 |    |    |    |    |    mover o agente para a posição escolhida
18 |    |    |    |    fim do se
19 |    |    |    fim do se
20 |    |    senão
21 |    |    |    permanecer dormindo no local em que está
22 |    |    |    rotular o agente com a classe majoritária em sua vizinhança
23 |    |    fim do se
24 |    fim do para
25 |    |    atualizar adaptativamente os parâmetros α e λ e incrementar t ← t + 1
26 fim do enquanto
27 inicializar o agente rebocador
28 para cada agente faça
29 |    calcule a probabilidade de ativação p(agente)
30 |    verificar por 100 vezes, em quantas o agente é ativado
31 |    se (quantidade de ativações >= 50%) então
32 |    |    sinalizar o gente como preso ou perdido removendo-o da grade
   |    |    e adicinando-o ao vetor de carga C do agente recobador
33 |    senão
34 |    |    registrar quais e quantos são os grupos distintos formados
35 |    fim do se
36 fim do para
37 para cada grupo identificado faça
38 |    criar população de RNA
39 |    enquanto (quantide de gerações < máximo de gerações) faça
40 |    |    para cada individuo da população faça
41 |    |    |    enquanto ((ERM > aceitável) E (iterações < iterMax) E (decorar = falso)) faça
42 |    |    |    |    agente rebocador apresenta um a um, 70% dos dados do grupo,
   |    |    |    |    treinando a RNA, codificada pelo individuo, e validando-a com
   |    |    |    |    os 30% restante, calculado o ERM e verificando se a rede está decorando
43 |    |    |    |    iterações ← iterações + 1
44 |    |    |    fim do enquanto
45 |    |    |    atualizar apitdão do indivíduo
46 |    |    fim do para
47 |    |    selecionar a metade mais apta e efetuar as operações genéticas
48 |    |    quantidade de gerações ← quantidade de gerações + 1
49 |    fim do enquanto
50 fim do para
51 para cada agente no vetor de carga C faça
52 |    apresentar o agente para cada RNA responsável pela classificação de cada grupo
53 |    rotular o agente com a classe da RNA que gerou o menor erro de classificação
54 |    posicionar o agente em local adjacente ao grupo em que foi classificado
55 fim para
56 exibir os agentes em cada grupo e sua localização na grade (clusterizados)
```

FIGURA 14 - Algoritmo Proposto.

Após estes passos, espera-se que os clusters estejam livres de agentes presos ou perdidos, o que diminuiria a sua qualidade, e que a formação final dos *clusters* seja a desejada.

5 AVALIAÇÃO DA ABORDAGEM PROPOSTA

Neste capítulo serão apresentadas a metodologia utilizada para implementar e testar a abordagem proposta, as configurações utilizadas nos testes e os resultado obtidos em comparação aos encontrados na literatura.

5.1 Metodologia

Para avaliar a abordagem proposta, foram realizados alguns testes com as bases de dados reais *Iris*, *Wine*, *Soybean(Small)*, *Glass* e *Thyroid (new)*, disponíveis no repositório de dados para aprendizagem de máquina - UCI. A escolha destas se deu para que o resultado pudesse ser comparado com os obtidos por Xu *et al.* (2007), uma vez que no presente trabalho foi utilizado o mesmo modelo de colônia de formigas, o *Ant Sleeping Model*.

As métricas utilizadas para realizar a avaliação foram a quantidade máxima de iterações, a quantidade mínima, máxima e média de erros, o percentual de erros e o tempo médio de execução. Estas são as mesmas utilizadas no trabalho dos referidos autores, visando assim uma comparação direta. Desta forma, para que os valores fossem obtidos, o algoritmo foi executado 100 vezes para cada conjunto de dados. Este valor não apresenta motivo especial, apenas para arbitrariamente formar uma amostra de resultados. Destaca-se ainda que para obter os valores referentes aos erros de classificação dos grupos, o resultado da clusterização foi comparado com a classificação original, já conhecida.

Com relação às bases utilizadas, a tabela 1 mostra um resumo de suas características e são descritas como segue:

Iris: Contém dados reais multivariados, cujos atributos refletem características medidas sobre as flores de plantas do gênero *Iris*. Este conjunto contém 150 exemplos divididos em 3 classes, cada uma com 50 instâncias e 4 atributos. Sabe-se também que apenas uma das classes é linearmente separável das outras.

Wine: Contém dados inteiros e reais multivariados, em que os atributos referem-se à análise química da composição de vários tipos de vinhos italianos. Neste conjunto encontram-

se 178 exemplos divididos em 3 classes, uma com 59 instâncias, outra com 71 e a última com 48, cada uma sendo composta por 13 atributos contínuos.

Soybean (Small): Esta base é uma versão reduzida de uma base maior que contém características sobre doenças na soja. Encontra-se aqui 47 exemplos divididos em 4 classes, três delas com 10 instâncias e outra com 17, sendo cada uma composta por 35 atributos.

Glass: Contém dados reais multivariados sobre tipos de vidro, sendo utilizada para classificar fragmentos de vidro deixados em cenas de crime. A referida base contém 214 exemplos com 9 atributos cada, sendo divididos em 6 classes. A distribuição de instâncias por classe é: 70, 76, 17, 13, 9, 29.

Thyroid (new): Esta é uma versão nova e reduzida de uma base que contém informações sobre pacientes com diversas disfunções da glândula tireóide. Encontra-se aqui 215 exemplos com 5 atributos cada, em uma divisão de 3 classes com a seguinte quantidade de instâncias: 150, 35, 30.

TABELA 1 - Resumo da composição das bases de dados utilizadas.

	Bases				
	Iris	***Wine***	***Soybean (Small)***	***Glass***	***Thyroid (new)***
Exemplos	150	178	47	214	215
Classes	3	3	4	6	3
Instâncias por Classe	50	59 / 71 / 48	10 / 10 / 10 / 17	70 / 76 / 17 / 13 / 9 / 29	150 / 35 / 30
Atributos	4	13	35	9	5

Conforme exposto no capítulo anterior, os dados que serão apresentados à RNA devem ser normalizados entre 0 e 1. Desta forma, as bases citadas foram inicialmente modificadas para atender a esta condição.

Uma vez que a abordagem proposta contém um algoritmo inspirado em colônia de formigas, baseado no ASM e A^4C de Xu *et al*. (2007), os parâmetros de inicialização da CF são iguais para todos os testes e são semelhantes aos utilizados pelos autores. Assim, tem-se que:

Tamanho da vizinhança: Assim como utilizado pelos autores citados, a vizinhança tem tamanho 3x3, logo $s_x = s_y = 1$.

λ: O valor inicial é dado pela equação 15, enquanto que no referido trabalho é inicializado com o valor 2.

α: O valor inicial é dado pela equação 10, enquanto que no trabalho dos referidos autores o valor inicial utilizado foi de 0.4483 para a versão adaptativa (A^4C) e 0,3 para a versão estática (SA^4C).

β: Valor constante e igual a 0.1.

k_α: Valor constante e igual a 0.5.

k_λ: Valor constante e igual a 1.

t_{max}**:** O número máximo de iterações da CF será 5.000.

Outra diferença com relação ao A^4C padrão, é a movimentação do agente quando ativo. Neste trabalho o agente sempre busca o local mais seguro, enquanto no outro esta escolha é condicionada a uma probabilidade de 90%, dada por $\theta = 0.9$.

Com relação à utilização da rede neural artificial, os parâmetros fixos de treinamento utilizados nos testes foram:

Taxa de aprendizagem: Fixada em 0.08, quando o grupo a ser classificado contém menos de 50 dados para treinamento e em 0.15, caso contrário.

Número máximo de iterações: Foi estabelecido o limite de 20 mil iterações, em parte devido ao tempo de execução e, por outro lado, em decorrência da quantidade de dados de treinamento de cada RNA ser pequena.

Condição de parada da RNA: Atingir o número máximo de iterações, ou obter um erro médio de validação na ordem de 10^{-4}, ou ainda ocorrer por 500 iterações consecutivas, uma diminuição do erro de treinamento e aumento do erro de validação, ou seja, a rede estará decorando ao invés de generalizar.

Ainda com relação a RNA, existem parâmetros variáveis de sua arquitetura que serão determinados através do algoritmo genético, o que a torna uma rede evolutiva. Neste trabalho, variam-se as quantidades de neurônios nas camadas de entrada e intermediária, mantendo-se fixa a topologia de conexão e a quantidade de camadas intermediária conforme exposto no capítulo anterior. Assim, as informações necessárias para a utilização do AG são:

Número de neurônios na camada de entrada: Este valor será no mínimo, a metade da quantidade de atributos da base de dados utilizada e no máximo a própria quantidade. Assim, existirá a possibilidade de atributos serem ou não selecionados, visto que cada neurônio da camada de entrada refere-se a um atributo.

Número de neurônios na camada intermediária: Este valor será no mínimo, a quantidade de neurônios da camada de entrada e no máximo 5 vezes este valor. Desta forma, será possível automatizar a definição da melhor estrutura de processamento interno da rede.

Dados da população: Serão criados 30 indivíduos, sendo escolhidos a cada geração, os 15 mais aptos. Estes serão os progenitores da nova geração.

Condição de parada do AG: Obter a vigésima (20ª) geração de indivíduos da população.

É importante observar que o agente rebocador é adaptável à grade e, portanto, não necessita de configurações iniciais. No entanto, pode-se adotar a taxa de erro mínima aceitável para que se aceite a classificação dada por um CNA. Neste trabalho adotou-se como aceitável um erro da ordem de 10^{-3}.

Desta forma, foi possível realizar uma comparação entre a clusterização realizada puramente com a colônia de formiga e a realizada com o algoritmo híbrido de CF, RNA e AG, ambas sobre os mesmos conjuntos de dados.

5.2 Configuração dos experimentos

Seguindo a metodologia apresentada, alguns parâmetros do AG são dependentes da base utilizada. Desta forma, as tabela 2 e 3 mostram as configurações dos experimentos para cada um dos conjuntos de dados.

TABELA 2 - Configurações dos experimentos.

Configurações dos experimentos com a base *Iris*
Número de neurônios na camada de entrada: 2 a 4
Número de neurônios na camada intermediária: quantidade na entrada a 20
Configurações dos experimentos com a base *Wine*
Número de neurônios na camada de entrada: 6 à 13
Número de neurônios na camada intermediária: quantidade na entrada a 65
Configurações dos experimentos com a base *Soybean (Small)*
Número de neurônios na camada de entrada: 17 a 35
Número de neurônios na camada intermediária: quantidade na entrada a 175
Configurações dos experimentos com a base *Glass*
Número de neurônios na camada de entrada: 4 a 9
Número de neurônios na camada intermediária: quantidade na entrada a 45

TABELA 3 - Configurações dos experimentos.

Configurações dos experimentos com a base *Thyroid (new)*
Número de neurônios na camada de entrada: 2 a 5
Número de neurônios na camada intermediária: quantidade na entrada a 25
Configurações fixas para todas as bases
Quantidade de execuções de cada base: 100
Quantidade de iterações da clusterização com CF: 5.000
Quantidade de indivíduos na população de RNA: 30
Quantidade de gerações máxima: 20
Quantidade de iterações do treinamento das RNA: 20.000
*Outros parâmetros de inicialização de acordo como descrito na metodologia

Nota-se com esta abordagem que muitos parâmetros ficam a cargo do próprio algoritmo, tornando todo o processo bastante automatizado.

5.3 Resultados

Nesta seção são apresentados os resultados obtidos após a aplicação do algoritmo proposto, aqui denominado Bio-Cluster ACGAANN (referente à clusterização bio-inspirada em *ant colony* (AC), *genetic algorithm* (GA) e *artificial neural network* (ANN)), nas bases de dados mencionadas, de acordo com a metodologia e configurações descritas nas seções anteriores.

Antes de seguir com a análise dos resultados quantitativos, é interessante exibir como a abordagem deste trabalho se comporta de maneira visual. Desta forma, as figuras a seguir mostrarão o processo de formação e o resultado dos *clusters* na base *Iris*, quando em uma das execuções do algoritmo, ocorreu a identificação correta de todos os grupos.

A figura 15 mostra o estado inicial em que os agentes (formigas) estão dispostos na grade, cada um representando uma classe. Em seguida, na figura 16, é exibido o resultado da clusterização pelo algoritmo de CF.

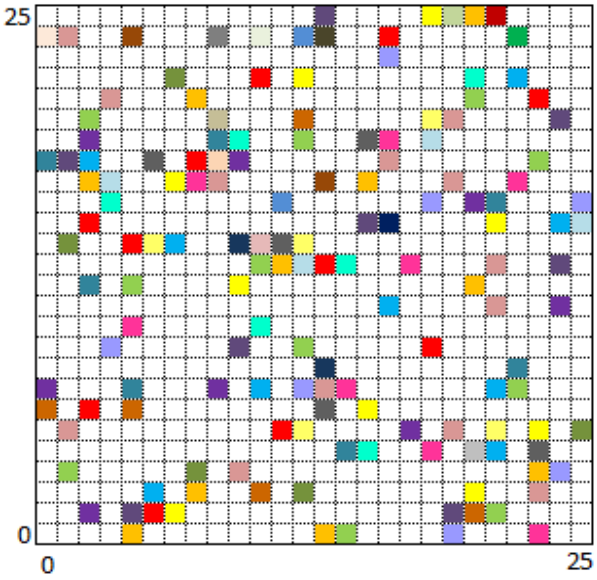


FIGURA 15 - Formigas (agentes) inicialmente dispostas na grade com dados da base *Iris*.

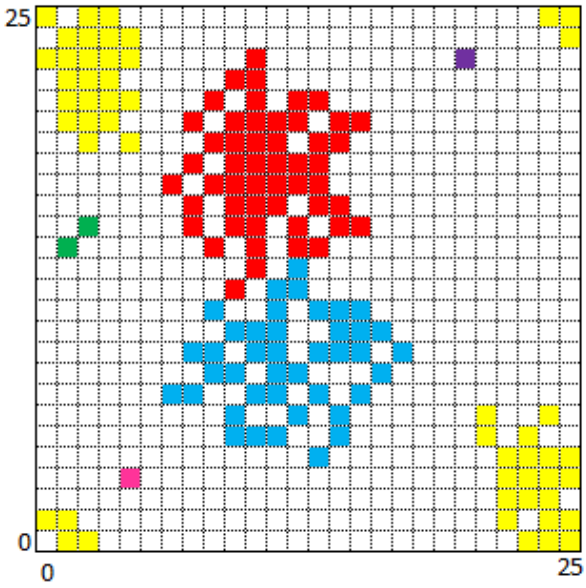


FIGURA 16 - Clusters da base Iris, formados pela CF.

Na figura 16 é possível perceber justamente um dos fatores que motivou a presente pesquisa, isto é, a existência, após a clusterização, de agentes perdidos. Estes estão representados pelas cores verde, lilás e roxo. Nota-se ainda que esta imagem representa os grupos reconhecidos pela CF, de modo que além dos perdidos, pode haver agentes presos. Assim, a figura 17 mostra, com base nesta disposição dos *clusters*, a classificação real.

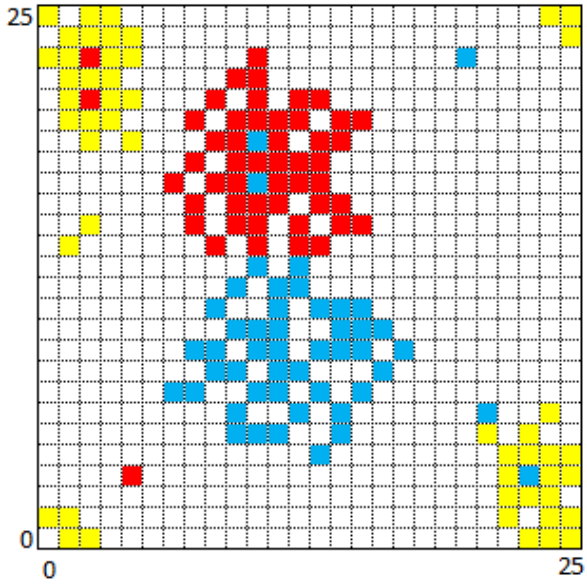


FIGURA 17 - Clusters da base Iris, formados pela CF, marcados com a classe real.

É possível notar através figura 17 que realmente existem dados presos. Dois agentes vermelhos estão presos no grupo amarelo, dois azuis estão presos no grupo vermelho e um azul está preso no grupo amarelo. Percebe-se também que havia um agente azul no grupo amarelo e os que estavam perdidos deveriam pertencer aos *clusters* maiores, de modo que os dois verdes deveriam estar no grupo amarelo, o lilás no grupo vermelho e o roxo no grupo azul.

Visando aprimorar a qualidade dos clusters formados, o agente rebocador identifica quais agentes estão ativos, isto é, que não estão seguros. A figura 18 (a) mostra a identificação destes agentes na grade e a figura 18 (b), a visão “limitada” do AR.

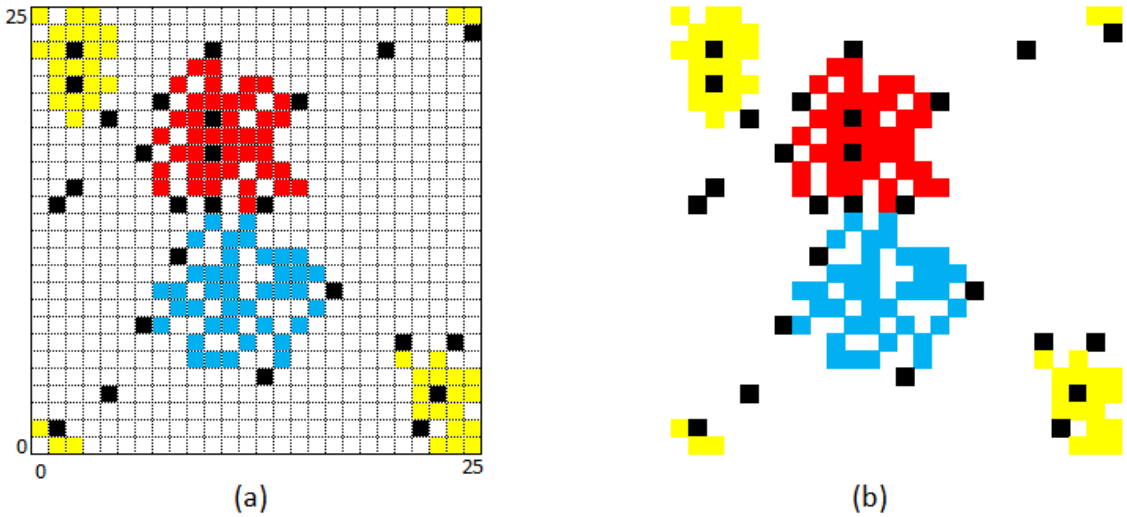


FIGURA 18: Identificação dos agentes inseguros (a). Visão do agente rebocador (b).

Observando a figura 18 é interessante notar que neste caso, além dos 5 agentes presos e dos 5 perdidos, outros 14 foram marcados como inseguros. Estes também são removidos da grade e apresentados aos CNA para que possam ter sua classificação confirmada, sendo então alocados em uma posição segura, próxima ao seu grupo. Por fim, a figura 19 (a) mostra o ambiente sem os agentes rebocados e, a figura 19 (b), a realocação terminada, isto é, o estado final da clusterização híbrida proposta.

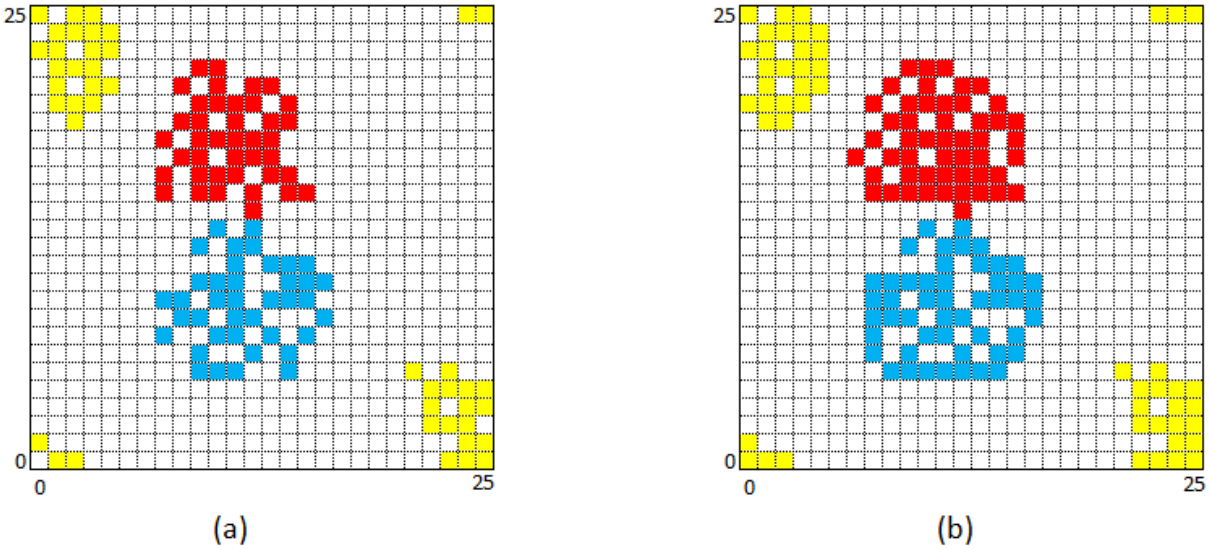


FIGURA 19 - Grade sem agentes inseguros (a). Final da clusterização híbrida (b).

5.3.1 Análise Comparativa

Nesta subseção, os resultados serão comparados aos obtidos com algoritmos baseados no modelo LF, no A^4C adaptativo e no A^4C estático (SA^4C), presentes no trabalho de Xu *et al.*(2007). Assim, serão utilizadas tabelas onde constam: o número de iterações e, após as 100 execuções dos algoritmos, a quantidade mínima, máxima e média de erros ocorridos na classificação dos dados, o percentual destes erros e o tempo médio de execução.

Nos testes comparativos com a base *Iris*, exibidos na tabela 4, percebe-se através dos resultados que a abordagem proposta de fato melhorou a qualidade final da clusterização para este conjunto de dados. Nota-se que a menor quantidade de erros que antes eram 2 no A^4C, chegou a zero. Isto quer dizer que dentre as 100 execuções do algoritmo ACGAANN, houve aquelas em que os dados foram todos agrupados corretamente. Além disto, a quantidade

máxima de erros também foi reduzida, de 5 para 2, e no geral a quantidade de erros e o percentual destes também foi menor do que nos demais algoritmos.

Um dos fatores que dificultam a distinção dos padrões nesta base é o fato de haver duas classes linearmente inseparáveis. Deste modo, é comum ocorrer a presença de dados mesclados entre as classes, sendo estes possivelmente capturados pelo agente rebocador para posterior re-classificação.

TABELA 4 - Resultado comparativo da base *Iris.*

Resultados com a base *Iris*				
Algoritmo:	*LF	*SA^{4}C	*A^{4}C	BioCluster-ACGAANN
Número máximo de iterações do AG e RNA	NA	NA	NA	20 e 20.000
Número máximo de iterações da CF	1.000.000	5.000	5.000	5000
Menor quantidade de erros	3	2	2	**0**
Maior quantidade de erros	13	8	5	**2**
Quantidade média de erros	6.68	4.39	2.13	**1.05**
Percentual de erros	4.45%	2.94%	1.31%	**0.95%**
Tempo médio de execução (s)	56.81	**1.36**	1.43	1025.35 (~17min)

*LF, SA4C e A4C: resultados de Xu *et al.* (2007)

Com relação aos testes comparativos com a base *Wine*, são exibidos na tabela 5, nota-se novamente uma redução na quantidade máxima, média e percentual de erros. Este fato mostra mais uma vez a melhora geral do processo de clusterização com a abordagem proposta, mesmo as classes do conjunto de dados *Wine* sendo considerada na literatura como "comportadas".

TABELA 5 - Resultado comparativo da base *Wine.*

Resultados com a base *Wine*				
Algoritmo:	*LF	*SA^{4}C	*A^{4}C	BioCluster-ACGAANN
Número máximo de iterações do AG e RNA	NA	NA	NA	20 e 20.000
Número máximo de iterações da CF	1.000.000	5.000	5.000	5000
Menor quantidade de erros	0	0	0	**0**
Maior quantidade de erros	18	11	6	**2**
Quantidade média de erros	8.05	4.83	2.02	**0.92**
Percentual de erros	4.52%	2.71%	1.13%	**0.86%**
Tempo médio de execução (s)	62.37	**1.44**	1.52	1242.48 (~20min)

*LF, SA4C e A4C: resultados de Xu *et al.* (2007)

Nos testes comparativos com a base *Soybean (Small)*, exibidos na tabela 6, houve também a melhora no resultado, embora menos expressiva. Como neste conjunto de dados a quantidade de exemplos e de instâncias por classe é muito pequena, quando o AR retira agentes da grade, os grupos remanescentes ficam ainda menores. Este fato dificulta um bom aprendizado da RNA classificadora. Sendo assim, era esperado que em algumas das execuções houvesse uma quantidade de erros semelhante a dos outros algoritmos, muito embora a média e o percentual de erros tenha sido melhor.

TABELA 6 - Resultado comparativo da base *Soybean (Small).*

Resultados com a base *Soybean (Small)*				
Algoritmo:	*LF	*SA^{4}C	*A^{4}C	BioCluster-ACGAANN
Número máximo de iterações do AG e RNA	NA	NA	NA	20 e 20.000
Número máximo de iterações da CF	1.000.000	5.000	5.000	5000
Menor quantidade de erros	0	0	0	**0**
Maior quantidade de erros	4	2	2	**2**
Quantidade média de erros	1.40	0.86	0.57	**0.21**
Percentual de erros	2.98%	1.83%	1.24%	**0.57%**
Tempo médio de execução (s)	17.68	**0.42**	0.44	905.45 (~15min)

*LF, SA4C e A4C: resultados de Xu *et al.* (2007)

Finalmente, os testes comparativos com as bases *Glass*, tabela 7, e *Thyroid (new)*, tabela 8, comprovam, de maneira mais expressiva, a melhora na qualidade geral do processo de clusterização com esta nova abordagem. Em ambos os casos a menor quantidade de erros foi reduzida para 1, sendo o motivo semelhante ao caso da *Soybean (Small)*, isto é, existem classes cuja quantidade de instâncias é bem reduzida, impactando negativamente no treinamento e classificação da RNA. Assim, é possível que um agente destas classes, quando rebocado, não seja classificado pelo CNA responsável pelo grupo correto.

No entanto, de maneira geral, a maior quantidade de erros presente nesta abordagem foi reduzida significantemente, além da média e do percentual de erros também terem diminuído. Observa-se este fato, principalmente, na base *Thyroid (new)*, onde estes valores foram reduzido praticamente pela metade.

TABELA 7 - Resultado comparativo da base *Glass.*

Resultados com a base *Glass*				
Algoritmo:	*LF	*SA^{4}C	*A^{4}C	BioCluster-ACGAANN
Número máximo de iterações do AG e RNA	NA	NA	NA	20 e 20.000
Número máximo de iterações da CF	1.000.000	5.000	5.000	5000
Menor quantidade de erros	7	4	2	**1**
Maior quantidade de erros	12	12	10	**4**
Quantidade média de erros	10.25	7.59	3.94	**2.06**
Percentual de erros	4.79%	3.55%	1.84%	**1.03%**
Tempo médio de execução (s)	106.21	**2.37**	2.44	1621.42 (~27min)

*LF, SA4C e A4C: resultados de Xu *et al.* (2007)

TABELA 8 - Resultado comparativo da base *Thyroid (new).*

Resultados com a base *Thyroid (new)*				
Algoritmo:	*LF	*SA^{4}C	*A^{4}C	BioCluster-ACGAANN
Número máximo de iterações do AG e RNA	NA	NA	NA	20 e 20.000
Número máximo de iterações da CF	1.000.000	5.000	5.000	5000
Menor quantidade de erros	10	7	5	**1**
Maior quantidade de erros	16	13	11	**5**
Quantidade média de erros	14.53	10.76	8.81	**4.13**
Percentual de erros	6.75%	5.00%	4.10%	**2.05%**
Tempo médio de execução (s)	106.50	**2.38**	2.46	1398.15 (~23min)

*LF, SA4C e A4C: resultados de Xu *et al.* (2007)

Além dos fatores já mencionados, é importante notar que em todos os testes o tempo médio de execução do algoritmo proposto foi bastante elevado. Este fato era esperado, pois conforme mencionado nos capítulos anteriores, a utilização conjunta de redes neurais e algoritmos genéticos, em geral, requer um elevado processamento computacional. No caso específico desta abordagem, serão criadas 30 redes neurais (30 indivíduos), as quais irão treinar, no máximo, por 20mil iterações. Após o treinamento, ocorrerá a seleção genética, sendo criadas, a cada geração (no total de 20), 15 novas RNA. Isto equivale a treinar 30 + 19x15 redes, gerando no máximo um total de 6.300.000 iterações de treinamento.

Pode-se observar também que o tempo de execução aumenta em conjunto com a quantidade de classes e de atributos. O primeiro impacta na criação de mais redes e o segundo, no tamanho destas.

CONCLUSÃO

Reconhecer padrões é uma atividade cotidiana realizada por nosso cérebro, que é uma máquina de processamento muito potente, capaz de receber diversas informações, processá-las a uma velocidade impressionante e ainda fornecer uma análise dos dados que as compõem. Deste modo, podemos reconhecer o tipo e as características da informação recebida. No entanto, embora tenhamos esta incrível capacidade, existem conjuntos de dados muito grandes, sobre os quais nossa máquina de processamento biológico não consegue extrair informação útil, como reconhecer algum padrão presente nestes dados.

Desta forma, tem-se estudado diversas técnicas e algoritmos que aproveitem o grande poder de processamento matemático computacional, para desenvolver sistemas capazes de reconhecer padrões em uma massa de dados extensa e rica em detalhes, que fogem da capacidade cerebral humana de processamento.

Embora existam vários algoritmos e sistemas bio-inspirados empenhados nesta tarefa, a literatura mostra que grande parte dos avanços é realizada no sentido de melhorar o desempenho ou a qualidade, com algoritmos bastante específicos. Poucos são os casos em que se busca uma nova inspiração biológica, ou ainda, uma nova abordagem que forme um sistema integrado de soluções bio-inspiradas, capaz de utilizar os benefícios que cada uma oferece.

A partir deste contexto é que a presente dissertação foi desenvolvida, oferecendo assim, uma nova abordagem híbrida de bio-inspirações, capaz de melhorar a qualidade no reconhecimento de padrões. Para atingir este objetivo, foram agregados vários algoritmos bio-inspirados apresentados na literatura, incorporando-se modificações artificiais e bio-inspiradas, desenvolvidas ao longo da pesquisa.

Assim, um algoritmo bio-inspirado híbrido e adaptativo foi desenvolvido, para tratar *clusters* em que existam dados presos ou perdidos. Utilizou-se então uma colônia de formigas nos moldes do *Ant Sleeping Model*, como clusterizador, um agente artificial, denominado agente rebocador, responsável pelo refinamento dos grupos formados e, várias redes neurais artificiais projetadas automaticamente com algoritmos genéticos, como classificadores.

O algoritmo foi testado com as bases de dados *Iris, Wine, Soybean (Small), Glass* e *Thyroid (new),* gerando resultados de acordo com o objetivo da pesquisa, isto é, melhorando, comparativamente com outros resultados da literatura, a qualidade do reconhecimento dos padrões presentes nos dados, através do refinamento dos *clusters*.

Destaca-se a relevância do trabalho no que diz respeito a sua contribuição à literatura de algoritmos bio-inspirados. A abordagem aqui detalhada contém conceitos bastante inovadores, bem como uma base teórica sólida, sendo assim capaz de inspirar novas soluções e tratar vários problemas em que estes tipos de algoritmos têm sido empregados.

Dentre as diversas possibilidades criadas através do estudo realizado para a elaboração desta dissertação, podemos citar como trabalhos futuros:

Aplicar em bases maiores: Neste trabalho foram utilizadas bases reais, porém utilizadas para testar a abordagem proposta. Será interessante aplicar o algoritmo desenvolvido em bases maiores e em bases que de fato não se conhecem os padrões, validando o modelo para aplicações reais.

Aprimorar a identificação dos agentes presos ou perdidos: A identificação dos agentes a serem rebocados pode ser aprimorada, como por exemplo, através de uma memória simples embutida no agente. Assim, pode-se controlar quais rótulos ele recebeu, quantas vezes pertenceu a cada classe e quantas vezes foi ativado em cada uma delas. Assim, é possível definir outros parâmetros que auxiliem nessa decisão, fator que também influência a qualidade final dos grupos formados.

Tratar o tempo de execução: Apesar de ser conhecida a problemática do tempo de execução de redes neurais artificiais evolutivas, existem algumas possibilidades de atacar esta questão, como implementar o algoritmo de forma paralela, aproveitando os recursos de *hardware*, ou ainda, aprimorando a função de avaliação de cada individuo, reduzindo sempre que possível os ciclos de treinamento. Outra tratativa seria modificar o esquema de codificação dos indivíduos.

Evoluir outros aspectos das redes neurais: Assim como apresentados na literatura, existem vários outros aspectos das RNA que podem ser evoluídos, como peso e topologia. Incorporar este tipo de autonomia na definição das redes, pode gerar resultados bastante interessantes.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAHAM, A.; GUO, H.; LIU, H. Swarm intelligence: foundations, perspectives and applications, in: N. Nedjah, L. Mourelle (Eds.), Swarm Intelligent Systems, Studies in Computational Intelligence, Springer-Verlag,Germany, 2006.

ABRAHAM, A.; RAMOS, V. Web usage mining using artificial ant colony clustering and genetic programming. Proceesings of IEEE Congress on Evolutionary Computation, Australia, 1384-1391. 2003.

ARANHA, C.; IBA, H. "The Effect of Using Evolutionary Algorithms on Ant Clustering Techniques", Proceedings of the 2006 Asia Pacific Workshop on Genetic Programming (ASPGP06), pp. 24-34.,2006.

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. Swarm Intelligence: From Natural to Artificial Systems, Santa Fe Institute in the Sciences of the Complexity, Oxford University Press, New York, Oxford, 1999.

BONGARD, J. Biologically Inspired Computing. IEEE Computer, 42(4): 95-98. 2009.

CAMPOS, L. Metáforas Biológicas Combinadas Para Projeto de Redes Neurais Artificiais. Dissertação de Mestrado – Universidade Federal de Santa Catarina, UFSC, Brasil. 2001.

CAUDILL, M. 1990. Using neural nets: Diagnostic expert nets. Al Expert 5, pp 43- 47. 1990
CHEN, L.; XU X.; CHEN, Y. An Adaptive Ant Colony Clustering Algorithm, Proc. Third International Conference on Machine Learning and Cybernetics (ICMLC’04), 1387–1392. 2004.

CHEN, L.; XU, X.; CHEN, Y. An Adaptive Ant Colony Clustering Algorithm, Proc. Third International Conference on Machine Learning and Cybernetics (ICMLC’04), 1387–1392. 2004.

DENEUBOURG, J.; GOSS, S.; FRANKS, N.; SENDOVA, A.; DETRAIN, C.; CHRETIEN, L. The Dynamic of Collective Sorting Robot-like Ants and Ant-like Robots. 1st Conf. On Simulation of Adaptive Behavior: From Animals to Animats, J.A. Meyer and S.W.Wilson, eds, MIT Press, pp. 356–365. 1990.

DUBES, R. Cluster analysis and related issues. In Handbook of Pattern Recognition & Computer Vision. Pg. 3–32. 1993.

GRASSÉ, P. "La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositerm.es Natalensis et Cubitermes sp. La theorie de la Stigmergie: Essai d'interpretation du Comportement des Termites Constructeurs." Insect. Soc. 6 41-80. 1959.

HANDL, J. Ant-based methods for tasks of clustering and topographic mapping: improvements, evaluation and comparison with alternative methods. Dissertação de Mestrado – Universidade de Erlangen-Nürnberg, Alemanha. 2003.

HANDL, J.; KNOWLES, J.; DORIGO, M. On the performance of ant-based clustering. In Design and Application of Hybrid Intelligent Systems, Vol. 104 of Frontiers in Artificial Intelligence and Applications (pp. 204-213). Amsterdam, The Netherlands: IOS Pres. 2003

HANDL, J.; MEYER, B. Improved Ant-based Clustering and Sorting in a Document Retrieval Interface. In Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature, Vol. 2439 of Lecture Notes in Computer Science (pp. 913-923). Berlin, Germany: Springer-Verlag. 2002.

HAUPT, L.; HAUPT, E. Practical Genetic Algorithms. John Wiley & Sons Inc., New York, USA. 1998.

HOLLAND, J. Adaption in Natural and Artificial Systems. University of Michigan Press. 1975.

IYODA, M. Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas. Tese de Mestrado – Unicamp. 2000.

JAIN, A.; DUBES, R. Algorithms for Clustering Data. Prentice-Hall advanced reference series. 1988

JAIN, A.; MURTY, M.; FLYNN, P. Data clustering: a review. ACM Computing Surveys, Vol. 31, No. 3, pp. 264–323, 1999.

JESAN, J. The neural approach to pattern recognition. Ubiquity: An ACM IT Magazine and Forum, 2004, URL http://www.acm.org/ubiquity/views/v5i7_jesan.html, acessado em junho de 2010.

KASABOV, K. N. Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering. MIT Press. London, England. 1998

KUNTZ, P.; Layzell, P.; Snyers, D. "A Colony of Ant-Like Agents for Partitioning in VLSI Technology." In Proceedings Fourth European Conference on Artificial Life, pg. 417-424. MIT Press, 1997.

LUMER, E.; FAIETA, B. Diversity and adaptation in populations of clustering ants. Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animats, (Vol. 3), J.-A. Meyer and S.W. Wilson, eds, MIT Press/Bradford Books, Cambridge,MA, pp. 501–508. 1994.

MELO, G.; CAMPOS, G; SILVA, J.; SOUZA, J. Projeto Automático de Redes Neurais Artificiais Para o Problema de Previsão de Séries Temporais. In: XLI SBPO - Simposio Brasileiro de Pesquisa Operacional, 2009.

MILLER, G.; TODD, P.; HEGDE, S. Designing neural networks using genetic algorithms. Proceedings of the Third International Conference on Genetic Algorithms. pg. 379-384. 1989.

MITCHELL, M. An Introduction to Genetic Algorithms. MIT Press. 1997.

MONMARCHÉ, N.; LABROCHE, N.; VENTURINI, G. Web sessions clustering with artificial ants colonies. Laboratoire d'Informatique de l'Universit ´e de Tours. 2003.

MONMARCHÉ, N.; SLIMANE, M.; VENTURINI, G. AntClass: discovery of clusters in numeric data by hybridization of an ant colony with Kmeans algorithm. Laboratoire d'Informatique de l'Universit ´e de Tours, Internal Report no 213, E3i. 1999.

MUNAKATA, T. Fundamentals of the New Artificial Intelligence. Springer. 2008.

OCA, M.; GARRIDO, L.; AGUIRRE, J. An hybridization of an ant-based clustering algorithm with growing neural gas networks for classification tasks. Proceedings of the ACM symposium on Applied computing. 2005

PAZ, E; KAMATH, C. Evolving Neural Networks for the Classification of Galaxies. Proceedings of the Genetic and Evolutionary Computation Conference. 2002.

RAMOS, V.; ALMEIDA, F. Artificial Ant Colonies in Digital Image Habitats – A Mass Behaviour Effect Study on Pattern Recognition. 2004.

RAMOS, V.; MUGE, F.; PINA, P. Self-Organized Data and Image Retrieval as a Consequence of Inter-Dynamic Synergistic Relationships in Artificial Ant Colonies, in Javier

Ruiz-del-Solar, Ajith Abraham and Mario Köppen (Eds.), Frontiers in Artificial Intelligence and Applications, Soft Computing Systems - Design, Management and Applications, 2nd Int. Conf. on Hybrid Intelligent Systems, IOS Press, Vol. 87, ISBN 1 5860 32976, pp. 500-509, Santiago, Chile, Dec. 2002.

RUMELHART, D.; HINTON, G.; WILLIAMS, R. Learning internal representations by error propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition (Vol 1: Foundations) MIT Press. 1986.

UCI - Machine Learning Repository – URL http://archive.ics.uci.edu/ml/datasets.html Acessado em outubro de 2010.

WEIGEND, A.; RUMELHART, D.; HUBERMAN, B. Back-propagation, Weight-Elimination and Time Series Prediction. Proceedings of the Connectionist Models Summer School. 1990.

WERBOS, P. The Roots of Backpropagation. Beyond Regression. 1974.

XU X.; CHEN, L.; HE P. A Novel Ant Clustering Algorithm Based on Cellular Automata, Proc. Web Inteligence and Agent Systems: An international journal 5. IOS Press, pp. 1570-1263. 2007.

YAO, X; LIU, Y. A New Evolutionary System for Evolving Artificial Neural Networks. IEEE Transactions on Neural Networks, 8. pp 694-713. 1997.

YAO, X. Evolutionary artificial neural networks. International Journal of Neural Systems, 4(3):203-222. 1993.

ZHANG, G. Neural Networks for Classification: A Survey. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 30, Nº 4, Nov, 2000.