ESCOLA POLITÉCNICA
DE PERNAMBUCO

d.s.c. UPE
Departamento de Sistemas e Computação

IMPLEMENTAÇÃO DE UM MECANISMO AUTOMÁTICO DE CONTROLE DE VIZINHANÇAS ENTRE AGENTES DE UMA POPULAÇÃO INSPIRADA EM ALCATÉIA

Trabalho de Conclusão de Curso

Engenharia da Computação

Nome do Aluno: Luís Filipe de Araújo Pessoa
Orientador: Prof. Fernando Buarque de Lima Neto, PhD

UNIVERSIDADE
DE PERNAMBUCO

LUÍS FILIPE DE ARAÚJO PESSOA

IMPLEMENTAÇÃO DE UM MECANISMO AUTOMÁTICO DE CONTROLE DE VIZINHANÇAS ENTRE AGENTES DE UMA POPULAÇÃO INSPIRADA EM ALCATÉIA

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

Recife, junho de 2009.

Agradecimentos

À Deus, em primeiro lugar, por me sustentar e guiar minhas ações e pensamentos em todos os momentos de minha vida. Agradeço pela Sua Fidelidade, pelos dons a mim concedidos, pela minha Família e Amigos, os quais são como ouro para mim. Não poderia deixar de agradecê-Lo por me ajudar, desde o início, a realizar o presente Curso. À Ele sejam dados todo o Louvor e Graças por mais essa etapa alcançada de minha vida.

Agradeço a todos da minha Família, em especial minha mãe Carmem Cursino, a qual nunca poupou esforços para que eu pudesse estudar e chegar onde estou. Às minhas tias, tios, primos e primas que sempre me apoiaram e me incentivaram em minhas investidas, inclusive patrocinando meus estudos por diversas vezes. À minha avó Cacilda Cursino (e mãe) pelo seu carinho, zelo, amor e orientação. Por fim, agradeço a meu avô José Soares (e pai), exemplo de Homem e caráter, cuja lembrança carrego comigo todos os dias. Por fim, agradeço a meu pai e irmãos paternos que, mesmo de longe, acompanharam meu desenvolvimento.

Agradeço a minha Namorada (em breve, Esposa) Fernanda Lima pelo seu Amor, carinho, compreensão, paciência, companheirismo, apoio, sabedoria, humildade, lealdade e fidelidade há quase 7 anos. Sua companhia e colaboração tornaram possível a realização deste trabalho. Agradeço também a sua família que sempre nos apoiou.

Aos meus Amigos da Igreja, da Universidade e do trabalho que me ajudaram, direta ou indiretamente, neste trabalho e a não desistir de meus objetivos. Agradeço pelas conversas nos corredores, pelos trabalhos realizados em grupo, pelo aprendizado compartilhado, pelos momentos de descontração e divertimento. Eles são imprescindíveis para mim.

Agradeço a todos os Professores do Departamento de Computação pelo empenho e dedicação em construir um bom curso. Tive a oportunidade de aprender sobre computação e sobre a vida com todos eles. Agradeço, em especial, ao professor Fernando Buarque pela sua orientação e conselhos pertinentes durante toda a graduação. Seus pensamentos e idéias foram imprescindíveis no desenvolvimento deste trabalho. Compartilho com ele essa minha conquista e os resultados deste.

Agradeço, também, à Sérgio Terra, pela sua compreensão e ajuda indireta para o desenvolvimento deste trabalho.

Luís Filipe de Araújo Pessoa

Resumo

Wolf-Pack Approximation (WPA) é um projeto de pesquisa liderado pelo Prof. Fernando Buarque que objetiva desenvolver um novo algoritmo de aproximação de funções inspirado no comportamento de alcatéias. WPA é, portanto, uma nova meta-heurística que se propõe como uma alternativa computacionalmente barata (em termos de processamento e memória) para atingir seu objetivo. Sua concepção pode ser dividida em dois módulos: o primeiro se destina ao controle dos comportamentos e estratégias de uma alcatéia no ambiente do WPA para uma dada nuvem de pontos (representada por uma floresta no ambiente computacional); o segundo utiliza os dados de saída do módulo um para calcular a função que se deseja aproximar.

Este trabalho tem por objetivo propor uma modelagem e implementação do primeiro módulo do WPA, cuja saída é uma lista de posicionamento e raio final para cada lobo. Através de técnicas de Inteligência Artificial, como Sistemas Multi-Agentes e Computação Social – inspirada no comportamento de populações naturais (*i.e.* uma alcatéia), foram definidos e construídos o repertório de comportamentos dos lobos, assim como as características da dinâmica do ambiente referentes ao módulo um do WPA.

Como prova do conceito aqui apresentado, foram realizadas simulações de aproximação com quatro funções diferentes: duas polinomiais e duas trigonométricas. Essas simulações objetivaram determinar o grau de influência no resultado final dos parâmetros escolhidos, além de apresentar os bons resultados obtidos com o algoritmo desenvolvido.

Baseado na boa qualidade dos resultados obtidos, este trabalho representa um passo importante para a viabilização de uma inovadora ferramenta inteligente de aproximação de funções baseada em inteligência coletiva.

Abstract

Wolf-Pack Approximation (WPA) is a research project led by Prof. Fernando Buarque of University of Pernambuco, which aims to develop a new algorithm for approximation of functions based on the behavior of wolf-packs in nature. Hence, WPA is a metaheuristics posed as a computationally unexpensive alternative (in terms of processing and memory) to achieve its goal. Its architecture can be divided into two modules: the first is for controlling behavior and strategies of a wolf-pack in the WPA environment given a cloud of points; target points are represented in the environment by the forest. The second module uses the output of the first to approximate the desired function.

This work aims to be the initial approach and implementation of the first module of the WPA, which generated output is a list of position and radius for each wolf of the artificial pack. Through techniques of Artificial Intelligence such as Multi-Agent Systems and Social Computing inspired by the behavior of natural populations (*i.e.* a wolf-packs), a repertoire of behavior of wolves were defined and constructed, together with the environment and its dynamics.

Some simulations were performed to prove the concept based on four different functions: two polynomial and two trigonometric functions. The simulation set aims to determine the degree of influence on the outcome of some selected parameters, and present the results obtained so far by the new algorithm.

Therefore, this work represents an important step towards the construction of a comprehensive tool for approximation of functions based on collective intelligence.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

(Dispostos por ordem de aparição no texto)

RNA – Redes Neurais Artificiais

WPA – *Wolf-Pack Approximation*

BDI – *Belief-Desire-Intention*

FSS – *Fish School Search*

UML - *Unified Modeling Language*

EMM – Erro Médio Mínimo (para o lobo)

DPM – Desvio-Padrão Mínimo (para o lobo)

Capítulo 1

Introdução

1.1 Motivação

Recentemente, surgiram diversas técnicas e algoritmos inspirados no comportamento de animais na natureza para a resolução de problemas. Essa abordagem mostra-se *promissora*, pelo seu poder de abstração e eficácia, porém *desafiadora*, pois requer conhecimento em diferentes áreas de atuação. Esse tipo de computação, denominada Computação inspirada na Biologia, é apenas uma das subáreas da Computação Natural [7], a qual ainda engloba, por exemplo, a Biocomputação e a Biologia inspirada na Computação. Existem na literatura diversos exemplos de sucesso utilizando Computação Bioinspirada [1][21].

Através de metáforas com os fenômenos da Natureza, pode-se construir um modelo computacional que se proponha a simular ambientes e resolver problemas complexos, de forma mais *intuitiva* [5], ou com *menos custo* computacional. Algumas técnicas de Inteligência Artificial são utilizadas para esses fins, principalmente: Computação Evolutiva [6][16], Sistemas Multi-agente Inteligentes [30][40] e Redes Neurais Artificiais (RNA) [4]. Computação Natural é considerada por muitos um novo paradigma dentro de computação e inteligência artificial.

As técnicas existentes de aproximação de funções são de relativa complexidade e de alto custo computacional. Abordagens que incluem RNA [23] possuem um alto custo de processamento (devido às diversas fases do treinamento), tempo de execução excessivo, além da possibilidade de acontecer *overfitting* e a função não ser aproximada com a precisão esperada.

Dessa forma, o WPA (*Wolf-Pack Approximation*) [26] busca utilizar os benefícios da Computação Natural para propor uma alternativa computacionalmente barata para aproximação de funções. Sua idéia principal é utilizar a metáfora com uma alcatéia para que, a partir da inteligência coletiva, seja possível a extração da função aproximada para uma dada

nuvem de pontos. Isto dado pela surpreendente e eficaz dinâmica de uma alcatéia na natureza ao, por exemplo, caçar.

O ambiente proposto para o WPA é composto por árvores - que representam um ponto da nuvem de pontos de entrada, por ovelhas – que representam as presas dos lobos (ou pontos a serem "caçados") e por lobos – que irão caçar as ovelhas. O que se imaginou foi que após a caça, os lobos estarão estrategicamente posicionados, de forma que seja possível a aplicação de técnicas matemáticas (*e.g.* interpolação ou extrapolação) para a determinação da função aproximada que representa melhor a nuvem de pontos de entrada. Sendo que esta operação não é objetivo deste trabalho.

Para um melhor desenvolvimento, é possível dividir o WPA em dois módulos: o primeiro implementa os mecanismos de interação entre os componentes do ambiente e determina o posicionamento do lobo após a caça; já o segundo, utiliza a saída do primeiro módulo para encontrar a função que melhor aproxima a nuvem de pontos de entrada. Devido à complexidade de cada um desses módulos, este trabalho se propõe a implementar apenas o primeiro módulo do WPA.

1.2 Objetivos e Resultados Esperados

O objetivo principal deste projeto é o desenvolvimento do primeiro módulo do WPA, a saber: um mecanismo automático de posicionamento dos lobos organizados em uma alcatéia. Os lobos poderão: (i) cercar as ovelhas, (ii) delimitar territórios, (iii) formar bandos ou (iv) coordenar ações sobre o rebanho, de maneira que seu objetivo seja cumprido dentro do plano de ação do algoritmo do WPA. Ao se posicionar corretamente toda a alcatéia, torna possível a extração de uma função de aproximação que descreve bem o conjunto de pontos de entrada.

Essa modelagem será aqui implementada através de Sistemas Multi-agentes inteligentes e aprendizado local usando cálculo de distâncias.

A partir desses axiomas, é possível estabelecer os seguintes objetivos e metas específicos deste trabalho:

I) Modelar a estrutura do ambiente e dos agentes inteligentes no WPA:

- Representação da nuvem de pontos da função;
- Representação das ovelhas;
- Representação dos lobos;

II) Definir quais os operadores que cada agente possuirá e seus respectivos objetivos;

III) Definir como será a interação entre os agentes do sistema para que os lobos sejam posicionados de maneira que facilite o processo de extração da função aproximada (o módulo de aproximação da função se encontra fora do escopo deste projeto).

O principal resultado consistirá em um algoritmo que possibilite o posicionamento correto dos lobos da matilha, de acordo com seus operadores e a configuração do ambiente que estão inseridos (distribuição da nuvem de pontos e das ovelhas, assim como o número de lobos existentes na alcatéia). Trata-se aqui, portanto, do módulo inicial do WPA. Uma abordagem inovadora para aproximar funções a partir de uma modelagem baseada em agentes inteligentes. Note-se que o módulo seguinte do WPA (fora do escopo deste trabalho) deverá aperfeiçoar a modelagem inicial e implementar mecanismos para aproximação da função, dado o posicionamento final dos lobos, resultante do módulo que este trabalho visa implementar.

Como um resultado esperado mais imediato, teremos um módulo inicial e funcional do WPA, que possibilitará uma maior discussão a respeito dessa abordagem, sujeitando-o a sucessivas melhorias. Como um resultado futuro, espera-se que o resultado desse trabalho sirva como fundamento para a construção do algoritmo completo, uma abordagem alternativa para aproximação de funções capaz de tratar eficientemente problemas complexos, utilizando Computação Bioinspirada.

1.3 Metodologia

A metodologia seguida nesse trabalho está definida abaixo. Entretanto, salienta-se que apesar de estruturada em uma ordem de execução, algumas das atividades foram executadas mais de uma vez, quando se fez necessária (*e.g.* redefinição das características da alcatéia, nos casos em que os testes não haviam sido satisfatórios).

- Estudo da Literatura:

- Estudo do comportamento de uma alcatéia na natureza;
- Reflexão sobre aspectos computacionais do comportamento de uma alcatéia;
- Sistemas multi-agentes inteligentes cooperativos e altruístas;
- Inteligência Coletiva;
- Computação Social;
- Computação Natural;
- Computação Bio-inspirada

- Definição das características de uma alcatéia que serão modeladas:
  - Comportamentos sociais coletivos e individuais que serão fundamentais para a sua modelagem;
  - Interação com o ambiente em que os lobos estarão inseridos.
- Implementação do modelo computacional do comportamento de uma alcatéia no contexto do WPA:
  - Influência dos comportamentos referentes às ovelhas, lobos e o papel da floresta – nuvem de pontos – no ambiente modelado para atingir o objetivo desejado;
  - Definir como os conceitos e teorias pesquisados durante o estudo da literatura podem contribuir na construção do algoritmo desejado;
  - Definir como ocorrerão os ajustes no decorrer da execução.
- Realização de experimentos no modelo computacional construído:
  - Executar o algoritmo para diferentes tipos de função (diferentes configurações da floresta);
  - Verificar qual a configuração de parâmetros que obteve menor erro para a posição final dos lobos.

1.4 Estrutura do Trabalho

A organização do presente trabalho é apresentada a seguir. Primeiramente, foram abordados os conteúdos teóricos que dão sustentação para o desenvolvimento do trabalho. Em seguida, toda a modelagem e implementação da contribuição é descrita e são mostrados e analisados os resultados obtidos nas simulações realizadas. Por fim, são apresentadas as conclusões finais e perspectivas futuras.

Capítulo 1. Contém uma descrição do problema a ser tratado por este trabalho, assim como suas motivações, objetivos, metodologias e resultados esperados.

Capítulo 2. São apresentados aqui todos os principais conceitos relacionados com a modelagem e implementação deste projeto. Os aspectos mais importantes do comportamento de uma alcatéia na natureza, sua comunicação e estrutura social são descritos e, em seguida são abordados os conceitos mais importantes para este trabalho relativos aos Sistemas Multi-Agente e Computação Social, com ênfase em ambientes cooperativos, os quais possibilitarão a modelagem deste trabalho. Por fim, são feitas considerações sobre a importância da aproximação de funções para várias áreas de conhecimento.

Capítulo 3. Descreve como foi realizada a modelagem sob os aspectos dos comportamentos de uma alcatéia que serviram de inspiração para o trabalho e das características dos Sistemas Multi-Agente e Computação Social incorporadas. Em seguida, apresentam-se os aspectos envolvidos na implementação, tais como plataforma de desenvolvimento, ferramentas e padrões de desenvolvimento utilizados, além de uma descrição dos principais operadores e mecanismos.

Capítulo 4. Contém os resultados das simulações e análises procedidas sobre os resultados obtidos.

Capítulo 5. São apresentadas as conclusões sobre o trabalho desenvolvido. Contém uma discussão final sobre aos resultados obtidos, são apresentadas sugestões de trabalhos futuros, além de conter algumas das dificuldades encontradas no desenvolvimento deste trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo, são discutidos todos os conceitos e teorias que foram utilizados para a modelagem e implementação do presente trabalho. De início, são discutidos os aspectos relevantes sobre o comportamento individual de lobos e de uma alcatéia na natureza. O foco principal desse estudo será o comportamento cooperativo do bando.

Em seguida, são discutidos aspectos de Computação Inteligente que dão suporte ao desenvolvimento desse projeto, tais como: Computação Social e Sistemas Multi-Agentes Inteligentes.

Por fim, são brevemente abordados também alguns métodos de aproximação de funções e seus aspectos computacionais.

2.1 Comportamento de uma Alcatéia na Natureza

As alcatéias geralmente são formadas por grupos entre 8 a 15 lobos [38], dependendo do tamanho de seu território, da abundância de comida e por fatores sociais. Elas possuem uma hierarquia bem definida, assumidas através da postura do lobo, características comportamentais e lutas pela dominância do bando.

O casal de lobos dominantes é denominado de *alfa*. Seu *status* social é conquistado através de seus comportamentos e atitudes, ao invés do seu tamanho e sua força. Entretanto, sua liderança consiste na liberdade que tem para escolher para onde ir e o que fazer. O restante do bando o acompanha em suas decisões em demonstração de respeito. Logo, não se faz necessária a imposição de suas vontades para o restante dos membros. O casal *alfa* é o primeiro a se alimentar e o único que pode se reproduzir dentro de uma alcatéia [24].

Seguindo essa hierarquia, existem os lobos *beta* que exercem uma liderança logo abaixo dos *alfa*. Enquanto isso, na base da pirâmide, encontra-se o lobo *ômega*, o qual não possui quase nenhum direito dentro do grupo. Ao contrário, torna-se alvo das injúrias e agressões do restante dos membros e é obrigado a cuidar dos filhotes do casal dominante. O

ômega possui uma expressão corporal bastante típica: rabo entre as pernas, orelhas e cabeça baixas e, algumas vezes, levantam a pata para mostrar o estômago e genitália; enquanto os *alfas* possuem uma postura e rabo eretos, uivos longos e dentes à mostra [39]. A Figura 1 mostra um caso de dominância de um *alfa* sobre o *ômega*.

Figura 1. Dominância de um lobo *alfa* sobre o *ômega*.

Ao contrário do que muitos pensam, os lobos não se comunicam apenas através dos uivos, mas, também, através da sua postura corporal e dos odores [18]. Através de sua expressão corporal, é possível demonstrar medo, submissão, dominância, felicidade, fúria, etc. Um exemplo é a demonstração de dominância e submissão relatada anteriormente.

Os odores servem, principalmente, para a alcatéia delimitar seu território, o qual pode atingir 500 km². Isso somente é possível devido ao seu faro apurado, o que possibilita a detecção de odores dez vezes melhor do que um cachorro [17]. Eles delimitam seu território através da urina e das fezes. Quando um lobo "estrangeiro" sente aquele cheiro, ele toma conhecimento que aquela área está ocupada e se ultrapassar, estará invadindo o espaço de outro bando. Seu olfato aguçado possibilita, também, a identificação de uma presa até 3 km de distância, o que faz desse animal o segundo maior predador da Europa.

Os uivos podem estabelecer diferentes tipos de comunicação entre os lobos. Eles ajudam a manter a alcatéia junta em ambientes densos como florestas, possibilitam chamar o bando para um local específico e podem avisar às alcatéias adjacentes que aquela área pertence ao grupo. Além disso, segundo alguns pesquisadores, os uivos ajudam a estabelecer um companheirismo entre os integrantes de uma alcatéia.

A caça é feita em grupo e todos os membros de uma alcatéia participam. Sua estratégia é perseguir a(s) presa(s) até ela(s) se cansar(em), buscando cercá-la(s). Em geral, visam ou o

animal mais velho ou o mais novo ou aquele que esteja ferido ou doente [19]. Dessa forma, o grupo gastará menos energia na perseguição.

Por precisarem de muito alimento por dia (cerca de 5 kg), os lobos procuram animais grandes (e.g. os alces) para que menos investidas sejam necessárias durante o dia. Entretanto, eles também se alimentam de pequenos mamíferos para suprir a necessidade diária de alimento. Uma vez que a caça está abatida, procuram comê-la rapidamente para evitar o roubo ou disputa do alimento por outro animal. A Figura 2 mostra um cerco bem-sucedido realizado pela alcatéia. Cercar o alvo e operar em coordenação de ações para obtenção de objetivos comuns são aspectos centrais a serem incorporados no modelo artificial.

Figura 2. Cerco efetuado pela alcatéia durante a caça.

Todas essas características sociais e comportamentais do lobo revelam o quanto esses animais e sua estrutura social são complexos e o quanto ainda pode-se aprender sobre eles. Seus modelos de comunicação e comportamento social podem servir de modelo para diferentes aplicações, inclusive a análise de seu comportamento de caça e a dinâmica do sistema por meio de modelagem computacional [25].

2.2 Sistemas Multi-Agentes Inteligentes

Sistemas de alta complexidade, por exemplo, de alta dimensionalidade ou não monotônicos, são completamente inviáveis ou ineficientes se programados de maneira convencional. Por convencional, deve-se entender: busca e mapeamento de todas as condições possíveis para determinação de ações necessárias. Dessa forma, qualquer condição não mapeada poderá resultar em uma ação não tomada, inconveniente ou errada.

Técnicas de Inteligência Artificial, em especial Agentes Inteligentes, possibilitam que sistemas sejam construídos e funcionem de maneira robusta, paralela e eficaz. Sistemas multi-agentes permitem que o programador crie uma infra-estrutura computacional capaz de aprender como ela deve operar e tomar decisões oportunas, sem que todas as situações possíveis precisem ser discriminadas. Além disso, seu uso permite que sistemas baseados em agentes sejam modelados e simulados visando (i) analisar a dinâmica de seus comportamentos; (ii) antever o seu funcionamento; (iii) testar sua robustez e eficiência em ambientes hostis; dentre outras possibilidades.

Um agente deve interagir com o ambiente, o qual está inserido, através de seus sensores e atuadores [30]. Os sensores possibilitam sua percepção do ambiente e são utilizados como entrada para a seleção correta da ação desejada. A escolha do atuador pode ser feita através de inferências, deduções, lógicas, cálculos ou de qualquer outro modo. A Figura 3 mostra, esquematicamente, o ciclo da interação do agente com o ambiente.

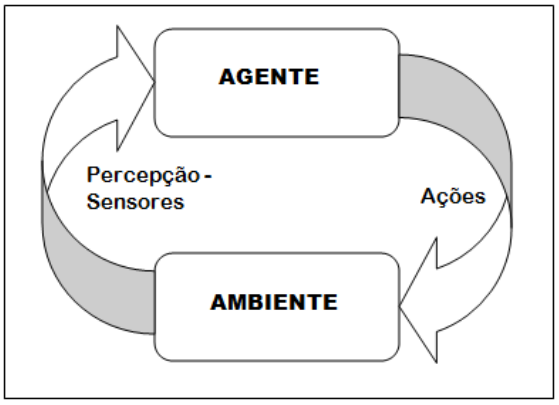

Figura 3. Interação do agente com o ambiente é feita por meio de suas percepções e respectivas ações.

Apesar de concordarem que a autonomia (*i.e.* a não interferência humana para tomadas de decisões) é o cerne de um agente, não existe um consenso sobre a sua conceitualização correta entre os pesquisadores [37] devido as suas diferentes características em variados ambientes e aplicações. Mas, uma boa conceituação foi proposta por Wooldridge e Jennings [40] e adaptada por Weiss [37]: "*Um agente é um sistema computacional que está situado em algum ambiente, e é capaz de ações autônomas nesse ambiente para alcançar seus objetivos projetados.*"

Para afirmar que um agente é inteligente, é necessário que ele seja flexível no sentido de incorporar experiências e autônomo na escolha de suas ações, a fim de atingir seu objetivo

planejado. Para isso, devem-se observar as seguintes características [40]: reatividade, pró-atividade e habilidade social. A pró-atividade ocorre quando o agente toma a ação por iniciativa própria, a partir do pressuposto que ele acredita e baseado em suas percepções. Quando o ambiente é parcialmente observável e a mudança de estado do ambiente é incerta, o agente deve escolher a sua ação baseado nos eventos de mudança (reatividade). Por fim, em sistemas multi-agentes a habilidade social é evidenciada pela sua comunicação tanto de intencionalidade, quanto de desejos, crenças, entre outros. Dessa forma, é possível construir sistemas cooperativos (*i.e.* intencionalidades não conflitantes) ou competitivos (*i.e.* intencionalidades conflitantes).

Alguns agentes podem, ainda, guardar estado e sua seqüência de percepções e ações (ou seja, a sua própria história). Dessa forma, o agente poderá raciocinar (*i.e.* inferir, deduzir, crer) sobre qual ação é a melhor a ser tomada (*i.e.* aquela que maximiza o seu objetivo), dado o estado atual do ambiente, suas percepções atuais e seus registros históricos. Desta forma, dota-se o agente de memória. E é justamente por isso que suas ações tendem a melhorar com o passar do tempo, caso ele utilize as informações passadas para inferir sua ação. Devido a essas características, sistemas baseados em agentes são bastante adaptáveis e robustos ao ambiente aos quais são inseridos.

Uma arquitetura de agentes bem conhecida é a BDI (*Belief – Desire - Intention*), Ela é mais utilizada em agentes orientados a objetivos inseridos em ambientes que mudam constantemente [28]. Essas informações ajudam o agente a deliberar melhor sobre as suas possíveis ações.

As intenções de um agente são as ações que ele pode tomar no ambiente atual, baseado nas suas crenças e desejos, visando aumentar sua medida de desempenho (*i.e.* medida de sucesso do agente para atingir seus objetivos). Ela deve ser reavaliada ocasionalmente, pois as mudanças no ambiente podem resultar na mudança da intenção [37]. Entretanto, essas re-análises devem ser bem projetadas e planejadas, pois custam tempo e recursos computacionais (por isso, só são realizadas em ambientes dinâmicos). Segundo Kinny e Georgeff, a reavaliação só é vantajosa se a taxa de mudança do ambiente for alta [22], principalmente se for considerado um sistema multi-agentes.

Em sistemas multi-agentes há ainda a preocupação com os recursos computacionais envolvidos com a comunicação e interação entre os agentes. A comunicação é feita por meio

de protocolos bem-definidos que podem informar aos outros agentes as suas intencionalidades e crenças, definir um plano de ação, entre outros. A interação entre os agentes pode ser cooperativa (*i.e.* intencionalidades ou objetivos não conflitantes entre os agentes) ou competitiva (*i.e.* intencionalidades ou objetivos conflitantes entre os agentes).

Alguns sistemas cooperativos possuem um objetivo comum, o qual deverá ser perseguido por todos os agentes do ambiente. Dessa forma, cada agente deverá analisar, também, se sua ação ajudará ao sistema atingir o objetivo planejado, além do seu próprio. Não obstante, é necessário um controle adicional para determinar se, mesmo diminuindo a sua medida de desempenho; então, um agente deverá escolher uma ação que vise atingir o objetivo comum. Esse conceito será utilizado de forma seminal no modelo apresentado neste trabalho, onde cada lobo deve conciliar ganhos individuais e sociais.

Em ambientes com essas características, também chamados de Sociedade de Agentes, a comunicação entre os agentes geralmente é intensa e onera bastante os recursos computacionais. Dessa forma, alguns pesquisadores estão buscando uma forma de diminuir a comunicação, sem perder as vantagens resultantes da interação e comunicação entre os agentes. Verbeeck, por exemplo, propôs a utilização aprendizado de agentes por reforço para alcançar o Equilíbrio de Nash ótimo segundo a distribuição de Pareto [36].

Por ser de fácil implementação e de baixo custo computacional, sistemas multi-agentes são bastante utilizados atualmente. Um gerenciador de servidor de email foi proposto baseado em agentes inteligentes hibridizado com Redes Neurais [4] através da arquitetura cliente-servidor [33]. Seu principal objetivo é otimizar a realização de atividades de gerenciamento da memória do servidor, como a exclusão de *emails* antigos, além de torná-lo adaptável às necessidades da organização.

Mais recentemente, foi desenvolvido um sistema multi-agente para avaliar a importância dos hospitais e explicar o impacto da educação na prevenção da doença da malária na população do Haiti [33]. O sistema contempla a infra-estrutura da região (estradas, hospitais, escolas), os agentes (*i.e.* seres humanos) que podem estar com a doença ou não, além de poderem se locomover no ambiente, se reproduzir, morrer e se infectarem e as interações entre eles. No modelo proposto por este trabalho, agentes de forma cooperativa vão lidar com estrutura da floresta e a movimentação das ovelhas (vide o Capítulo 3).

2.3 Computação Social

Alguns estudiosos apontam a cooperação como o próximo passo do futuro da humanidade. Esse pensamento tem sido comprovado com a enorme aceitação de jogos eletrônicos e aplicações *web*, os quais priorizam a colaboração entre os indivíduos para atingirem um mesmo objetivo ou são dedicadas ao compartilhamento de informações e criação de grandes comunidades com interesses em comum.

Entretanto, a cooperação não é uma particularidade do ser humano. Diversas populações animais também vivem e usufruem da vida em grupos sociais. As formigas, por exemplo, possuem uma estrutura social hierárquica bem definida, na qual todas participam e colaboram com as atividades da colônia [31].

As principais vantagens da cooperação são o fortalecimento do grupo e o conhecimento que emerge da interação entres seus membros. As formigas estariam completamente vulneráveis ao ataque de predadores e teriam dificuldades de coletar alimentos, caso não vivessem em grupo. Além disso, a comunicação entre elas permite a coordenação de suas atividades e o surgimento de uma inteligência coletiva, o que possibilita que elas percorram o menor caminho do formigueiro para a sua comida.

Utilizando o comportamento de populações sociais como uma metáfora, a Computação Social é uma subárea da Computação Natural a qual se propõe a:

- simular a dinâmica de sistemas reais e analisar aspectos específicos da interação entre os agentes que compõem o ambiente. Logo, é possível prever ou projetar o impacto sobre o grupo que determinadas modificações no ambiente podem ocasionar; e,

- resolver problemas complexos com um menor custo computacional e de forma mais intuitiva [2].

A expressão ‘populações sociais’ é utilizada aqui para se referir a um conjunto de agentes que podem interagir entre si. Qualquer conjunto de agentes com comportamentos coletivos (*i.e.* uma multidão, uma manada, uma alcatéia, etc.) são considerados populações sociais. Entretanto, não é obrigatória a movimentação desses agentes pelo ambiente, basta haver uma comunicação e interação entre eles.

A modelagem e implementação desse paradigma de computação são feitas utilizando-se a técnica de Sistemas Multi-Agentes. Entretanto, várias aplicações incorporam algum mecanismo de Computação Evolucionária [16] para aperfeiçoar a adaptabilidade e mutação dos agentes dentro da população. Assim, é possível reproduzir melhor a dinâmica natural de sistemas reais no ambiente artificial.

Várias aplicações e pesquisas estão sendo desenvolvidas nessa área. Algumas em simulações de comportamentos e características sociais de uma determinada população, com a finalidade de analisar aspectos específicos ou prever como alguns comportamentos ou influências no ambiente resultarão em boas ou más mudanças. Enquanto outras, com a finalidade de obter um resultado mais eficiente ou menos oneroso computacionalmente, buscam inspiração na natureza para atingirem seus objetivos.

Por exemplo, recentemente foi desenvolvida uma aplicação de Computação Social que modela e simula a dinâmica da infra-estrutura urbana da cidade de Bogotá com o aumento das famílias e das unidades domiciliares, inclusive das negociações do mercado imobiliário [12].

Trabalhos recentes visam simular as mudanças nos comportamentos de agentes sociais relacionados com a disseminação e contaminação de doenças [28] e a influência de elementos estruturantes (*e.g.* casas, rodovias, hospitais, locais de trabalho) no comportamento desses agentes, para possibilitar o entendimento e controle de epidemias [27]. O avanço dessas pesquisas poderá resultar em ferramentas que auxiliem as autoridades definirem políticas públicas em saúde e em outros setores da sociedade.

Já o FSS (*Fish School Search*) [1] é uma aplicação de Computação Social que buscou inspiração no cardume para desenvolver algoritmos inteligentes de busca. Seus resultados foram bastante satisfatórios para espaços hiper-dimensionais.

2.4 Aproximação de Funções

A aproximação de funções é uma classe de problemas bastante importante quando os problemas do mundo real possuem leis de formação – as funções – complexas e caras de serem realizadas, em termos computacionais. Ou ainda quando se deseja saber a função que melhor aproxima uma dada nuvem de pontos que descrevem alguma realidade. Dessa forma,

pode-se de forma compacta prever o comportamento de determinadas variantes em qualquer instante do futuro.

Em diversas áreas do conhecimento existe a necessidade de aproximar funções. Em Saúde, por exemplo, funções podem representar a epidemiologia de uma doença infecciosa; o crescimento de uma colônia de bactérias; modelos para a dinâmica do sistema cardiovascular e respiratórios dos animais; crescimento de tumores etc. Em ciências humanas, funções podem representar variações monetárias; crescimento de uma empresa; aceitação, demanda e oferta de um produto; consumo da população; a receita, o lucro e o custo de uma fábrica; censos demográficos. Em ciências exatas, elas são largamente utilizadas. Por exemplo, projeto de filtros digitais [10].

Os métodos convencionais de aproximação de funções como aproximações polinomiais, aproximações trigonométricas e *splines* podem ser inviáveis ou demasiadamente complexas de serem implementadas computacionalmente. Alguns dos métodos computacionais existentes para essa finalidade são a Programação Linear (através do algoritmo SIMPLEX) [10] e Redes Neurais Artificiais (RNA). Entretanto, ambas possuem um alto custo computacional. Argumenta-se que o WPA pode ser uma melhor solução para aproximar funções.

Os métodos que utilizam o SIMPLEX possuem, no pior caso, uma complexidade exponencial $O(2^n)$. Isso é ruim, pois há um grande consumo dos recursos computacionais existentes, além de possuírem um alto tempo de execução que escala exponencialmente com o tamanho da entrada do problema de aproximação.

Por outro lado, as técnicas que utilizam RNA possuem um alto custo de processamento (devido às diversas fases do treinamento), algumas vezes tempo de execução elevado, alto consumo de memória além da possibilidade de acontecer *overfitting* e a função não ser aproximada corretamente. Recentemente, surgiram novas abordagens que utilizam Algoritmos Genéticos para determinar a arquitetura de uma RNA de alta ordem, minimizando a possibilidade da rede não conseguir aproximar a função [29]. Contudo, com a inserção de um novo passo no processo de aproximação, o custo computacional aumenta.

Capítulo 3

Modelagem e Implementação

A modelagem foi realizada a partir dos estudos sobre o comportamento de uma alcatéia na natureza e suas características relevantes para atingir os objetivos do projeto. Foram considerados, ainda, aspectos relativos aos sistemas multi-agentes inteligentes, Computação Social e padrões de projeto orientados a objeto.

O projeto foi implementado na linguagem JAVA [9], devido a sua portabilidade, usabilidade e facilidade de encontrar componentes e bibliotecas úteis para esse projeto, tais como ferramentas de desenho de gráficos, leitura e escrita de arquivos e construção de diagramas de UML [3]. Ademais, como este projeto será muito provavelmente estendido, a opção por JAVA facilitará muito esse trabalho.

Este capítulo descreve, em detalhes, cada um dos aspectos considerados na modelagem e na implementação do módulo um do WPA. As dificuldades encontradas e sugestões de trabalhos futuros serão descritas no capítulo 5.

3.1 Modelagem

Vários aspectos foram considerados com a finalidade de prover uma modelagem coerente e eficaz para a resolução do problema. Desde a análise do comportamento das alcatéias na natureza, até os aspectos relativos à Computação Social. Nesse projeto foram considerados problemas com apenas duas dimensões, até mesmo por se tratar apenas de uma prova de conceito. A seguir está como a modelagem foi realizada.

3.1.1 Aspectos do Comportamento de uma Alcatéia na Natureza

Como visto no Capítulo 2, os lobos agem em grupo ou clãs durante a caça a fim de atingirem um objetivo em comum. A estratégia é cansar a presa, cercá-la e, por fim, capturá-la. Esse comportamento cooperativo e com objetivos em comum inspirou em parte o presente trabalho

no intuito de construir uma ferramenta que modele seus comportamentos mais relevantes *no contexto do WPA*.

Imaginou-se que, na alcatéia quando em busca de suas presas (as ovelhas), cada lobo possui sua área de atuação própria e o clã fica distribuído por sobre uma frente (*i.e.* intervalo no eixo das abscissas) de forma que cerque as ovelhas e as canse. Às ovelhas, portanto, resta apenas a fuga para não se tornarem alimento. Durante esse processo imaginado de caça, além dos deslocamentos dos lobos na direção da presa, há o ajuste de sua área de atuação, de forma que o torne mais eficiente (ou seja, evite deslocamentos desnecessários) e se adapte à nova situação atual da investida. As ovelhas, por sua vez, buscarão fugir dos lobos em direção às árvores da floresta (local mais seguro para essas). Note-se que é esperado, portanto, uma convergência de ovelhas para a nuvem de pontos de entrada (*i.e.* as árvores) e isso auxiliará fundamentalmente a tarefa no futuro de aproximar funções – objetivo do WPA.

Apesar de encontrar-se fora do escopo deste projeto, a comunicação entre os lobos durante a caça, informando suas intencionalidades e informando suas medidas de desempenho serão fundamentais para um melhor aproveitamento dessa abordagem, assim como acontece no mundo real. Entretanto, a convergência acima mencionada acontece mesmo sem haver protocolo explícito de comunicação entre os lobos já que, como será apresentado a seguir, existem informações globais implícitas (*i.e.* indicadores sociais da matilha) que são consideradas nas ações dos lobos.

Dessa forma, os elementos considerados para o projeto foram as árvores (a floresta), as ovelhas (as presas) e os lobos (alcatéia). Os lobos, descrição detalhada na próxima seção, são agentes inteligentes que neste ponto do desenvolvimento do WPA possuem dois operadores básicos: (i) o de deslocamento e (ii) o de ajuste do raio de atuação do lobo. O deslocamento é utilizado para mover o lobo em direção às ovelhas, enquanto o de ajuste do raio aumenta a área de atuação do lobo, buscando aumentar o seu desempenho. Apesar de atuarem sobre o mesmo objetivo, perseguirem as ovelhas, esses dois operadores não foram combinados já que podem cumprir funções ligeiramente diferentes, incluindo economia de recursos computacionais já que, por exemplo, um lobo sozinho sem se mover – apenas aumentando seu raio de atuação, pode controlar (*i.e.* forçar o movimento) de várias ovelhas.

As ovelhas em nossa abordagem são agentes reativos, que agem em conformidade com a operação do(s) lobo(s) ao(s) qual(ais) estão sob influência; os lobos apenas têm

influência sobre as ovelhas que estão dentro de seu raio de atuação. Assume-se que as ovelhas se deslocarão sempre em direção à floresta.

Por fim, a floresta representa o local em que as ovelhas "estão a salvo do ataque dos lobos". Entretanto, nenhuma ovelha deverá atravessar a floresta, tendo em vista que voltará a ser um alvo fácil dos lobos. No contexto do WPA, a floresta representa a nuvem de pontos a qual se deseja encontrar uma função aproximada.

Esses elementos estão organizados no ambiente da seguinte forma: há dois grupos de lobos envelopando um acima e outro abaixo as árvores da floresta; dois grupos de ovelhas, um acima e outro abaixo da floresta e ambos dentro do "envelopamento" dos lobos; e uma floresta (*i.e.* a nuvem de pontos). Os lobos que estão acima da floresta caçarão as ovelhas que estão nessa mesma região, de forma que as ovelhas fujam em direção às árvores. Da mesma forma ocorre para os lobos e ovelhas que estão abaixo da floresta. Entretanto, para que esse ambiente funcione, é necessário que as ovelhas estejam sempre entre os lobos e a floresta. A Figura 4 apresenta de forma ilustrativa o ambiente e os elementos que compõem o WPA.

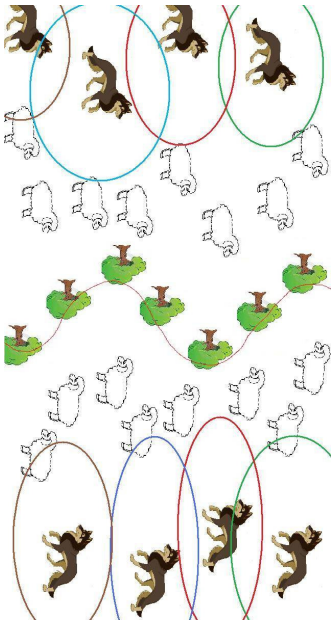

Figura 4. Elementos do ambiente modelado: lobos e seus respectivos raios de atuação, ovelhas e as árvores (floresta).

O algoritmo foi concebido para que, ao final de sua execução, as ovelhas estejam o mais próximo possível da floresta (*i.e.* a nuvem de pontos que representa uma função) e os lobos estejam posicionados de forma que seus raios de atuação tangenciem a floresta.

Com isso, espera-se que o futuro módulo dois do WPA consiga aproximar a função baseando-se apenas no conjunto de tuplas geradas pelo módulo 1 (atual). Ou seja, o par posicionamento e raio final de cada lobo.

3.1.2 Aspectos de Sistemas Multi-Agentes Inteligentes

O ambiente de tarefas considerado e descrito anteriormente pode ser classificado da seguinte forma, sob os aspectos das propriedades dos sistemas multi-agentes sugeridos por Russel [30]:

- **Parcialmente observável**: os lobos não possuem percepção completa do ambiente;
- **Determinístico**: a partir do estado atual e da ação do lobo, poderá ser determinado o próximo estado;
- **Seqüencial**: as decisões atuais podem afetar as decisões futuras do lobo;
- **Estático**: o ambiente não muda enquanto o lobo está tomando a sua decisão;
- **Discreto**: o conjunto das ações e percepções do lobo é discreto; e
- **Multi-agente cooperativo**: os lobos buscam atingir seus objetivos individuais não conflitantes. Apesar de no futuro já se conceber algum tipo de litígio entre lobos por território – quando haverá, provavelmente, a necessidade de mais um operador: o de recrutamento.

A seguir, descreve-se em detalhe a composição de cada agente do sistema e seus comportamentos e ações.

Lobo

Os lobos são agentes que possuem uma percepção limitada do ambiente que estão inseridos, pois só têm conhecimento daquele compreendido dentro de seu raio de atuação (sensor). É baseado nessa percepção que o lobo escolherá qual dos dois operadores será executado, no momento: deslocamento ou aumento do raio de atuação (*i.e.* seus atuadores numa perspectiva de modelagem de agentes).

Os lobos são agentes baseados em objetivos, pois além da percepção e do seu estado atual, eles também precisam de objetivos que os auxiliem na escolha da ação apropriada.

Eles possuem dois objetivos que os guiarão nas suas ações: diminuir o erro médio individual (e, conseqüentemente o global) e diminuir o seu desvio-padrão. A forma de escolha do operador apropriado é descrita na seção 3.2.1.

O operador de movimentação desloca o lobo na ordenada do plano cartesiano em unidades de um valor previamente definido. O operador de ajuste da área de atuação aumenta ou diminui o raio do lobo de outro valor escalar previamente definido. Entretanto, caso um desses operadores seja escolhido e o lobo não possa se movimentar ou ajustar o seu raio conforme as magnitudes calculadas, o novo valor de ajuste será igual a 90% da menor distância entre a ovelha e a árvore verticalmente mais próxima.

Um dos motivos que pode impedir o ajuste do raio ou do posicionamento do lobo de acordo com o valor pré-definido é quando, após a utilização do operador, alguma das ovelhas ultrapassa a floresta, situação atualmente proibida dentro do ambiente.

Conceitualmente, cada lobo pode possuir uma função de vizinhança associada própria, a qual será utilizada para compor o cálculo da função global aproximada pelo WPA, esta fora do escopo desse módulo. Entretanto, para facilitar a implementação deste e do módulo seguinte, assume-se a função Gaussiana como a única função de vizinhança, definida como padrão para todos os lobos.

Ovelhas

As ovelhas são agentes reativos simples. Suas ações limitam-se a se deslocar (por enquanto no eixo das ordenadas) na direção e sentido da floresta, toda vez que estiver dentro do raio de atuação do lobo. A ovelha sempre se deslocará para o limite do raio de atuação do lobo, tendo em vista que é o lugar mais perto e seguro que ela poderá ir a cada momento.

A cada movimentação do lobo, as ovelhas verificam se estão dentro da área de atuação do predador e, assim, conseguem fugir da caça.

Quanto mais próximas as ovelhas estiverem das árvores, menor será o erro global e melhor será o desempenho do sistema. O papel da ovelha no módulo um do WPA é

fundamental no sentido de auxiliar cada lobo a ajustar seu posicionamento e seu raio final, os quais serão utilizados para calcular a função aproximada (módulo dois do WPA).

Árvores

As árvores não são agentes dentro do sistema, são apenas elementos que representam de forma constante a nuvem de pontos de entrada.

3.1.3 Aspectos de Computação Social

Este trabalho buscou inspiração no comportamento da alcatéia na natureza durante a caça para ajudar na resolução do problema de aproximação de funções. O comportamento de uma alcatéia foi modelado de acordo com as necessidades específicas do problema, buscando reproduzir um ambiente cooperativo entre os lobos. Desta forma ao buscarem atingir seus objetivos individuais, os lobos cooperam para a diminuição do erro global. Contudo, como a comunicação entre os agentes está fora do escopo deste trabalho, essa colaboração ainda está limitada.

3.2 Implementação

A implementação foi realizada na linguagem JAVA, por motivos anteriormente citados. Porém, a construção de alguns requisitos do sistema, como a construção do gráfico que representa o ambiente, exigiu a utilização de outras ferramentas e aplicativos. Bibliotecas Java de desenho de funções e de suporte a construção de UML foram utilizadas para minimizar o esforço de construção e oferecer recursos mais avançados. Todo o projeto foi desenvolvido na plataforma *Eclipse Ganymede* [11].

A seguir serão descritos como os princípios de funcionamento do ambiente descritos na seção anterior foram implementados; serão apresentados alguns dos principais métodos que compõem o ambiente e será discutido cada aspecto da implementação deste trabalho.

3.2.1 Princípios de Funcionamento do Algoritmo

A implementação do algoritmo foi realizada de acordo com a modelagem anteriormente descrita. Nessa seção, apresenta-se em detalhes a forma como cada aspecto foi construído, ou

seja, os meios utilizados para atender a modelagem realizada. A Figura 5 ilustra como é caracterizado computacionalmente o ambiente de execução do WPA.

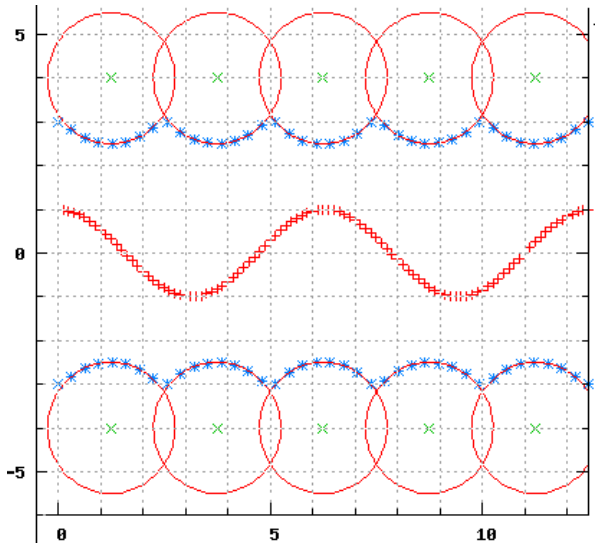

Figura 5. Ambiente modelado computacionalmente: ovelhas (*), lobos (x) e seus respectivos raios e árvores (+).

Como pode ser observado, o número de lobos e ovelhas iniciais, acima e abaixo da floresta, foi definido ser sempre igual. Metade das ovelhas e lobos estará acima das árvores e o restante abaixo. Além disso, cada ovelha e cada lobo que estiverem abaixo da floresta, sempre terão um correspondente na mesma abscissa acima da floresta.

A dinâmica do ambiente consiste na movimentação dos lobos e no ajuste de suas respectivas áreas de atuação em direção às ovelhas. As ovelhas, por sua vez, reagirão a esse "ataque" dos lobos fugindo para a posição limite da área de atuação dos lobos, pois, dessa forma, elas estarão momentaneamente seguras. Ressalta-se, entretanto, que os lobos não poderão ultrapassar uma ovelha nem as ovelhas poderão ultrapassar a floresta.

A ação do lobo é definida de acordo com seu erro médio e o seu desvio-padrão. O erro médio (E) do lobo é a média aritmética da soma das distâncias (d) das (n) ovelhas (sob sua coerção) para as árvores verticalmente mais próximas. O seu desvio-padrão (σ_L) é calculado baseado na média do erro do lobo e nas distâncias de cada ovelha para a árvore verticalmente mais próxima. As equações (1) e (2) mostram como o erro médio do lobo e o seu desvio-padrão são calculados, respectivamente.

$$E = \frac{1}{n}\sum_{i=1}^{n} d_{O_i,F} \quad (1)$$

$$\sigma_L = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_{O_i,F} - E)^2} \quad (2)$$

A distância ($d_{O,F}$) entre uma ovelha e a floresta (F) é igual a distância euclidiana ($d_{O,A}$) obtida entre a ovelha e a árvore mais próxima de sua abscissa, de acordo com (3).

$$d_{O,F} \in \{d_{O,A} \mid A \in F \text{ and } \forall A' : F, |A_x - O_x| \leq |A'_x - O_x|\} \quad (3)$$

Caso o erro médio do lobo esteja alto (maior do que um nível pré-estabelecido) e o desvio-padrão baixo (menor do que um nível também pré-estabelecido), o seu raio de atuação é aumentado, visando diminuir o seu erro médio. Dessa forma, as ovelhas tenderão a se aproximar mais das árvores e o erro médio do lobo tenderá a diminuir.

Se o erro médio do lobo estiver baixo e o desvio-padrão alto, o lobo irá se deslocar em direção à floresta. Assim, as ovelhas estarão mais próximas das árvores e o desvio-padrão do lobo tenderá a diminuir, assim como seu erro médio.

De forma a facilitar o entendimento, a Figura 6 mostra as ovelhas e as respectivas árvores mais próximas de suas abcissas, cujas ditâncias entre esses elementos do ambiente representam a distância das ovelhas para a floresta. Além disso, são apresentadas duas situações que ilustram a relação entre o erro médio individual e o desvio-padrão do lobo na escolha dos operadores adequados.

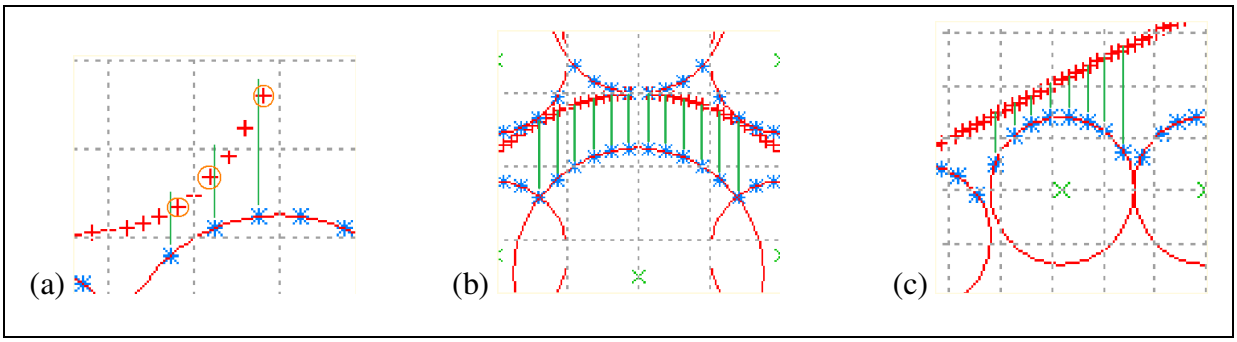

Figura 6. (a) Árvores mais próximas da abcissa das ovelhas; (b) Situação de ajuste de raio (baixo desvio-padrão e alto erro médio); (c) Situação de ajuste de posicionamento (alto desvio-padrão e alto erro médio).

A cada iteração, um lobo é selecionado e verifica qual a ação tomará baseado nos critérios acima definidos. As ovelhas compreendidas dentro de seu raio de atuação reagem movimentando-se em direção à floresta (quando necessário). Quando todos os lobos tiverem sido selecionados, volta-se a escolher o lobo inicial (como numa lista circular) – nesse momento, diz-se que foi completada uma época. Esse mecanismo repete-se até que seja atingido algum critério de parada.

Existem três possíveis critérios de parada: o número de épocas, o erro global e o número de lobos no ambiente. Esses critérios podem ser utilizados simultaneamente. Assim, quando o primeiro critério for satisfeito, a execução para e o resultado será aquele obtido na última iteração.

Durante a execução, são calculados dois erros que ajudam a verificar e analisar a corretude e eficácia do algoritmo desenvolvido. O erro médio global de cada época e o erro médio global de cada iteração. Estes, indicadores de sucesso ou fracasso da "caça".

Ao final de toda a interação entre os elementos do ambiente, quando os lobos não agirem mais ou for atingido algum dos critérios de parada do algoritmo, pode-se afirmar que os lobos atingiram seus posicionamentos e raios finais. O conjunto dessas tupla, referentes a todos os lobos, corresponde ao objetivo do módulo um do WPA e ao conhecimento emergente da ação individual de cada lobo. A Figura 7 mostra o fluxograma da execução deste módulo.

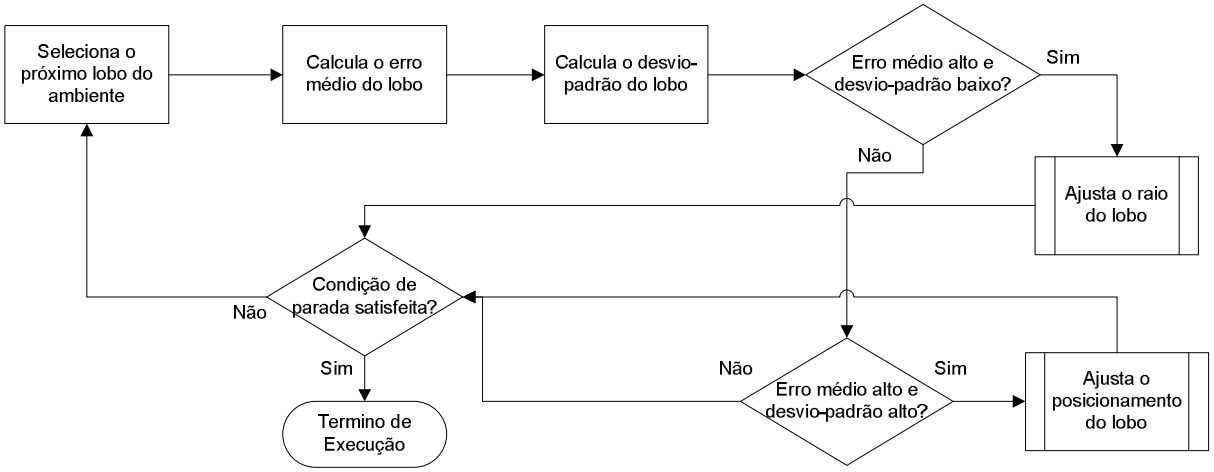

Figura 7. Fluxograma da execução do módulo um do WPA.

Para ajustar o raio do lobo, são selecionadas todas as ovelhas que podem ser afetadas com o uso desse operador. Essas ovelhas são as que estão compreendidas dentro ou na

fronteira da região definida pelo novo raio do lobo (raio atual do lobo acrescido do valor de ajuste padrão *p*). Após isso, é verificado se cada ovelha que será afetada pelo ajuste do raio do lobo poderá se movimentar do mesmo valor *p*, sem que ultrapasse a floresta.

Caso alguma ovelha não possa se movimentar desse valor, é definido o novo valor de ajuste do raio. Esse novo ajuste será igual a 90% da menor distância encontrada da ovelha para a árvore verticalmente mais próxima. Caso nenhuma ovelha ultrapasse a floresta, o lobo ajusta seu raio do valor *p* pré-definido.

Em seguida, todas as ovelhas afetadas por esse ajuste do lobo (*i.e.* encontram-se dentro de seu raio de atuação) serão movimentadas até a extremidade de sua área de atuação. A Figura 8 mostra o fluxograma do ajuste do raio do lobo.

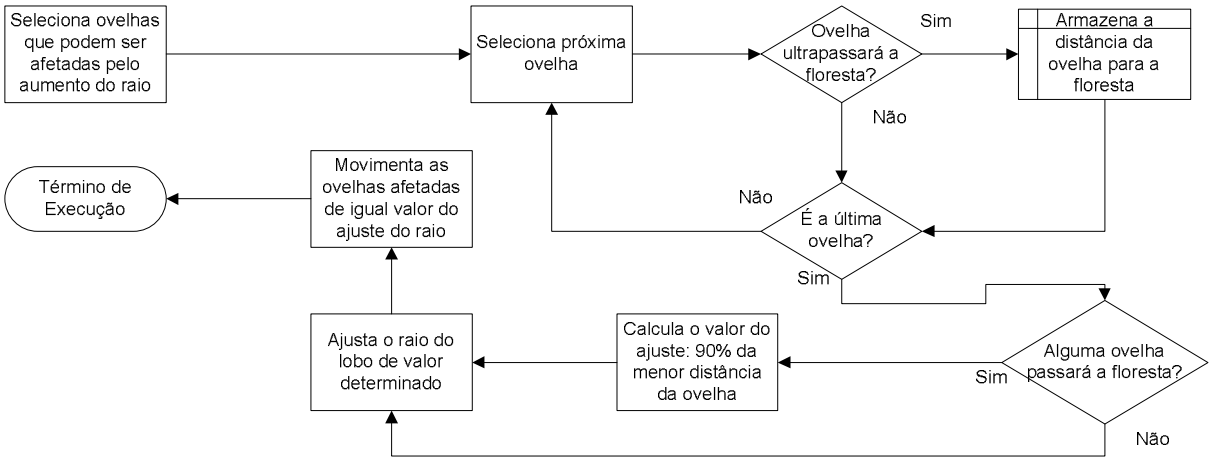

Figura 8. Fluxograma do ajuste do raio do lobo.

Por fim, o ajuste de posicionamento do lobo é bastante semelhante ao do operador de ajuste do raio. A única diferença é que o lobo movimenta-se em direção à floresta, ao invés de aumentar a sua área de atuação. A Figura 9 mostra o fluxograma do ajuste do posicionamento dos lobos. Se comparado com o fluxograma anterior, notam-se, facilmente, as semelhanças e diferenças entre ambos.

O Apêndice A mostra um passo-a-passo da execução do algoritmo para um exemplo com a função tangente hiperbólica.

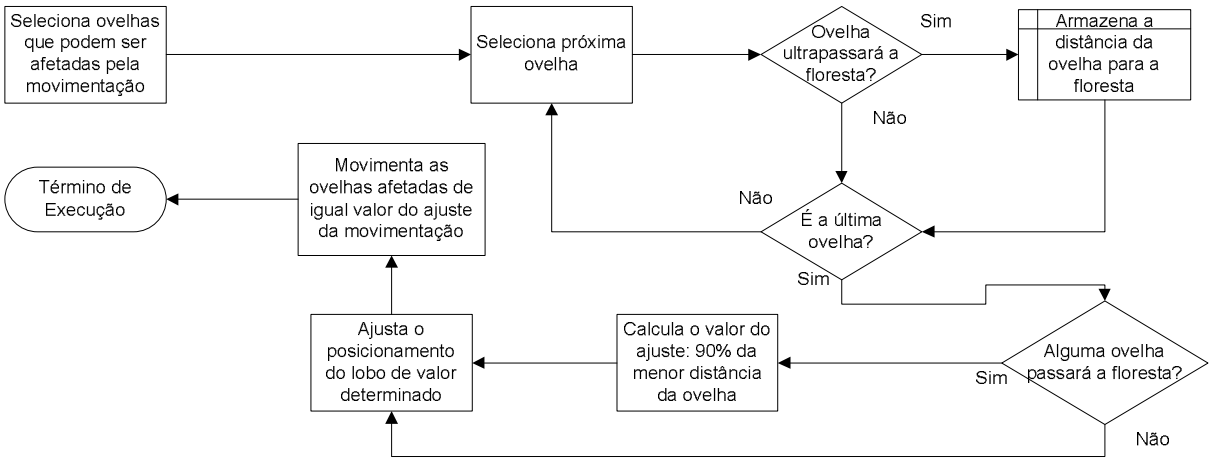

Figura 9. Fluxograma do ajuste de posicionamento do lobo.

3.2.2 Descrição das Classes que Compõem o Sistema

O projeto é composto de 15 classes divididas em quatro pacotes: *wpa.entity* (agrupa as entidades básicas do projeto); *wpa.function* (agrupa as classes referentes à função associada ao lobo); *wpa.main* (possui uma classe que é responsável por executar o projeto); e, *wpa.util* (agrupa classes auxiliares e utilitários do sistema). Para uma maior compreensão da estruturação e relacionamento das classes do sistema, recomenda-se observar o diagrama de classes encontrado no Apêndice B.

Pacote *wpa.entity*

Contém os elementos que compõem o ambiente (*Wolf, Sheep, Tree, Forest)*, o mundo do WPA (*World*) e o próprio ambiente (*Enviroment*). Essas são as classes principais do sistema, as quais modelam todo o comportamento e interação no ambiente.

O lobo, modelado pela classe *Wolf*, possui como atributos a sua posição atual no mundo, o seu raio de atuação atual e a sua função agregada (utilizada pelo módulo dois do WPA). Seus principais comportamentos são ajustar o seu raio e seu posicionamento, verificar se uma ovelha está sob seu controle e retornar o valor de sua função agregada para um dado parâmetro.

A ovelha (*Sheep*) e a árvore (*Tree*) possuem características iguais, porém comportamentos diferentes. Ambas possuem apenas um atributo: o seu posicionamento atual

no mundo WPA. Porém, enquanto a árvore não possui nenhum comportamento (a não ser retornar o valor da sua posição), a ovelha pode ajustar o seu posicionamento.

A floresta (*Forest*) é simplesmente uma coleção de árvores que, além dos métodos específicos de manutenção da coleção (inclusão, exclusão e alteração), possui alguns métodos que auxiliam na construção gráfica do ambiente, como retornar o ponto mínimo e máximo que compreende a floresta e seu respectivo tamanho.

O mundo (*World*) é a entidade que integra todos os elementos que compõem o ambiente de execução e simulação. Ela contém a floresta, a coleção de lobos que se encontra acima e outra abaixo da floresta, a coleção de ovelhas acima e abaixo da floresta e os limites do mundo. Ela implementa todos os mecanismos necessários para que haja a interação entre os elementos do ambiente (cálculo do erro e desvio-padrão individual do lobo, cálculo do erro global, cálculo da mínima distância das ovelhas para as árvores, entre outros), além de ser responsável pela inicialização do mundo.

Por fim, o ambiente (*Environment*) é a entidade responsável por controlar o comportamento dos elementos no mundo e armazenar os erros de cada iteração e de cada época. Possui como atributos o mundo a ser controlado, as coleções que armazenam os erros e as condições de parada. Sua principal funcionalidade é possibilitar a interação entre os elementos e prover os mecanismos de ajuste de raio e de posicionamento do lobo, assim como movimentar a ovelha quando necessário. A Figura 7, a Figura 8, e a Figura 9 mostram o fluxograma do funcionamento desses mecanismos.

Pacote *wpa.function*

Este pacote contém classes responsáveis por representar a função agregada do lobo, a qual somente será utilizada no módulo dois do WPA. Contudo, sua modelagem e implementação inicial foi realizada neste projeto visando otimizar o seu desenvolvimento posterior. Nela encontram-se apenas uma interface (*Function*) e uma classe que a implementa (*GaussianFunction*).

A interface contém apenas um método a ser implementado: o valor de retorno da função, dado os parâmetros necessários. A classe *GaussianFunction*, a qual implementa a

interface, simplesmente retorna o valor da função para um determinado ponto, dado seu centro e raio.

Pacote *wpa.util*

Este pacote contém as classes utilitárias da aplicação, as quais são usadas para desenhar os gráficos do ambiente e dos erros (*PlotGraphics* e *Charts*, respectivamente), para ler e gerar a nuvem de pontos (*Input* e *FunctionGenerator*, respectivamente), para configurar os parâmetros da execução (*Constants*) e para modelar o ponto que representa o posicionamento atual dos elementos no mundo (*Point*). As classes de desenhos dos gráficos utilizam as ferramentas *JFreeChart* e *GNUPlot* e seu funcionamento e utilização são descritos nas seções 3.2.5 e 3.2.7, respectivamente.

A classe *FunctionGenerator* é responsável por gerar uma nuvem de pontos para uma dada função, um intervalo de amostragem e um número de amostras. Enquanto a leitura dos pontos que representam a função a qual se deseja aproximar é feita através da classe *Input*. As informações são lidas do arquivo e transformadas em uma coleção de pontos (*Point*), os quais serão atribuídos às árvores que irão compor a floresta. A seção 3.2.4 descreve outros detalhes desse funcionamento.

Todos os parâmetros e constantes do sistema são declarados e definidos na classe *Constants*. Assim, é possível alterar o funcionamento de vários aspectos da aplicação em somente um lugar, de forma centralizada, sem que seja necessário saber em que local exatamente do código são utilizados. Por exemplo, nela pode-se definir o valor para cada critério de parada, o valor de comparação para o erro médio e desvio padrão do lobo que definirá se estão altos ou baixos, o valor padrão do ajuste de raio e posicionamento do lobo, o percentual de lobos e ovelhas no ambiente, o número de dimensões do ambiente, entre outros. A definição desses parâmetros e constantes citados influencia diretamente no resultado final do algoritmo e no seu tempo de convergência.

Pacote wpa.main

Contém apenas a classe *Main*, a que inicia toda execução da aplicação. Seu funcionamento é descrito pelo fluxograma da Figura 10.

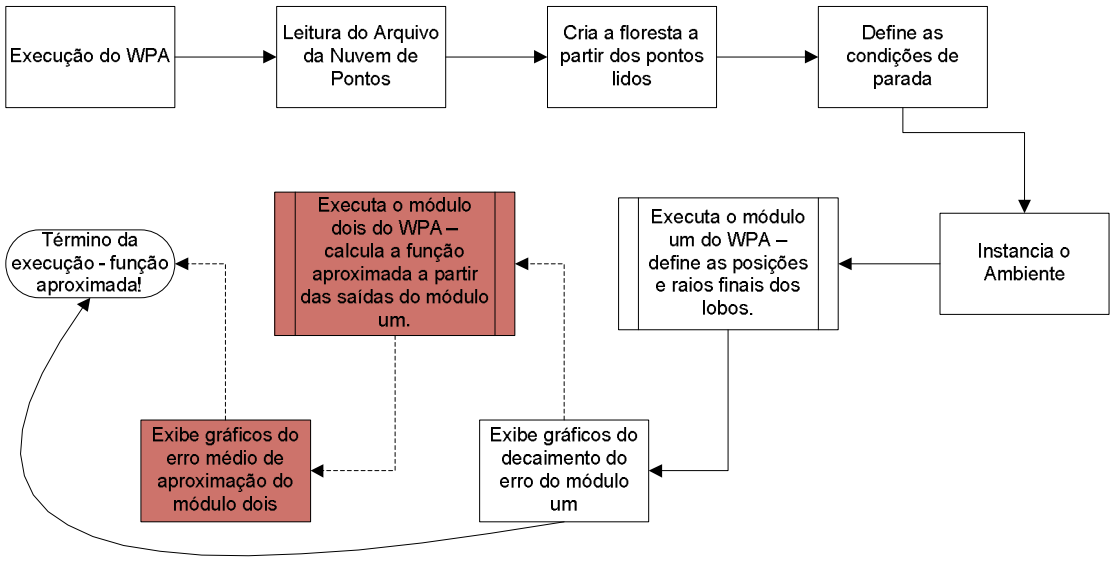

Figura 10. Fluxograma do funcionamento do algoritmo do WPA. Processos em vermelho e setas tracejadas indicam que ainda não foram implementadas (referem-se ao módulo dois).

3.2.3 Padrões de Implementação, Análise e Projetos Orientados a Objetos com JAVA

O trabalho foi construído seguindo padrões de codificação definidos pela *Sun Microsystems* [35], com forte ênfase na documentação e na nomenclatura dos métodos, atributos e variáveis. Todas as constantes e atributos de classe, assim como os métodos, possuem comentários *Javadoc* que explicam a sua razão de ser e como devem ser utilizados. Outros comentários foram inseridos para explicar, em detalhes, o funcionamento e os objetivos de trechos do código mais complexos. Toda codificação foi extremamente documentada para auxiliar e facilitar a manutenabilidade do sistema.

Além da recomendação da *Sun* para nomeação dos atributos, variáveis, constantes e parâmetros, foi utilizado um padrão particular que o complementa. Todos os atributos de classe começam com o prefixo ‘a-’ (e.g. *aRadius*), enquanto os parâmetros começam o prefixo ‘p-’ (e.g. *pNewRadius*). Os nomes de classes, atributos, variáveis, constantes e métodos são todos em inglês. E, todas as constantes são declaradas em uma classe específica para facilitar a configuração de execução do sistema.

Desde o começo do desenvolvimento houve uma grande preocupação e cuidado com a passagem de parâmetros e cópias dos objetos. Na linguagem JAVA, não existe passagem de

valor e sim passagem por referência. Como existem várias operações com listas e modificações nos objetos que as compõem, foi necessária bastante atenção com as operações implementadas para que não ocorressem erros semânticos e a lógica de controle do sistema fugisse do controle. Sabe-se que erros dessa natureza são bastante difíceis de serem identificados, por isso houve grande preocupação durante a construção do projeto.

Aspectos inerentes à Análise e Projeto Orientados à Objetos [14] também foram considerados durante o planejamento e construção da aplicação. Dentre eles, a modularidade, visibilidade dos dados e dos métodos pertencentes a cada objeto, manutenabilidade, coesão alta, generalizações, agregações e o nível de acoplamento baixo foram avaliados em todas as etapas do desenvolvimento. A finalidade disso foi a construção de um *software* de simples entendimento e de fácil configuração e manutenção.

3.2.4 Acesso para Escrita e Leitura em Arquivos

O projeto faz interface com arquivos em dois momentos: na captura dos dados referentes à nuvem de pontos – a qual é representada pela floresta – e na escrita dos dados de saída que servirão para o desenho dos gráficos. A utilização de arquivos, ao invés de banco de dados ou outro tipo de armazenamento de informações, tornou a aplicação mais flexível e o desenvolvimento mais rápido, uma vez que não há a necessidade de nenhum tipo de gerenciamento de controle.

Houve a necessidade de criar arquivos de saída durante e ao final da execução do módulo devido ao desenho dos gráficos que representam a floresta, as ovelhas e os lobos (com seus respectivos raios). Não foram encontradas bibliotecas para esses fins e que atendessem aos requisitos do sistema. Por isso, foi utilizada uma aplicação externa, cujas entradas são os arquivos de saída do projeto (os quais descrevem o ambiente exato a cada época de execução).

Para a leitura e escrita desses arquivos, foi utilizada a biblioteca de acesso e controle de arquivos de JAVA, disponíveis no pacote *java.io*. Essa ferramenta possibilita o controle completo sobre os arquivos de forma simples e intuitiva.

3.2.5 Desenho dos Gráficos de Erro

Durante a execução do algoritmo, são calculados dois erros que ajudam na mensuração do sucesso do algoritmo, como descrito na seção 3.2.1. Esses erros são armazenados em uma coleção e, ao final da execução, são desenhados os gráficos desses erros para visualização da função de decaimento do erro. Esse artefato é de grande importância para a análise da eficácia do sistema e pode auxiliar na modificação correta dos parâmetros de execução. De forma que é uma maneira de realizar um *feedback* no sistema, a partir de seu desempenho atual.

Para construção e visualização desses gráficos, foi utilizada a biblioteca *JFreeChart* [20], devido a sua facilidade de uso e qualidade na geração dos gráficos. No sistema, simplesmente são passados os erros para cada época ou iteração e a ferramenta desenha o gráfico escalonado, com título definido pelo usuário. É possível, também, selecionar, ampliar e reduzir regiões do gráfico para uma análise mais detalhada. Outra vantagem desse *framework* é que ele permite a alteração de propriedades do gráfico, tais como cor do título, rótulo dos eixos cartesianos, intervalo dos eixos, fonte, entre outras. A Figura 11 apresenta um exemplo de gráfico gerado pelo *JFreeChart* para a função de decaimento do erro médio global dos lobos para cada época.

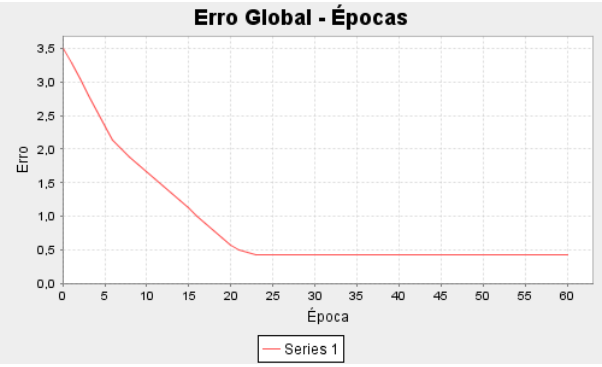

Figura 11. Exemplo de gráfico do *JFreeChart* para o erro médio global dos lobos a cada época.

3.2.6 Construção de Diagramas UML

Devido à preocupação com a documentação e facilidade de manutenção do projeto futuramente, houve uma necessidade de utilizar uma ferramenta UML [3] para criar o seu diagrama de classes. Apesar do JUDE [8] ser bastante utilizado e difundido no mercado, buscou-se uma ferramenta que automatizasse o processo de criação dos diagramas necessários

para uma boa documentação mínima. De maneira que, durante o desenvolvimento do projeto, o diagrama fosse gerado e atualizado automaticamente, sem a necessidade da criação manual desses artefatos. Por isso, o *eUML 2* [32] foi utilizado nesse trabalho.

Além de automatizar todo o processo de criação, essa ferramenta permite realizar engenharia reversa. Isto é, a partir da implementação atual, o *framework* cria e atualiza o diagrama de classes com todas as dependências, associações e herança. Outra vantagem é a sua integração com o *Eclipse*, não sendo necessária a instalação de outro aplicativo.

Nenhum de seus recursos manuais (como a utilização de *annotations* [34]) foi utilizado. Apenas foram ajustados alguns gráficos gerados por ela através de interface gráfica disponibilizada. Todos os diagramas UML foram gerados pela *eUML 2*.

3.2.7 Desenho dos Gráficos que Representam o Ambiente

O desenho do gráfico que representa o ambiente a cada época de execução do algoritmo é bastante relevante para a análise e entendimento do funcionamento do sistema. Juntamente com o gráfico da função de decaimento do erro médio global, esses gráficos ajudarão na definição dos novos parâmetros para o ambiente, como a redefinição dos critérios de parada, o número de lobos e ovelhas iniciais, entre outros.

O gráfico em questão deve conter: todas as árvores, ovelhas e lobos, assim como a circunferência de cada lobo, a qual delimita a sua área de atuação. Entretanto, não foi encontrada nenhuma biblioteca JAVA que viabilizasse o desenho do ambiente com os requisitos acima especificados.

Por isso, recorreu-se ao aplicativo de desenho de gráficos e funções *GNUPlot* [15]. Esse programa, de código aberto, permite que, através de instruções de linha de comando, os gráficos sejam gerados e salvos em imagens. A sua facilidade de uso e configuração, sua robustez e eficácia foram fundamentais para sua escolha. Ele permite a configuração de grade; o formato da imagem de saída; o tamanho da imagem; o uso de funções paramétricas, entre outros recursos.

A cada época de execução do algoritmo, são gerados dois arquivos: um armazena o posicionamento atual de todas as ovelhas, enquanto o outro armazena o dos lobos. Como as árvores nunca mudam seu posicionamento, o arquivo que armazena o posicionamento delas é

gerado uma única vez – no início de execução do sistema. Além disso, o arquivo principal (responsável por gerar os gráficos que representam o ambiente) é atualizado com os novos valores dos raios dos lobos e com informações de configurações para o gráfico, além dos novos posicionamentos dos lobos e ovelhas. A Figura 12 mostra um trecho desse arquivo principal.

```
set terminal png size 800,800
set output 'output_images\\environment\\wpa_iteration_400.png'
set parametric
set multiplot
set xrange[-79.28522536500536: 79.28522536500536]
set yrange[-79.28522536500536: 79.28522536500536]
set mxtics 5
set mytics 5
set grid xtics ytics mxtics mytics
set nokey
plot 'output\\environment\\trees.dat' notitle, 'output\\environment\\wolves400.dat' notitle,
'output\\environment\\sheeps400.dat' notitle
plot [0:2*pi] 2.75*sin(t)+2.5, 2.75*cos(t)-0.9914527298810901 notitle
plot [0:2*pi] 2.75*sin(t)+7.5, 2.75*cos(t)-1.3729436797283618 notitle
plot [0:2*pi] 2.75*sin(t)+12.5, 2.75*cos(t)-0.5438260366013565 notitle
plot [0:2*pi] 3.7564741611256283*sin(t)+17.5, 3.7564741611256283*cos(t)-0.8867433869574921 notitle
.
.
.
plot [0:2*pi] 2.75*sin(t)+47.5, 2.75*cos(t)+75.43086928874435 notitle
set nomultiplot
replot
set terminal png size 800,800
set output 'output_images\\environment\\wpa_iteration_401.png'
set parametric
set multiplot
set xrange[-79.28522536500536: 79.28522536500536]
set yrange[-79.28522536500536: 79.28522536500536]
.
.
.
```

Figura 12. Trecho do arquivo principal de construção dos gráficos que representam o ambiente a cada época.

As dez primeiras linhas são referentes à configuração do gráfico (*i.e.* imagem) a ser gerada, tais como: tamanho, formato e nome da imagem; o intervalo para os eixos cartesianos; definição e configuração da grade, entre outros (para mais detalhes sobre essas configurações, recomenda-se a leitura da documentação do *GNUPlot* [15]). As duas linhas seguintes especificam os arquivos que contêm o posicionamento dos elementos do ambiente para aquela época.

A partir da linha treze até a vinte, são especificadas as circunferências de cada lobo a partir das funções paramétricas: *r*sin(t)+a* e *r*cos(t)+b*, onde *r* corresponde ao raio do lobo posicionado no ponto cartesiano (*a*, *b*) e *t* é o parâmetro, o qual varia dentro do intervalo [0: 2π]. Essa forma de representação geométrica foi utilizada devido a sua simplicidade de

descrição e por utilizar poucos operadores matemáticos, aumentando o desempenho do programa de geração de gráficos.

A partir da linha vinte e três, em diante, começa o trecho de código referente à configuração do gráfico a ser gerado para a época seguinte.

A geração das imagens correspondentes aos gráficos descritos no arquivo principal é realizada através de um arquivo *batch*, o qual chama o compilador do *GNUPlot* passando como parâmetro o arquivo principal criado pelo projeto.

Capítulo 4

Análise dos Resultados

Neste capítulo, serão apresentados os resultados dos experimentos realizados para analisar o funcionamento do algoritmo desenvolvido. Serão discutidos o método, as variáveis e os parâmetros utilizados nos experimentos e, em seguida, serão apresentados os resultados para quatro diferentes funções (*i.e.* configurações de árvores): linear, quadrática, cosseno e tangente hiperbólica.

4.1 Planejamento dos Experimentos

4.1.1 Descrição dos Parâmetros Utilizados e seus Valores

A Tabela 1 lista os valores de todos os parâmetros do sistema que foram utilizados e investigados durante os experimentos.

Tabela 1. Parâmetros utilizados no sistema

Parâmetro	Valor	Parâmetro	Valor
Núm. Dimensões	*2*	**Dist. Inicial Ovelha-Árvore**	*2 unidades cartesiana*
Função Lobo	*Gaussiana*	**Núm. Épocas**	*60*
Arquivo Entrada	*Variável*	**Erro Mín. Global**	*0.05 unidades cartesianas*
Núm. Inicial Lobos[1]	*Variável*	**Erro Mín. Lobo**	*Variável*
Núm. Inicial Ovelhas[1]	*40% das árvores*	**Desvio-Padrão Mín. Lobo**	*Variável*
Dist. Inicial Lobo-Ovelha	*1 unidade cartesiana*	**Tamanho da Floresta**	*100 árvores*

[1] Relativo a um lado da floresta. Isto é, o número de elementos no ambiente será o dobro desse valor.

Os parâmetros da Tabela 1 definem tanto a configuração do ambiente quanto o modo como o algoritmo irá executar. Os parâmetros que configuram o ambiente são: número de dimensões do ambiente; o arquivo de entrada que representa a floresta (*i.e.* nuvem de pontos); o tamanho da floresta; o número inicial de lobos; o número inicial de ovelhas; a distância inicial dos lobos para as ovelhas; e, a distância inicial das ovelhas para as árvores. Por exemplo, este último parâmetro determina a distância que as ovelhas ficarão da árvore com maior ou com menor ordenada. A Figura 13 exemplifica a função desse parâmetro no ambiente inicial.

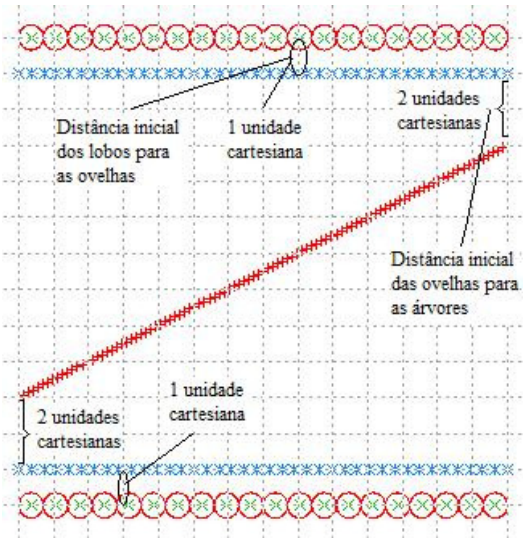

Figura 13. Distâncias iniciais entre os elementos do ambiente. Lobos (x), ovelhas (*) e árvores (+).

Já os parâmetros que definem como o algoritmo será executado são: a função associada ao lobo (não utilizada nesse módulo do WPA); o erro médio mínimo do lobo; e, o desvio-padrão mínimo do lobo. Esses dois últimos parâmetros definirão a base de comparação para determinar o operador a ser utilizado pelo lobo. Por exemplo, se o desvio-padrão do lobo for menor do que o parâmetro desvio-padrão mínimo do lobo, então se diz que o desvio-padrão do lobo é baixo.

4.1.2 Método de Experimentação Utilizado

Os experimentos foram planejados através da técnica de Fatorial 2^k [13] com três fatores (logo, k = 3), obtendo-se oito simulações para cada função. Essa técnica foi escolhida pois é simples de ser realizada e permite a compreensão e análise dos efeitos de cada fator sobre as

variáveis de resposta. Dessa forma, os dois níveis (*i.e.* valores) assumidos pelos fatores foram determinados para que viabilizassem essa análise.

Os fatores utilizados para a realização dos experimentos foram: o número de lobos iniciais, o erro médio e o desvio-padrão mínimo do lobo. Dessa forma, foram escolhidos dois fatores que modificam o funcionamento do algoritmo e um fator que modifica a configuração inicial do ambiente, para que fosse possível a análise da influência desses tipos de parâmetros no sistema. Além disso, as condições de parada do sistema são determinadas pelo número de épocas e erro mínimo global.

Os níveis utilizados para cada fator foram: número inicial de lobos (5% e 20% do número de árvores); erro médio mínimo do lobo – EMM - (0.5 e 0.15 unidades cartesianas); e, desvio-padrão mínimo do lobo – DPM - (0.5 e 0.15 unidades cartesianas). A Tabela 2 mostra as configurações utilizadas nos experimentos para cada função.

Tabela 2. Configurações dos experimentos e variação dos fatores.

Configurações	Mín. Erro Lobo (EMM)	Mín. Desvio Lobo (DPM)	Núm. Inicial Lobos
Configuração 1	*0.5 un. cartesianas*	*0.5 un. cartesianas*	*5% das árvores*
Configuração 2	*0.5 un. cartesianas*	*0.5 un. cartesianas*	*20% das árvores*
Configuração 3	*0.5 un. cartesianas*	*0.15 un. cartesianas*	*5% das árvores*
Configuração 4	*0.5 un. cartesianas*	*0.15 un. cartesianas*	*20% das árvores*
Configuração 5	*0.15 un. cartesianas*	*0.5 un. cartesianas*	*5% das árvores*
Configuração 6	*0.15 un. cartesianas*	*0.5 un. cartesianas*	*20% das árvores*
Configuração 7	*0.15 un. cartesianas*	*0.15 un. cartesianas*	*5% das árvores*
Configuração 8	*0.15 un. cartesianas*	*0.15 un. cartesianas*	*20% das árvores*

4.2 Experimentos

Esta seção apresenta os resultados obtidos para as diferentes configurações de floresta, a saber: função linear, função quadrática, função cosseno e função tangente hiperbólica. Foram escolhidas duas funções polinomiais e duas funções trigonométricas, a fim de avaliar o comportamento do algoritmo para dois grupos de funções diferentes. Os resultados obtidos

serão analisados através do erro médio global (em unidades cartesianas) para cada configuração do experimento e através da influência de cada fator no resultado final.

Ressalta-se que os resultados obtidos com a execução do algoritmo para uma mesma configuração de parâmetros e da floresta foram sempre os mesmos. Pois, da maneira que o sistema foi modelado, para haver uma variação dos resultados obtidos, os lobos deveriam ser posicionados de forma aleatória no ambiente. Portanto, como os lobos são deterministicamente posicionados no ambiente, os resultados obtidos dada certa configuração de parâmetros serão sempre os mesmos.

4.2.1 Função Linear

A Tabela 3 apresenta os resultados dos experimentos para a função linear $y\ (x) = 0{,}5\ x + 5$, definida no intervalo fechado [0, 14].

Tabela 3. Resultados da experimentação obtidos para a função linear.

Configurações	Erro Médio Global	Desvio-Padrão	Convergência
Configuração 1	*0,29270*	*0,30518*	*33ª Época*
Configuração 2	*0,21891*	*0,23823*	*35ª Época*
Configuração 3	*0,47177*	*0,45831*	*31ª Época*
Configuração 4	*0,30459*	*0,10149*	*37ª Época*
Configuração 5	*0,27912*	*0,29624*	*41ª Época*
Configuração 6	*0,13837*	*0,14793*	*37ª Época*
Configuração 7	*0,46275*	*0,45784*	*32ª Época*
Configuração 8	*0,09772*	*0,05912*	*41ª Época*

A partir da Tabela 3, nota-se que uma maior quantidade de lobos no ambiente refletiu em uma diminuição do erro global, como esperado. A maior diferença obtida pela variação exclusiva desse fator foi de 0,36503 unidades (entre as Configurações 7 e 8), enquanto a menor foi de 0,06379 unidades (entre as Configurações 1 e 2).

A análise também revelou que quando apenas o EMM reduziu, houve uma diminuição do erro global em todas as comparações, onde a menor diferença foi de 0,00902 unidades (entre as Configurações 3 e 7) e a maior, 0,20687 unidades (entre as Configurações 4 e 8).

Entretanto, foi observado que quando ocorreu apenas uma diminuição no DPM houve um aumento do erro global, com exceção da comparação entre as Configurações 6 e 8.

Ao analisar as configurações que possuíam valores diferentes para o EMM e para o DPM e um mesmo número de lobos no ambiente, os melhores resultados foram os obtidos quando o EMM era menor do que o DPM. Já quando os seus valores foram iguais para um mesmo número inicial de lobos no ambiente, nada pôde ser afirmado em relação ao comportamento do erro global.

Por fim, notou-se que quando o EMM assumiu o menor valor, provocou um aumento no número de épocas necessárias para que o algoritmo pudesse convergir.

A Figura 14 e a Figura 15 apresentam o gráfico do decaimento do erro e o seu respectivo estado final, para as configurações que apresentaram o menor e o maior erro médio global, respectivamente.

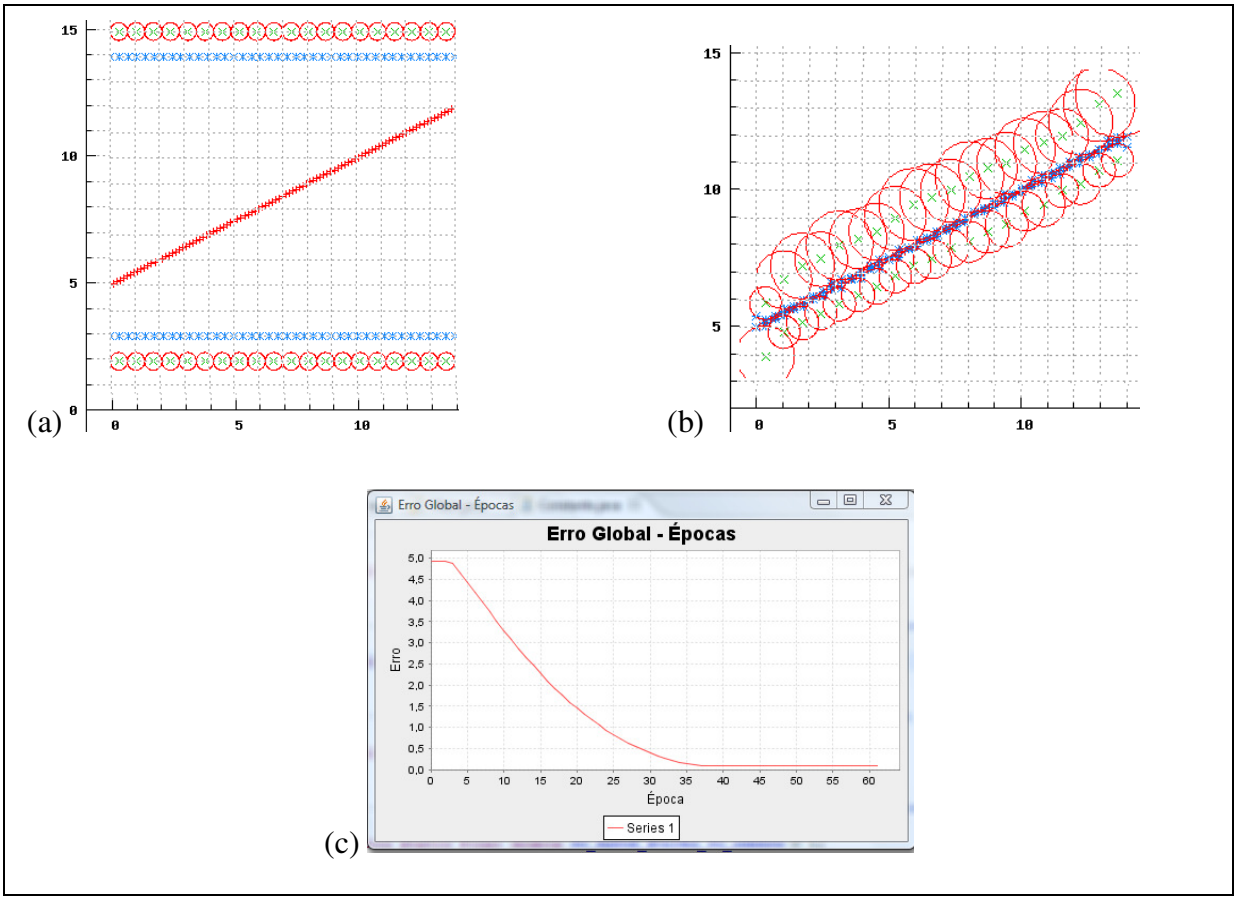

Figura 14. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a melhor configuração de experimento: Configuração 8.

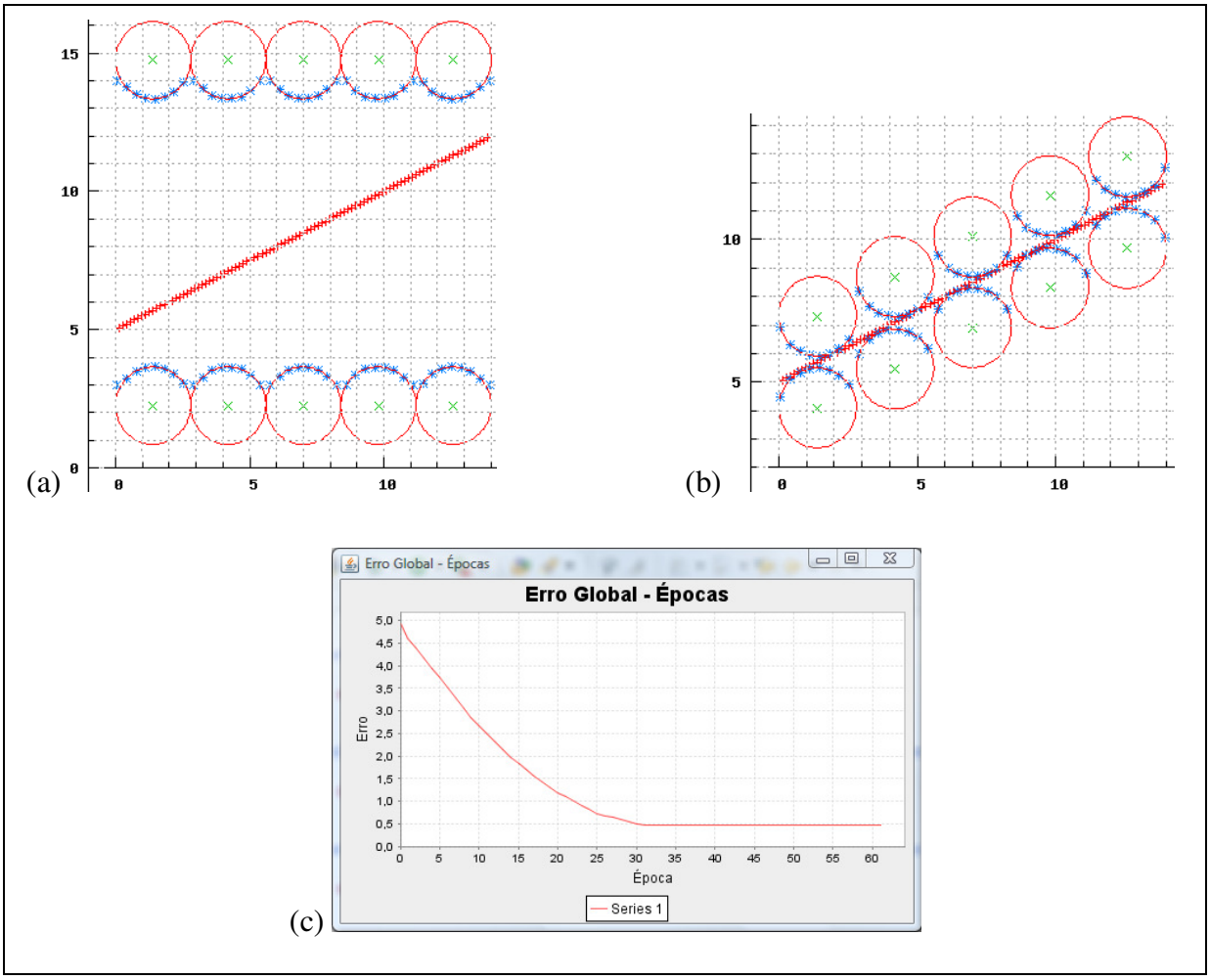

Figura 15. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a pior configuração de experimento: Configuração 3.

Através dessas observações e variações do erro médio global para a mudança de cada um dos fatores, conclui-se que para a função linear o número de lobos no ambiente é o mais relevante para a melhora do resultado final, seguido pelo EMM. Pois, a variação exclusiva do número inicial de lobos provocou uma maior redução do erro médio global, quando comparado com o EMM. Além disso, o resultado obtido pela Configuração 8 foi satisfatório e apresentou o menor desvio-padrão global, pois as ovelhas ficaram bem próximas das árvores, formando o contorno da função.

4.2.2 Função Quadrática

A Tabela 4 apresenta os resultados dos experimentos para a função quadrática $y\,(x) = -0{,}2\,x^2 + 5$, definida no intervalo fechado [-5, 5].

Tabela 4. Resultados da experimentação obtidos para a função quadrática.

Configurações	Erro Médio Global	Desvio-Padrão	Convergência
Configuração 1	*0,43595*	*0,39527*	*25ª Época*
Configuração 2	*0,31977*	*0,34710*	*61ª Época*
Configuração 3	*0,47849*	*0,40943*	*26ª Época*
Configuração 4	*0,26992*	*0,12166*	*31ª Época*
Configuração 5	*0,35100*	*0,36953*	*60ª Época*
Configuração 6	*0,28757*	*0,32814*	*60ª Época*
Configuração 7	*0,38403*	*0,44194*	*28ª Época*
Configuração 8	*0,11890*	*0,09962*	*32ª Época*

A partir da análise dos resultados apresentados na Tabela 4, percebe-se que quando ocorreu somente um aumento do número de lobos do ambiente, houve uma diminuição do erro médio global para todas as comparações realizadas. Dentre elas, a que apresentou maior variação foi entre as Configurações 7 e 8 (0,26513 unidades), enquanto a menor, foi entre as Configurações 5 e 6 (0,06343 unidades).

A diminuição, somente, do EMM provocou uma diminuição do erro médio global em todos os experimentos sendo a menor diferença 0,0322 unidades (entre as Configurações 2 e 6) e a maior 0,15102 unidades (entre as Configurações 4 e 8). A diminuição do EMM provocou, na maioria das comparações realizadas, um aumento no número de épocas necessárias para que o algoritmo pudesse convergir.

Nos experimentos em que o DPM e o EMM eram diferentes, os melhores resultados foram obtidos quando o EMM e o número de lobos assumiram o menor nível (i.e. Configuração 5) e quando o DPM era igual a 0,15 e o número de lobos igual a 20% das árvores (i.e. Configuração 4). Entretanto, quando os valores do EMM e do DPM foram iguais e o número de lobos foi o mesmo, houve uma diminuição do erro médio global.

A diminuição do DPM associada ao maior número inicial de lobos provocou uma diminuição do erro médio global (entre as Configurações 2 e 4; e, entre as Configurações 6 e 8). Entretanto, notou-se que a diminuição do DPM associada ao menor número inicial de lobos provocou um aumento do erro médio global (entre as Configurações 1 e 3; e, entre as

Configurações 5 e 7). A maior variação do resultado final ao modificar exclusivamente o DPM foi de 0,16867 unidades, entre as Configurações 6 e 8.

Através dessa experimentação, conclui-se que para a função quadrática o fator com maior influência sobre o resultado final é o número de lobos inicial, seguido pelo DPM, ao contrário da função linear. Pois, refletiram uma maior variabilidade do erro médio global ao modificar unicamente seu valor. Além disso, o desvio-padrão global foi maior do que o erro médio global para a metade dos experimentos. Logo, conclui-se que há uma grande variabilidade das distâncias das ovelhas à função. Ou seja, enquanto algumas ovelhas estão bastante próximas da floresta, outras estão proporcionalmente bem distantes das árvores.

A Figura 16 e a Figura 17 apresentam a melhor e pior configuração de experimento, respectivamente.

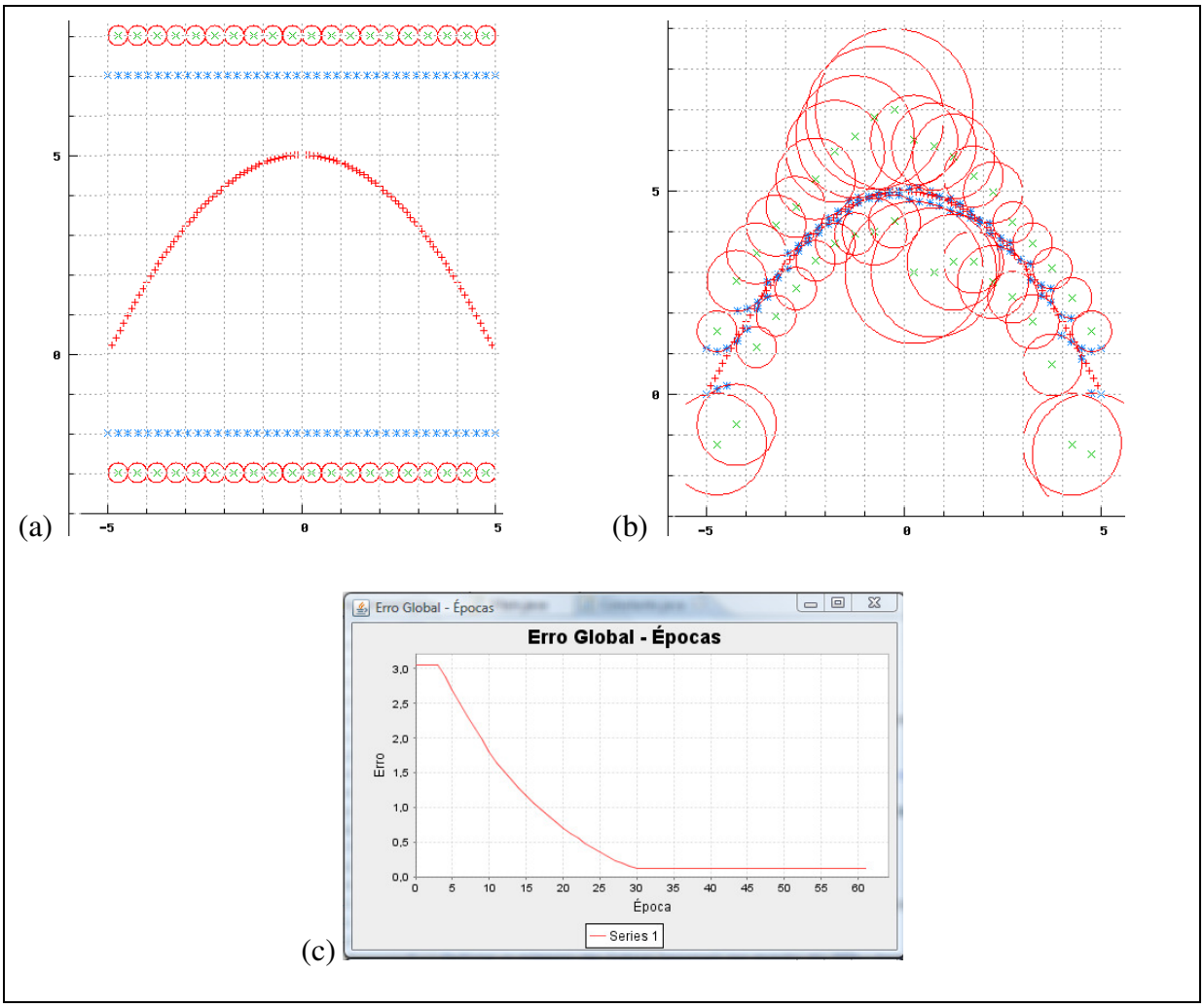

Figura 16. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a melhor configuração de experimento: Configuração 8.

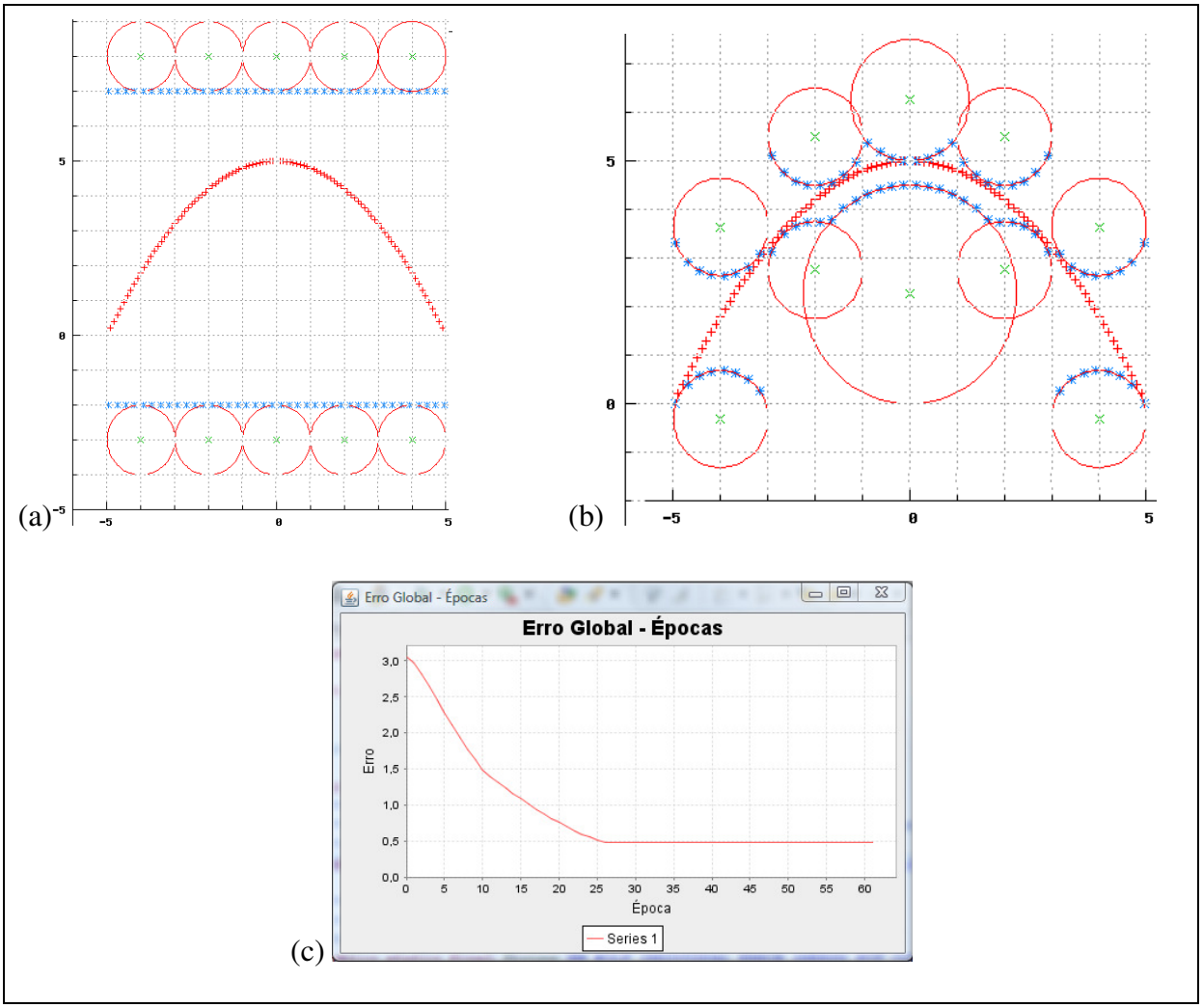

Figura 17. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a pior configuração de experimento: Configuração 3.

4.2.3 Função Cosseno

A Tabela 5 apresenta os resultados dos experimentos para a função cosseno $y\ (x) = cos(x)$, definida no intervalo fechado [0, 12,5].

Ao analisar os resultados apresentados na Tabela 5, verifica-se que quando ocorreu exclusivamente um aumento do número de lobos do ambiente, houve uma diminuição do erro médio global para todas as comparações realizadas. Dentre elas, a que apresentou maior variação foi entre as Configurações 7 e 8 (0,32086 unidades), enquanto a menor, foi entre as Configurações 1 e 2 (0,0439 unidades).

Tabela 5. Resultados da experimentação obtidos para a função cosseno.

Configurações	Erro Médio Global	Desvio-Padrão	Convergência
Configuração 1	*0,47492*	*0,42865*	*11ª Época*
Configuração 2	*0,43102*	*0,37498*	*53ª Época*
Configuração 3	*0,47950*	*0,41291*	*13ª Época*
Configuração 4	*0,25939*	*0,11985*	*18ª Época*
Configuração 5	*0,44491*	*0,43191*	*16ª Época*
Configuração 6	*0,37049*	*0,34426*	*60ª Época*
Configuração 7	*0,43186*	*0,42759*	*24ª Época*
Configuração 8	*0,11100*	*0,07625*	*42ª Época*

Ao analisar exclusivamente a variação do EMM, verifica-se que quando houve uma diminuição, o erro médio global reduziu em todos os experimentos sendo a menor diferença 0,03001 unidades (entre as Configurações 1 e 5) e a maior 0,14839 unidades (entre as Configurações 4 e 8). Foi observado, também, que a diminuição do EMM provocou um aumento no número de épocas necessárias para que o algoritmo pudesse convergir em todos os experimentos.

A análise do impacto sobre o resultado final da variação do DPM mostrou que quando somente o DPM diminuiu, houve uma diminuição do erro médio global, com exceção da comparação entre as Configurações 1 e 3, na qual o erro foi praticamente o mesmo – diferença de 0,00458 unidades. A maior variação do erro médio global ao variar exclusivamente o DPM foi de 0,25949 unidades (entre as Configurações 6 e 8).

Em seguida, a análise da relação entre o DPM e o EMM revelou que quando os valores eram diferentes, os melhores resultados foram obtidos quando o EMM e o número inicial de lobos eram os menores (Configuração 5) e quando o DPM era o menor e o número de lobos, o maior (Configuração 4). Contudo, quando as configurações eram iguais e o número inicial de lobos o mesmo, houve uma diminuição do erro médio global.

Os resultados obtidos com a função cosseno foram bastante semelhantes aos obtidos pela função quadrática, tanto no comportamento do erro médio global ao variar os fatores quanto na relevância desses fatores para a obtenção de melhores resultados. Entretanto, houve

uma melhora significativa em relação à função cosseno, pois em todos os experimentos o desvio-padrão foi menor do que o erro médio global. Logo, as distâncias das ovelhas para as árvores não variaram tanto quanto os experimentos anteriores.

A Figura 18 e a Figura 19 apresentam a melhor e pior configuração de experimento, respectivamente.

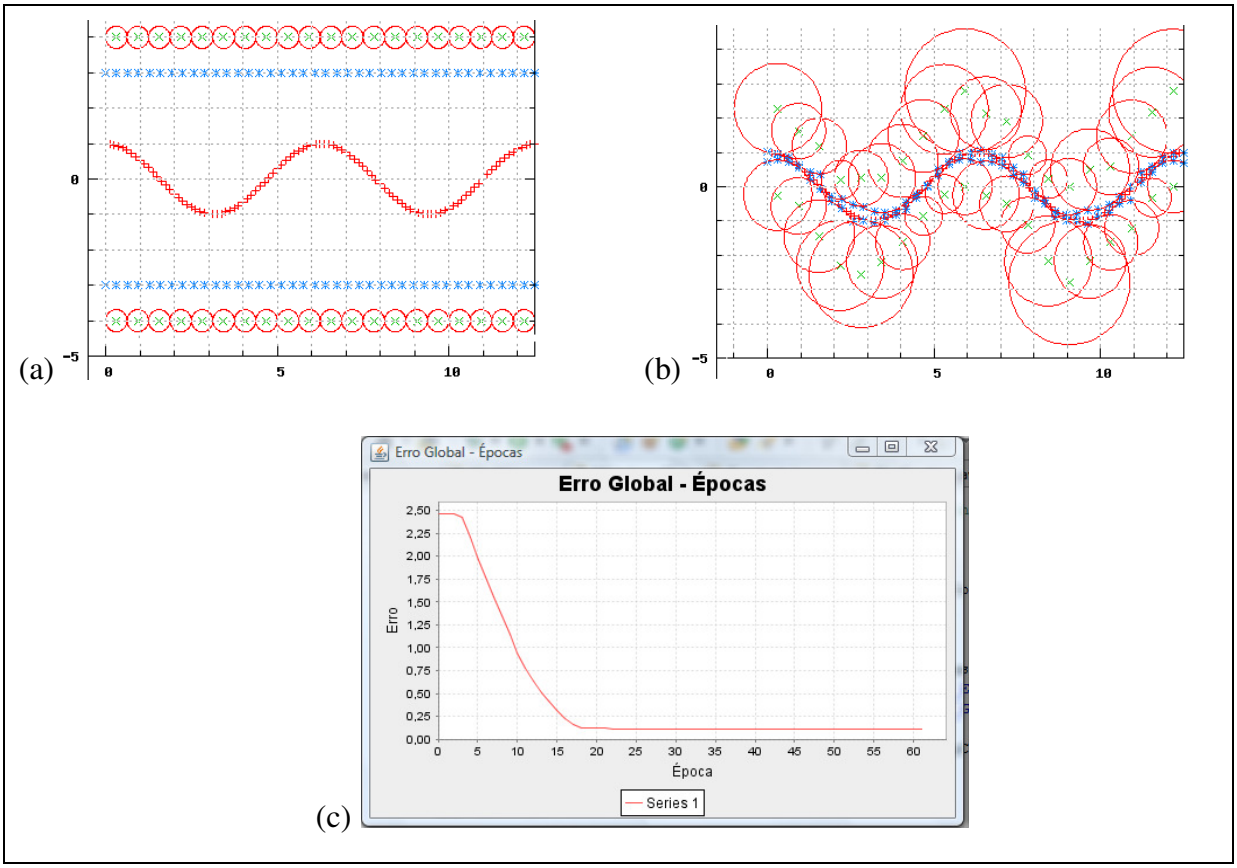

Figura 18. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a melhor configuração de experimento: Configuração 8.

4.2.4 Função Tangente Hiperbólica

A Tabela 6 apresenta os resultados dos experimentos para a função tangente hiperbólica *y (x) = 2 * tangh(x)*, definida no intervalo fechado [-10, 10].

A análise dos resultados apresentados na Tabela 6 mostra que quando ocorreu somente uma diminuição do EMM houve uma redução do erro médio global em todos os experimentos, onde a menor diferença foi de 0,00177 unidades (entre as Configurações 3 e 7) e a maior foi de 0,26155 unidades (entre as Configurações 4 e 8). Como na experimentação

anterior, foi observado que a diminuição do EMM provocou um aumento no número de épocas necessárias para que o algoritmo pudesse convergir.

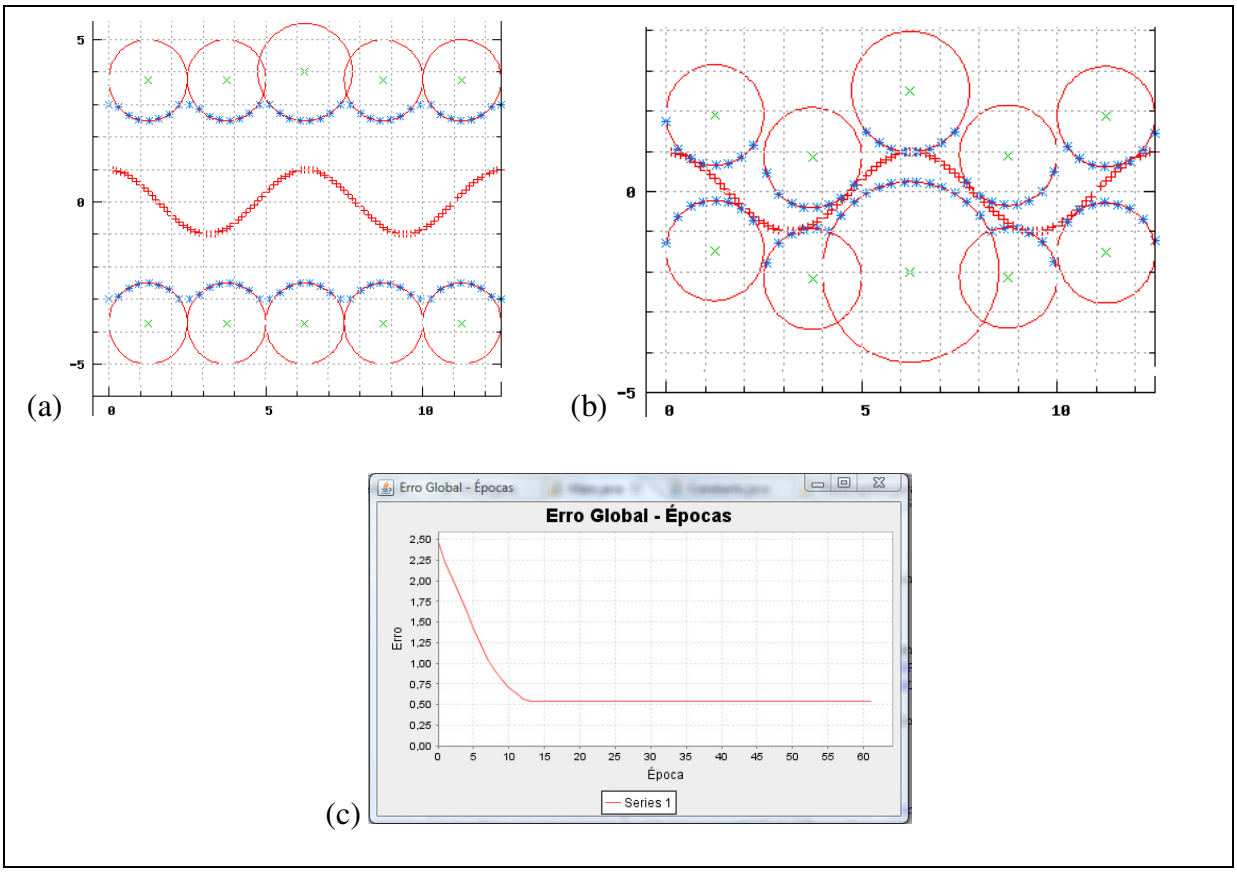

Figura 19. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a pior configuração de experimento: Configuração 3.

Tabela 6. Resultados da experimentação obtidos para a função tangente hiperbólica.

Configurações	Erro Médio Global	Desvio-Padrão	Convergência
Configuração 1	*0,35804*	*0,24899*	*21ª Época*
Configuração 2	*0,37126*	*0,18189*	*25ª Época*
Configuração 3	*0,52966*	*0,58227*	*19ª Época*
Configuração 4	*0,39397*	*0,13101*	*26ª Época*
Configuração 5	*0,30564*	*0,24201*	*61ª Época*
Configuração 6	*0,16434*	*0,17146*	*61ª Época*
Configuração 7	*0,52789*	*0,58237*	*21ª Época*
Configuração 8	*0,13242*	*0,08611*	*27ª Época*

A variação unicamente do DPM mostrou que quando houve uma diminuição do seu valor, o erro médio global aumentou (entre as Configurações 1 e 3; Configurações 2 e 4; e, Configurações 5 e 7), com exceção da comparação entre as Configurações 6 e 8, a qual o erro médio global diminui. A maior variação do resultado final obtida com a modificação de unicamente do DPM foi de 0,22225 unidades (entre as Configurações 5 e 7).

Ao analisar a variação do fator número inicial de lobos, observa-se que quando ocorreu unicamente um aumento do seu valor, houve uma diminuição do erro global para a maioria das comparações realizadas, com exceção da comparação entre as Configurações 1 e 2. A maior variação do resultado final obtida com a modificação de unicamente o número inicial de lobos foi de 0,39547 unidades (entre as Configurações 7 e 8).

A partir dessas observações, conclui-se que, assim como na função linear, o fator mais relevante para a diminuição do erro médio global é o número inicial de lobos, seguido pelo EMM, pois sua variação apresenta uma maior modificação do erro médio global. Além disso, os valores do desvio-padrão global apresentados informam que há uma variação grande das distâncias das ovelhas para a floresta, principalmente para as Configurações 3, 6 e 7, as quais o desvio-padrão é maior do que o erro médio global.

A Figura 20 e a Figura 21 apresentam a melhor e pior configuração de experimento, respectivamente.

4.2.5 Experimentos com Permutação dos Operadores do Lobo

Os resultados dos experimentos apresentados nesta seção foram obtidos através da permutação dos operadores do lobo (ajuste de raio e ajuste de posicionamento) para o pior e o melhor caso de cada experimentação descrita anteriormente. Ao invés de escolher o operador de ajuste de raio, o lobo escolherá o operador de ajuste de posicionamento, e vice-versa.

O objetivo desses experimentos foi verificar se o algoritmo apresentaria um melhor resultado com a modificação do módulo de decisão do operador a ser utilizado. A Tabela 7 mostra os resultados obtidos para todos os experimentos realizados.

A análise comparativa dos resultados obtidos nesses experimentos com os obtidos nas seções anteriores mostra que o erro médio global diminuiu apenas para a Configuração 3 das funções linear e tangente hiperbólica.

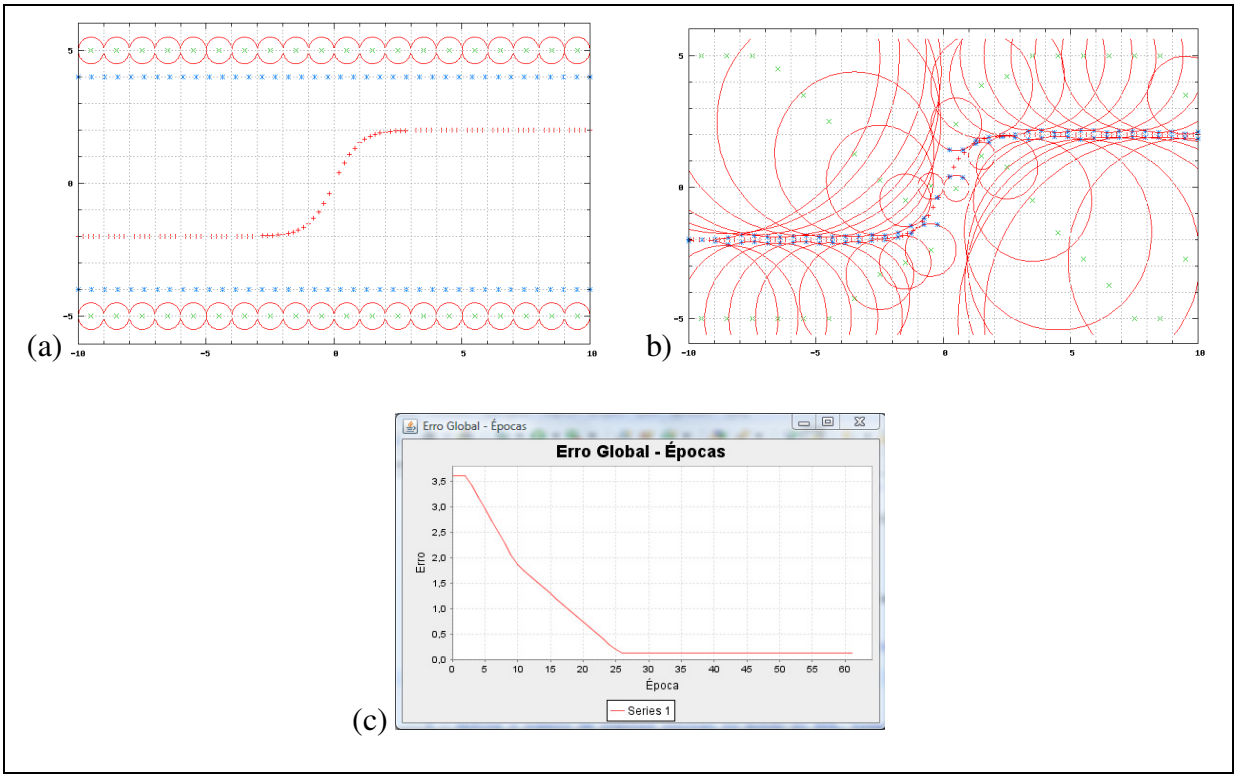

Figura 20. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a melhor configuração de experimento: Configuração 8.

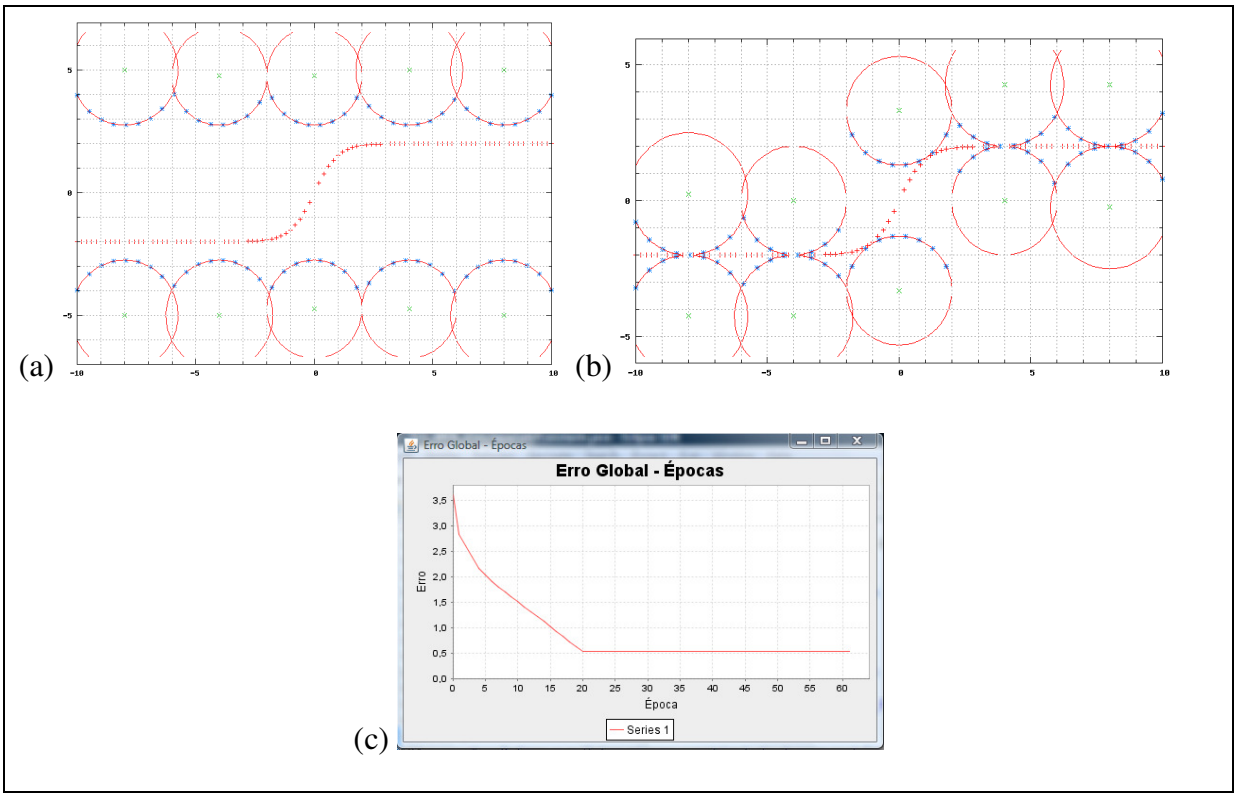

Figura 21. Estado inicial do ambiente (a), seu estado final (b) e o decaimento do erro médio global (c) para a pior configuração de experimento: Configuração 3

Tabela 7. Resultados dos experimentos com a permutação dos operadores.

Função	Configuração	Erro Médio Global	Desvio-Padrão	Convergência
Linear	**Configuração 3**	*0,36871*	*0,46042*	*28ª Época*
	Configuração 8	*0,21863*	*0,33515*	*51ª Época*
Quadrática	**Configuração 3**	*0,83351*	*0,70213*	*21ª Época*
	Configuração 8	*0,64305*	*0,70923*	*61ª Época*
Cosseno	**Configuração 3**	*0,56925*	*0,46344*	*13ª Época*
	Configuração 8	*0,26945*	*0,30389*	*60ª Época*
Tang. Hiper.	**Configuração 3**	*0,41985*	*0,38304*	*20ª Época*
	Configuração 8	*0,25242*	*0,28039*	*61ª Época*

A época de convergência do algoritmo aumentou para cinco das configurações testadas; permaneceu a mesma para a Configuração 3 da função cosseno; e, diminuiu para a Configuração 3 das funções linear e quadrática.

Dessa forma, pode-se concluir que a configuração inicial dos operadores a serem utilizados apresentou melhores resultados do que quando são permutados. Entretanto, como houve a diminuição do erro médio global para duas configurações de funções distintas (as quais representam 25% dos casos testados), somente uma análise mais aprofundada poderá indicar qual a melhor configuração para a decisão do operador a ser utilizado em cada caso.

A Figura 22 mostra o estado final e a sua respectiva função de decaimento do erro – obtidos através da permutação de seus operadores – para a Configuração 3 da função linear e para a Configuração 8 da função quadrática.

Através da Figura 22, é possível perceber que as ovelhas ficaram melhores adaptadas para a função linear utilizando os operadores permutados. Isso deve-se à tendência dos lobos aumentarem seus raios de atuação, ao invés de ajustarem seus posicionamentos. Dessa forma, o erro médio global obtido foi menor.

Entretanto, para a função quadrática ilustrada na Figura 22, o erro médio global obtido foi maior, uma vez que as ovelhas não formaram corretamente o contorno da nuvem de pontos de entrada.

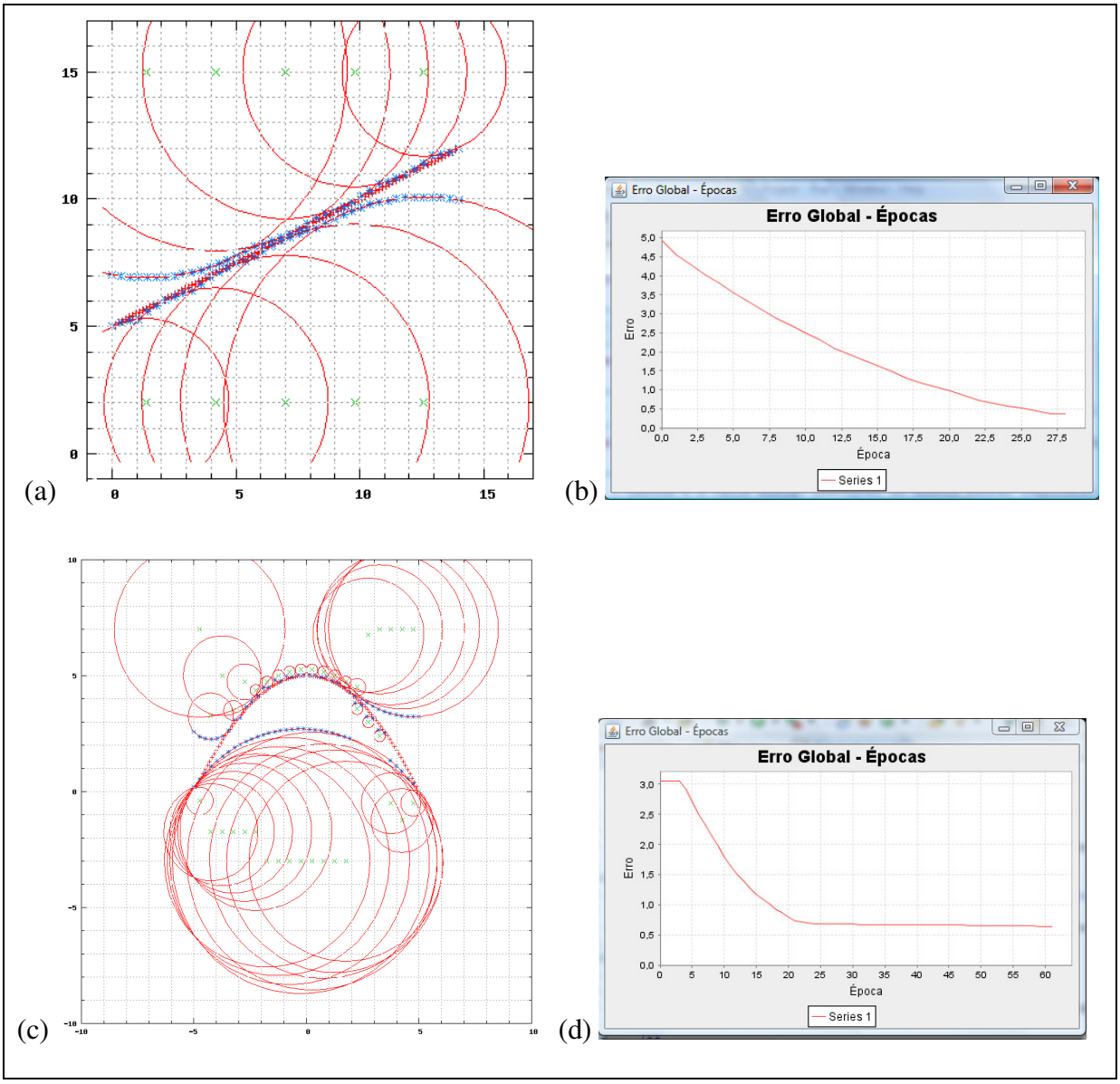

Figura 22. Estado final (a) e função de decaimento do erro (b) para a Configuração 3 da função linear. Estado final (c) e função de decaimento do erro (d) para a Configuração 8 da função quadrática.

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Discussão dos Resultados e Contribuições

Este trabalho teve como principal motivação o desenvolvimento inicial do primeiro módulo do WPA. Ele é responsável por coordenar as ações dos lobos e ovelhas no ambiente, a fim de posicioná-los corretamente e definir seus respectivos raios de atuação. De forma que somente essas informações de saída sejam utilizadas como entrada do módulo seguinte, o qual completa o cálculo da função a ser aproximada.

Aqui, lobos são agentes que agem em conformidade com seus objetivos planejados. Através do erro médio e do desvio-padrão individual, eles podem inferir qual a operação mais adequada para cada situação, ou seja, deslocamento ou ajuste do raio de atuação. Conforme a análise realizada no Capítulo 4, uma variação dos limiares do erro médio e desvio-padrão individual têm uma relativa influência sobre o resultado final (*i.e.* o erro médio global), principalmente se combinada com um aumento do número de lobos no ambiente (*i.e.* matilha).

Experimentalmente, observou-se que o número de lobos não deve ser baixo (< 5% das árvores) nem muito alto (> 70% das árvores). Um número baixo de lobos provocaria ou um aumento do erro médio global ou em um posicionamento e raio finais não adequados corretamente à floresta. Caso a quantidade de lobos seja maior que o necessário, a razão entre a cardinalidade da alcatéia e da floresta será próxima de um, o que indica que existe aproximadamente um lobo para cada árvore. Dessa forma, não vale a pena o uso do algoritmo pois, a tarefa será a mesma que aproximar a função diretamente pela nuvem de pontos dada.

Na abordagem adotada no WPA, as ovelhas apenas reagem à ação do lobo. Elas servem para guiar as ações da alcatéia e ajudar a definir o posicionamento e raio final para cada lobo individualmente. Sua população inicial é também fundamental para um melhor funcionamento do algoritmo. Por exemplo, seja uma floresta de cardinalidade 100 e uma população inicial de ovelhas igual a 5. Pode-se facilmente chegar a um erro global médio

muito próximo de zero. Entretanto, isso não significará que o resultado final da aproximação será bom, tendo em vista que a representatividade da maioria dos pontos foi perdida em decorrência da pouca quantidade de ovelhas.

Por outro lado, caso seja definido um número inicial de ovelhas no ambiente igual a 90, há um aumento considerável no processamento do algoritmo e não há garantias de que o resultado da aproximação será muito melhor do que uma população inicial de 20 ovelhas. Por isso a população inicial de ovelhas tem uma influência grande na precisão da aproximação da função.

Conclui-se também, de acordo com as experimentações realizadas, que o número de lobos é o fator de maior influência no resultado final (dentre todos os que foram analisados). O DPM e o EMM mostraram possuir igual relevância para o resultado final. Pois, uma variação exclusiva do DPM provocou uma variação maior do erro médio global para as funções quadrática e cosseno, enquanto o EMM foi mais relevante para as funções linear e tangente hiperbólica. Isso permite afirmar que é a combinação das duas métricas combinadas que produziu os bons resultados obtidos.

Este trabalho refere-se à primeira versão do módulo um do WPA. Várias melhorias e análises mais detalhadas podem e devem ser realizadas, a fim de obter melhores resultados e até mesmo conhecer melhor a real influência das métricas sobre os operadores da técnica. A principal tarefa a implementar é estabelecer um mecanismo e protocolo de comunicação entre os lobos, para que eles possam coordenar melhor suas ações como time; no momento as informações sociais são utilizadas, mas para o benefício de cada lobo individualmente. De qualquer forma, esse trabalho deu um passo importante para a viabilização de uma ferramenta inteligente de aproximação de funções baseada em inteligência coletiva e com um custo computacional menor do que as técnicas tradicionais.

Apesar desse módulo inicial do WPA não possuir uma comunicação entre os lobos, os resultados obtidos foram satisfatórios e motivadores para a continuação de seu desenvolvimento. Em longo prazo, pode-se dizer que este trabalho contribuiu para solução do problema de aproximação de funções utilizando Computação Inteligente. Contudo, em curto prazo, este trabalho contribuiu para a modelagem e para o início da implementação do WPA.

5.2 Sugestões de Trabalhos Futuros

Durante o desenvolvimento deste trabalho várias idéias surgiram. Porém, nem todas puderam ser implementadas por razões de escopo escolhido e complexidade da implementação dado o tempo exíguo de execução do trabalho. Abaixo, listam-se sugestões de trabalhos futuros e melhorias a serem realizadas no algoritmo:

- Estudar e construir mecanismo que possibilite a comunicação entre os lobos;
- Definir e construir operadores de recrutamento e de descarte de lobos, baseados em comunicação e resultados instantâneos obtidos pela alcatéia durante sua "caça";
- Construir mecanismo que descarte o lobo menos adaptado (com maior erro médio individual) em uma mesma abscissa ou outro método de remoção seletiva dos lobos para evitar "superpopulação";
- Realizar testes de escalabilidade do algoritmo;
- Verificar quais outros parâmetros têm influência direta sobre o resultado final e quão grande é o impacto referente às variações desses parâmetros (*e.g.* número inicial de ovelhas);
- Construir mecanismo automático e inteligente de escolha dos melhores valores para os parâmetros, dada uma nuvem de pontos qualquer;
- Equipar o algoritmo para trabalhar com coordenadas multidimensionais;
- Variar a distribuição inicial das ovelhas e lobos dependendo da configuração da floresta (*e.g.* determinar uma distribuição de probabilidade que melhor se adéque);
- Determinar quando será necessária a alcatéia em ambos os lados da floresta;
- Desenvolver interface gráfica para melhor informar o usuário sobre o processamento instantâneo;
- Modificar o algoritmo para que seja possível sua execução distribuída de forma mais eficiente;

- Experimentar novas funções de vizinhança para os lobos, inclusive experimentar mudanças da dinâmica de função de vizinhança;
- Experimentar versões do WPA com nuvens de pontos dinâmicas;
- Transformar o vetor de saída na função aproximada, ou seja implementar o módulo 2 do WPA.

5.3 Dificuldades Encontradas

A construção deste módulo inicial do WPA esteve associada a algumas dificuldades, tais como: (i) ausência de biblioteca JAVA para desenho de gráficos com recursos suficientes para o desenho dos gráficos que representam o ambiente; (ii) o aprendizado da sintaxe e planejamento da construção dos arquivos de saída que representam o ambiente para serem desenhados no *GNUPlot* levou algum tempo para ser resolvido; e, (iii) por se tratar de um algoritmo inovador, não foram encontradas referências sobre algo parecido na literatura para que pudesse servir de base de consulta ou comparação.

Bibliografia

[1] BASTOS FILHO, C. J. A., et al. **A Novel Search Algorithm based on Fish School Behavior**. In: 2008 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2008), 2008, Cingapura, pp. 2646-2651.

[2] BONABEAU, E.; DORIGO, M.; THERAULAZ, T. **Swarm Intelligence: From Natural to Artificial Systems**. New York: Oxford University Press, 1999.

[3] BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML – Guia do Usuário**. 2 ed. São Paulo: Editora Campus, 2005, 474 p.

[4] BRAGA, A. P.; CARVALHO, A. C. P. L. F.; LUDERMIR, T. B.. **Redes Neurais Artificiais: teoria e aplicações**. 2ª Edição. Rio de Janeiro: LTC, 2007.

[5] CASTRO, L.; VON ZUBEN, F. **Conceitualização**. Tópicos em Sistemas Inteligentes II. Pós-Graduação. DCA/FEE/Unicamp. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/topico2_03.pdf> Acesso em: 19 de março de 2009.

[6] CASTRO, L.; VON ZUBEN, F.. **Computação Evolutiva**. Tópicos em Sistemas Inteligentes II. Pós-Graduação. DCA/FEE/Unicamp. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/topico3_03.pdf> Acesso em: 19 de março de 2009.

[7] CASTRO, L.; VON ZUBEN, F.. **Introdução à Computação Natural**. Tópicos em Sistemas Inteligentes II. Pós-Graduação. DCA/FEE/Unicamp. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/topico1_03.pdf> Acesso em: 19 de março de 2009.

[8] CHANGE VISION INC. **Jude: Desing & Communication**. Disponível em: <http://jude.change-vision.com/jude-web/index.html> Acesso em: 16 de maio de 2009.

[9] DEITEL. **Java – Como Programar**. 6. ed. São Paulo: Pearson Education Brasil, 2005, 1152 p.

[10] DOBGENSKI, J. **Programação Linear para Aproximação de Funções Aplicada ao Projeto de Filtros Digitais**. 1997. 87 f. Dissertação Mestrado em Engenharia Elétrica, UNICAMP, Campinas, São Paulo.

[11] ECLIPSE. **Eclipse Ganymede**. Disponível em: < http://www.eclipse.org/ganymede/ > Acesso em: 15 de maio de 2009.

[12] EDMONDS, B.; HERNÁNDEZ, C.; TROITZSCH, K. **Social Simulation – Technologies, Advances, and New Discoveries**. New York: IGI Global, 2008.

[13] FREITAS, P. J. **Introdução à Modelagem e Simulação de Sistemas com Aplicações Arena**. 2ª Edição. Florianópolis: Visual Books, 2008.

[14] GILBERT, S.; MCCARTY, B. **Mitchell Waite Signature Series: Object-Oriented Design in Java**. California: Waite Group Press, 1998.

[15] GNUPLOT. **GNUPlot: An Interactive Plotting Program**. Disponível em: <http://www.gnuplot.info/docs/gnuplot.html> Acesso em: 16 de maio de 2009.

[16] HOLLAND, John H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**. The MIT Press, 1992.

[17] HOW STUFF WORKS. **What is a Wolf Pack Mentality? – Pack Behavior**. Disponível em: <http://animals.howstuffworks.com/mammals/wolf-pack-mentality1.htm> Acesso em: 10 de maio de 2009.

[18] INTERNATIONAL WOLF CENTER. **Learn – Basic Wolf Information: Communication**. Disponível em: <http://www.wolf.org/wolves/learn/basic/biology/communication.asp> Acesso em: 10 de maio de 2009.

[19] INTERNATIONAL WOLF CENTER. **Learn – Basic Wolf Information: Wolf Predation on Ungulates**. Disponível em: <http://www.wolf.org/wolves/learn/basic/biology/communication.asp> Acesso em: 10 de maio de 2009.

[20] JFREE. **JFreeChart**. Disponível em: < http://www.jfree.org/jfreechart/> Acesso em: 16 de maio de 2009.

[21] KENNEDY, J.; EBERHART, R. C. **Particle Swarm Optimization**. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995.

[22] KINNY, D.; GEORGEFF, M. **Commitment and effectiveness of situated agents**. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, Australia, 1991, p. 82–88.

[23] LAURENCE, S.; TSOI, A. C.; BACK, A. D. **Function Approximation with Neural Networks and Local Methods: Bias, Variance and Smoothness**. In: Australian Conference on Neural Networks – ACNN, 1996, Australian National University, p. 16 – 21.

[24] LIFE COEX. **Melhorar a coexistência dos grandes carnívoros com as actividades agrícolas: O Lobo**. Disponível em: <http://www.life-coex.net/Downloads/portoghese/o%20lobo.pdf> Acesso em: 09 de maio de 2009.

[25] NET LOGO. **Wolf Sheep Predation**. Disponível em: <http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation(docked)> Acesso em: 10 de maio de 2009.

[26] LIMA NETO, F. B. **Research Project – *WPA***. Portal do Professor Fernando Buarque. Disponível em: <http://www.fbln.pro.br/htm_content/pesquisa_wpa.htm> Acesso em: 21 de março de 2009.

[27] PITA, M. R. S.; LIMA NETO, F. B. **Impact of Structuring Elements on Agents' Behavior in Social Simulations**. In: 2009 IEEE Symposium on Intelligent Agents (IA2009). Nashville, United States of America, 2009, p. 106-113.

[28] PITA, M. R. S.; LIMA NETO, F. B. **Simulations of Disease Dissemination Using the Vidya Multi-Agent Systems Platform**. In: 1st Brazilian Workshop on Social Simulation (BWSS2008). Salvador, Brazil, 2008, p. 109-120.

[29] ROVITHAKIS, G. A.; CHALKIADAKIS, I.; ZERVAKIS, M. E. **High-order neural network structure selection for function approximation applications using genetic algorithms**. In: Systems, Man, and Cybernetics, Part B: Cybernetics. 2004, p. 150 – 158.

[30] RUSSELL, S. & NORVIG, P., **Artificial Intelligence: A Modern Approach**. *Prentice Hall,* 2003.

[31] KIP SIEMENS. **Ant Behavior**. Disponível em: <http://www.colostate.edu/Depts/Entomology/courses/en507/papers_1995/siemens.html> Acesso em: 16 de junho de 2009.

[32] SOYATEC OPEN SOLUTION COMPANY. **eUML 2**. Disponível em: <http://www.soyatec.com/euml2/> Acesso em: 16 de maio de 2009.

[33] SUGUMARAN, V. **Application of Agents and Intelligent Information Technologies**. Hershey: IGI Global, 2007, p. 392.

[34] SUN MICROSYSTEMS. **Annotations**. Disponível em: <http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html> Acesso em: 16 de maio de 2009.

[35] SUN MICROSYSTEMS. **Code Conventions for the JAVA™ Programming Language**. Disponível em: <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html> Acesso em: 15 de maio de 2009.

[36] VERBEECK, K.; ET AL. **Learning to Reach the Pareto Optimal Nash Equilibrium as a Team**. In: Lecture Notes in Computer Science (LNCS) – AI 2002: Advances in Artificial Intelligence. Berlin, Germany, 2002, p. 407 – 418.

[37] WEISS, G. **Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence**. Cambridge: *The MIT Press*, 1999.

[38] WOLF COUNTRY. **The Wolf Pack**. Disponível em: <http://www.wolfcountry.net/information/WolfPack.html> Acesso em: 09 de maio de 2009.

[39] WOLF SA – THE LUPUS FUNDATION. **Wolf Behavior**. Disponível em: <http://www.wolfsa.org.za/content_behaviour.htm> Acesso em: 10 de maio de 2009.

[40] WOOLDRIDGE, M. & JENNINGS, N., **Intelligent Agents: Theory and Practice**. The Knowledge Engineering Review, **1995**, 10, 115-152.

Apêndice A

PASSO-A-PASSO DO ALGORITMO

Este apêndice ilustrará o passo-a-passo de cada iteração do algoritmo desenvolvido aplicado à função tangente hiperbólica. Em seguida, serão apresentados os resultados obtidos por cada época da execução do algoritmo aplicado à função cosseno.

Os valores para os parâmetros e constantes do sistema utilizados nas simulações de ambas as funções são os mesmo definidos pela Tabela 1, exceto os seguintes parâmetros que possuem valores específicos:

- EMM – *Erro Médio Mínimo para o lobo*: 0,5 unidades cartesianas;
- DPM – Desvio-Padrão Mínimo para o lobo: 0,5 unidades cartesianas; e,
- Número Inicial de Lobos em cada lado da floresta: 5% das árvores.

Uma vez entendida a configuração do ambiente na qual o algoritmo foi executado, será detalhado como é feita a inicialização do sistema. Em primeiro lugar, determina-se o intervalo no qual a nuvem de pontos está distribuída. Em seguida, as ovelhas são distribuídas – igualmente espaçadas – dentro desse intervalo, distante das árvores por 2 unidades cartesianas.

Os lobos também são igualmente distribuídos dentro do intervalo definido pela nuvem de pontos, de forma que todos eles possuam um mesmo valor inicial para o raio. Os lobos estão distantes das ovelhas por uma unidade cartesiana.

A seguir, serão apresentadas as Figuras relativas às iterações realizadas na 5ª época de execução do algoritmo para a função cosseno para os lobos abaixo da floresta. É possível perceber como as ovelhas buscarão sempre a extremidade da área de atuação do lobo. Adicionalmente, para cada passo será mostrado o operador escolhido pelo lobo e o motivo dessa escolha.

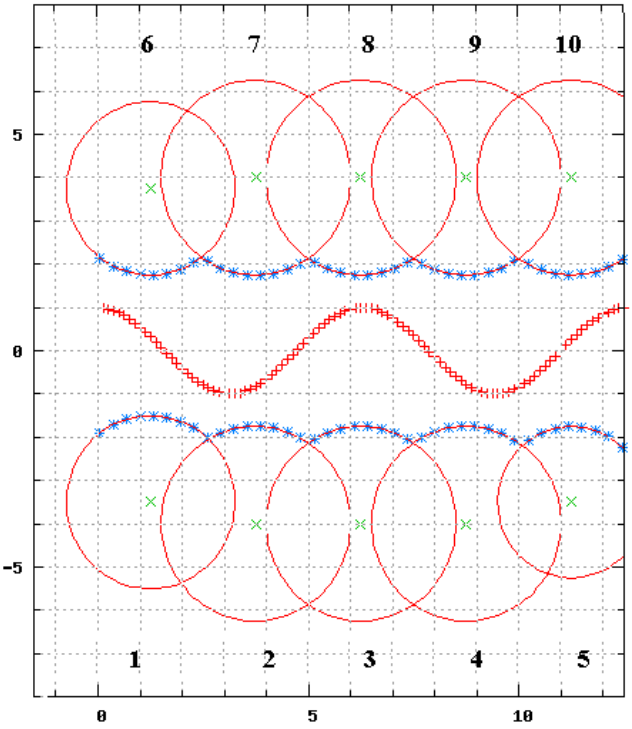

Figura 23. 1ª Iteração: movimentação do lobo 1, pois tanto o EMM quanto o DPM foram maiores do que 0,5.

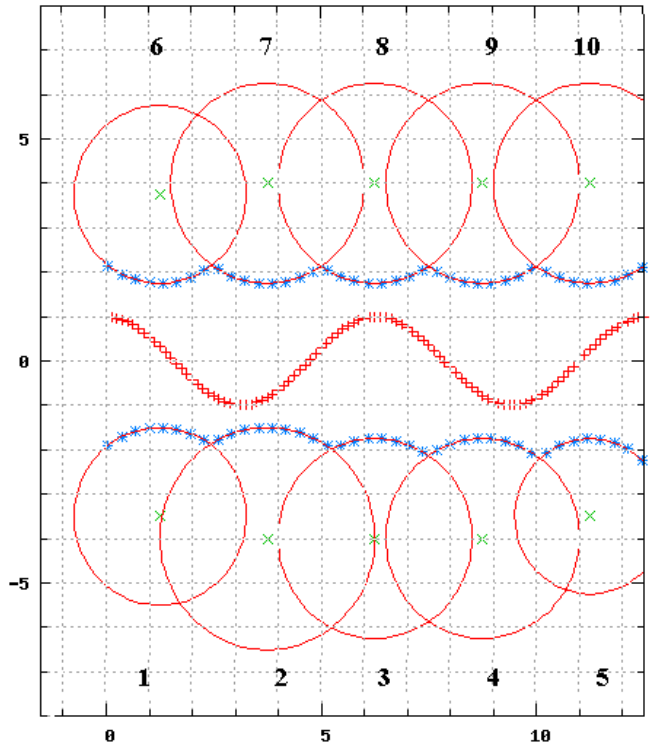

Figura 24. 2ª Iteração: ajuste do raio do lobo 2, pois o EMM foi maior do que 0,5 e o DPM foi menor do que 0,5.

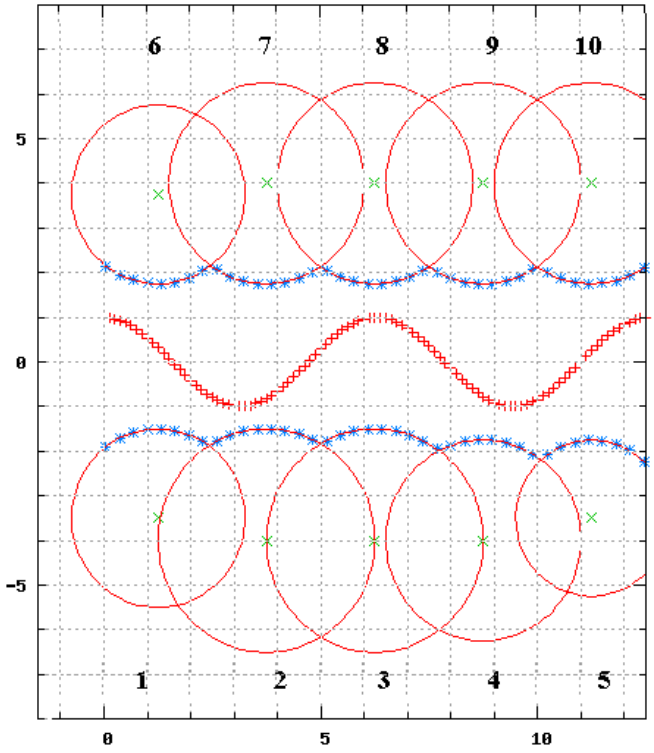

Figura 25. 3ª Iteração: ajuste do raio do lobo 3, pois o EMM foi maior do que 0,5 e o DPM foi menor do que 0,5.

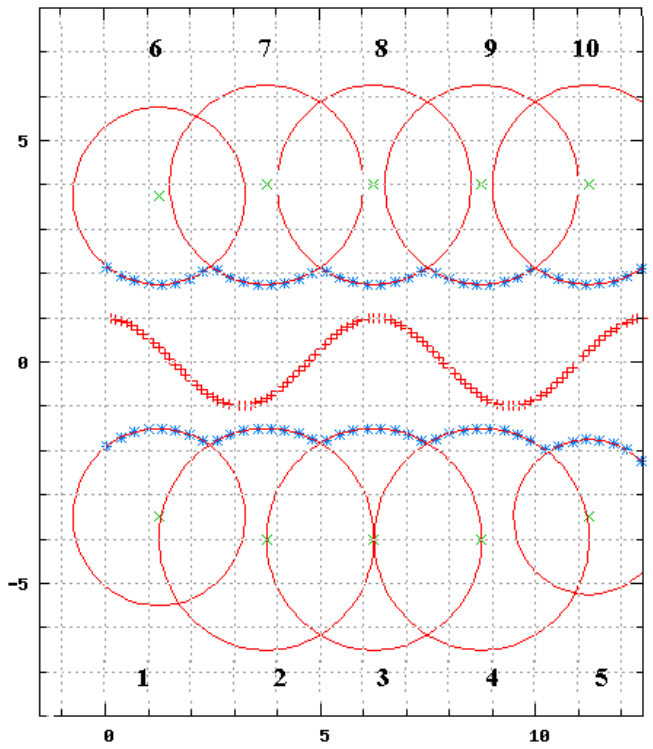

Figura 26. 4ª Iteração: ajuste do raio do lobo 4, pois o EMM foi maior do que 0,5 e o DPM foi menor do que 0,5.

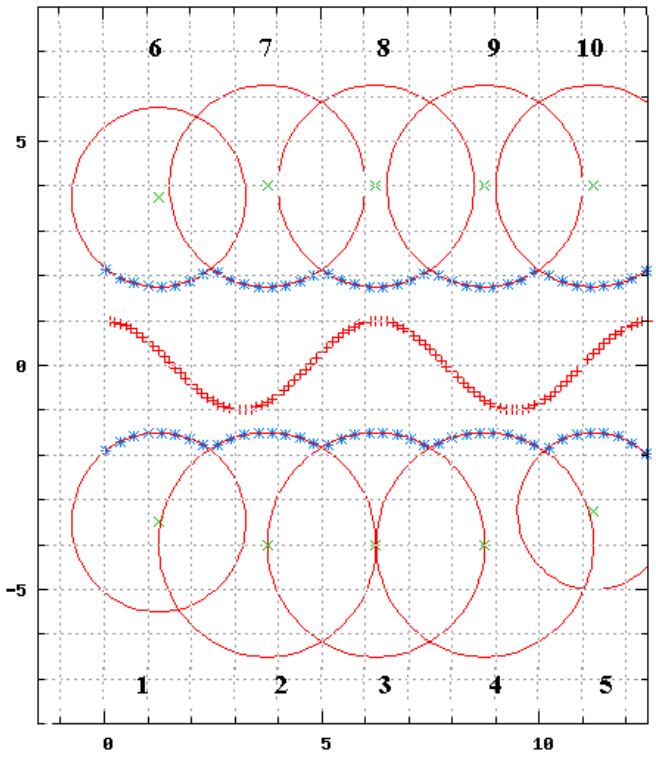

Figura 27. 5ª Iteração: movimentação do lobo 5, pois tanto o EMM quanto o DPM foram maiores do que 0,5.

A seguir, serão apresentados os estados finais de cada época da execução do algoritmo referente à função cosseno, conforme definida na seção 4.2.3. Através deles, é possível observar os operadores utilizados por cada lobo em cada época e como a ação realizada irá colaborar para a diminuição do erro médio global. O valor do erro médio global, assim como o do desvio-padrão global, pode ser obtido na Tabela 5, referente à Configuração 1. As Figuras a seguir ilustram o estado final de cada época, até a convergência do algoritmo (11ª época).

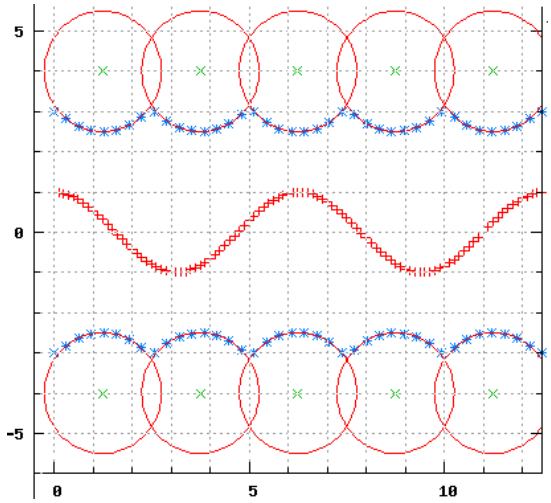

Figura 28. Estado final da 1ª época de execução para a função cosseno.

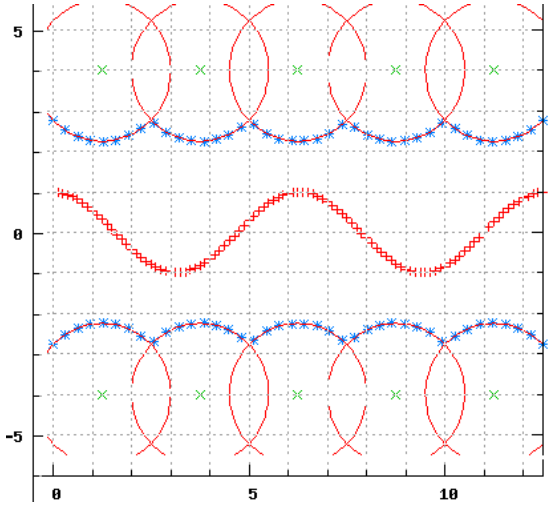

Figura 29. Estado final da 2ª época de execução para a função cosseno.

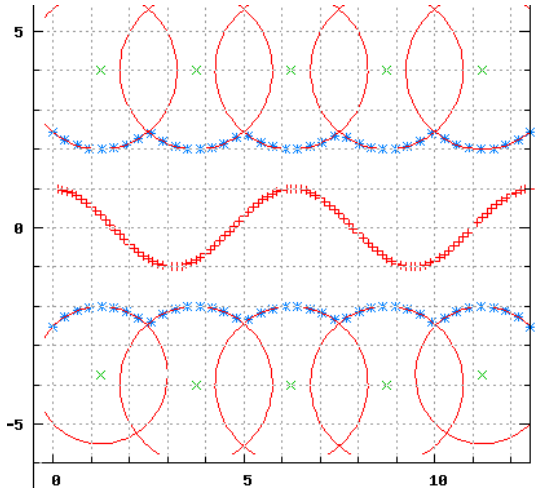

Figura 30. Estado final da 3ª época de execução para a função cosseno.

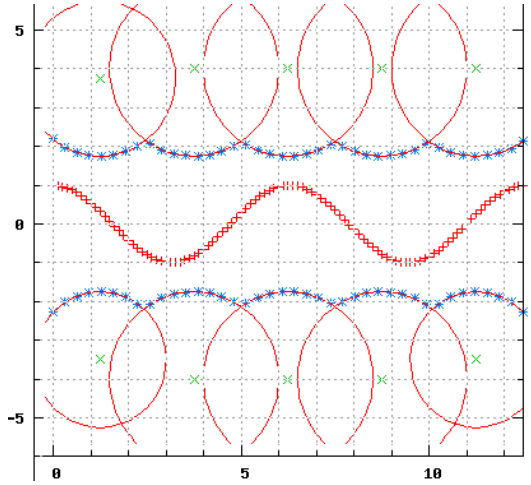

Figura 31. Estado final da 4ª época de execução para a função cosseno.

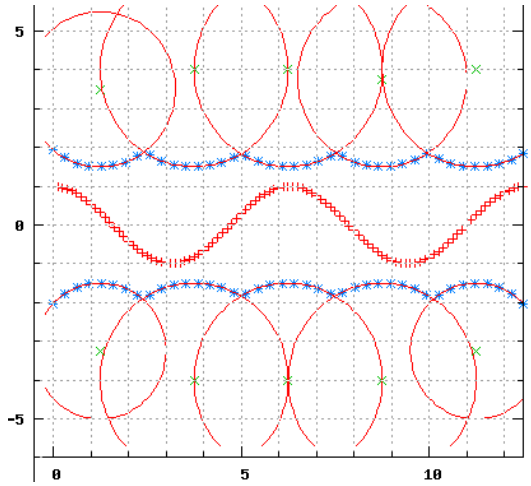

Figura 32. Estado final da 5ª época de execução para a função cosseno.

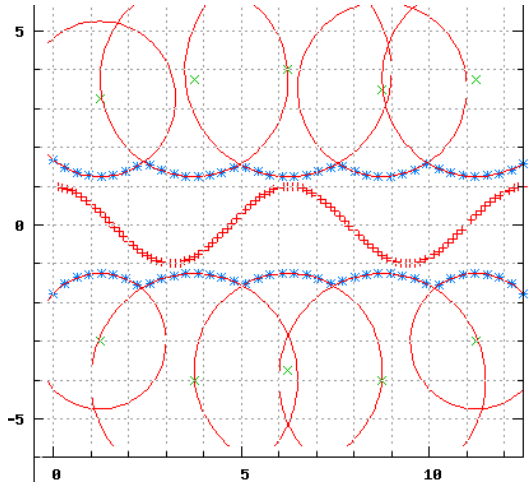

Figura 33. Estado final da 6ª época de execução para a função cosseno.

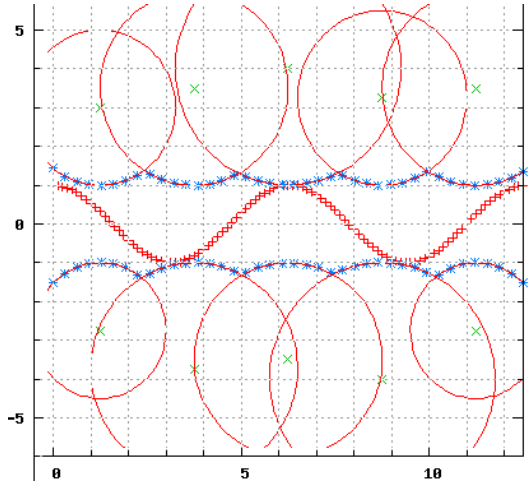

Figura 34. Estado final da 7ª época de execução para a função cosseno.

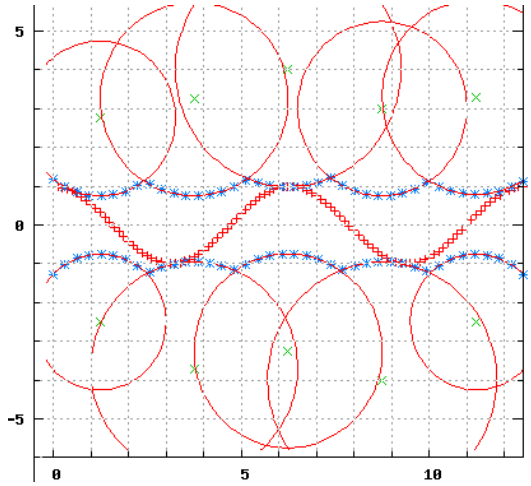

Figura 35. Estado final da 8ª época de execução para a função cosseno.

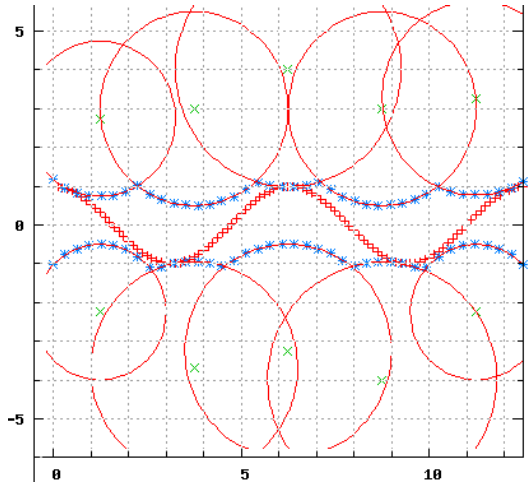

Figura 36. Estado final da 9ª época de execução para a função cosseno.

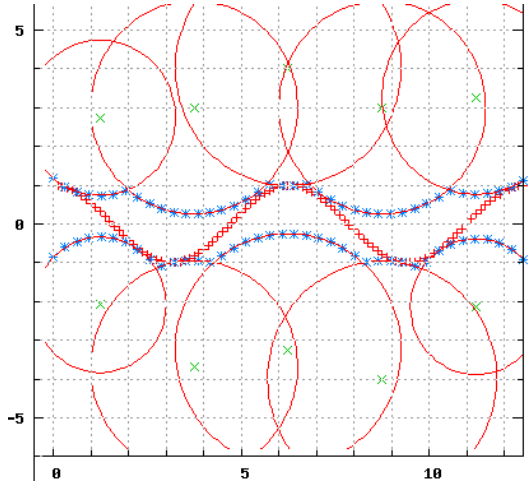

Figura 37. Estado final da 10ª época de execução para a função cosseno.

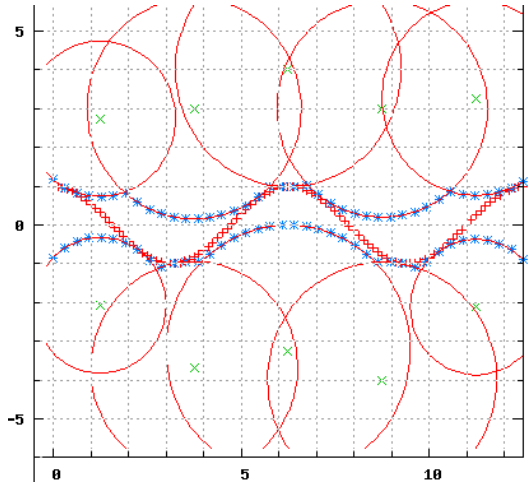

Figura 38. Estado final da 11ª época de execução para a função cosseno.

Apêndice B

DIAGRAMA DE CLASSES UML

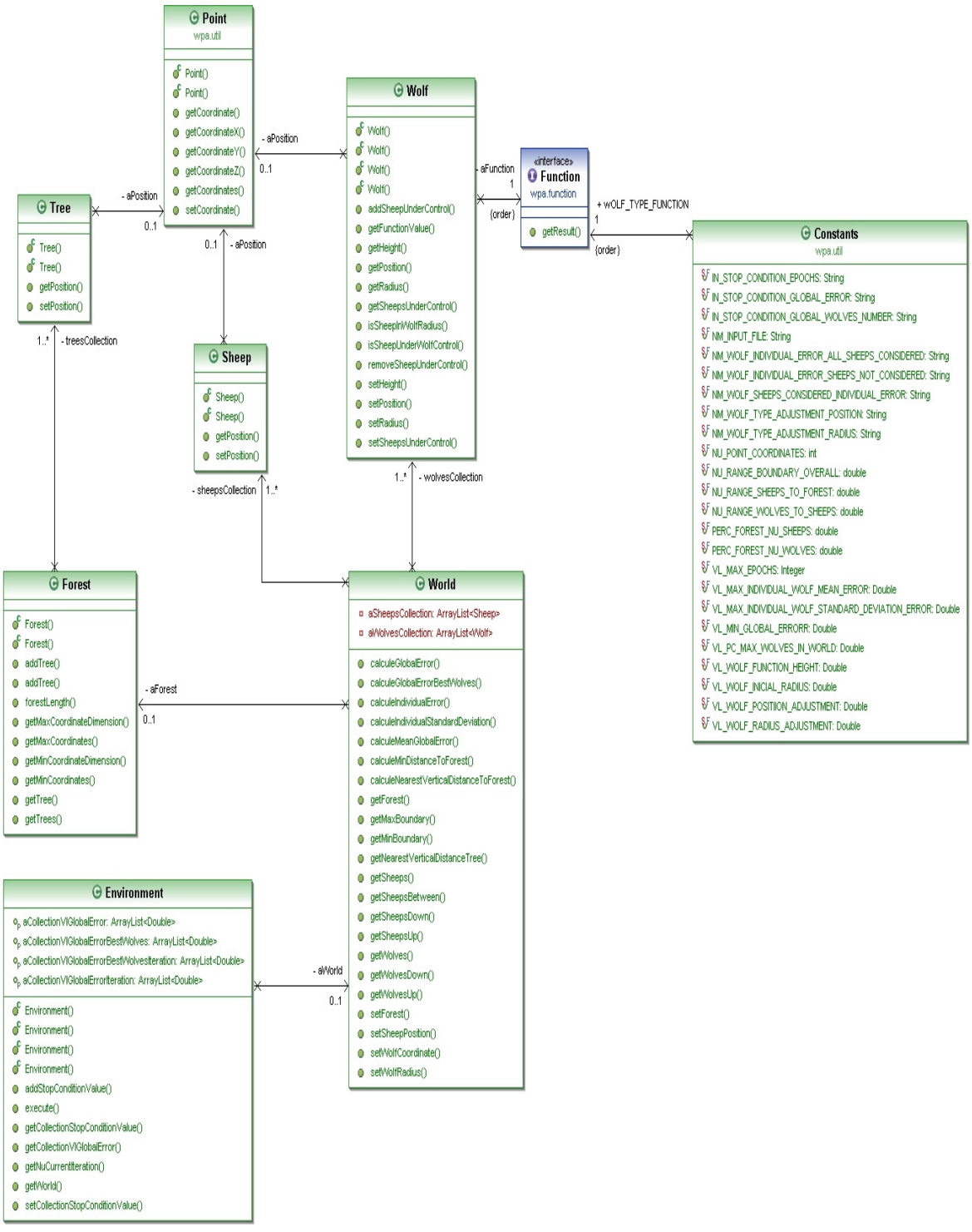

Figura 39. Diagrama de Classes dos principais componentes do sistema.