Calibração de Modelos Complexos utilizando Agentes Inteligentes e Metodologias de Optimização

Pedro Ricardo da Nova Valente

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Informática
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Janeiro de 2009

Calibração de Modelos Complexos utilizando Agentes Inteligentes e Metodologias de Optimização

Pedro Ricardo da Nova Valente

Licenciado em Engenharia da Comunicação (ramo Sistemas de Informação) pela Faculdade de Ciências e Tecnologia da Universidade Fernando Pessoa

Dissertação realizada no âmbito do Mestrado em Engenharia Informática da Faculdade de Engenharia da Universidade do Porto, orientada pelo Professor Luís Paulo Reis e co-orientada pelo Mestre António Pereira.

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Informática
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Janeiro de 2009

Resumo

Nesta dissertação estuda-se a problemática em torno da calibração de modelos, em ambientes de simulações ecológicas, nomeadamente, simulações aquáticas.

Na implementação de um modelo matemático, o primeiro conjunto de simulações é geralmente planeado de forma a testar a lógica interna do modelo. Quando esta tarefa é concluída, é necessário calibrar o modelo, realizando um segundo conjunto de simulações para ajustar os parâmetros do modelo de modo a reproduzir adequadamente os resultados observados. Os parâmetros regulam o comportamento das equações que descrevem as variáveis ao longo do tempo e do espaço, bem como as suas interacções. Geralmente, há alguma incerteza quanto ao valor de cada parâmetro.

A solução proposta nesta dissertação baseia-se na utilização de metodologias de optimização, para parametrização de modelos ecológicos, usando a plataforma de simulação Multi-Agente EcoSimNet, desenvolvida no projecto ABSES – "Agent Based Simulation of Ecological Systems". Um modelo diz-se calibrado, quando os resultados de simulação são similares aos resultados obtidos, nas mesmas características do mundo real. A partir desse ponto, o simulador conseguirá realizar simulações com um grau de confiança acrescido.

Com essa finalidade, foi desenvolvido uma aplicação intitulada "Agente Calibrador" que faz uso de três métodos de optimização: Subida em Colina, Arrefecimento Simulado e Algoritmos Genéticos. Apesar do uso destes métodos, o agente calibrador foi desenvolvido de forma a ser possível a adição de mais métodos de optimização com características diferentes. O agente interage com o simulador EcoDynamo através de mensagens ECOLANG - linguagem de comunicação ecológica.

O trabalho de dissertação inclui um conjunto de testes em dois modelos ecológicos: Predador-Presa e baía de Sangoo. O primeiro modelo, predador-presa, permitiu testar os métodos de optimização, devido à simplicidade em termos de modelo e ao reduzido número de interacções entre as duas entidades presentes. Por sua vez, o modelo da baía de Sangoo, sendo mais complexo, permitiu estudar o desempenho e resultados dos métodos de optimização, sem uso de metodologias de aprendizagem automática; os

resultados obtidos são indicadores da importância da pré-análise de sensibilidade dos parâmetros, como forma de garantir boas soluções dentro de um tempo aceitável. Os algoritmos testados, apesar da sua simplicidade de implementação, provaram serem uma boa ferramenta de auxílio para o processo de calibração.

Abstract

This dissertation studies the issues concerning the calibration of simulations of ecological environments, particularly aquatic simulations.

In the implementation of a mathematical model, the first set of simulations is usually planned in order to test the internal logic of the model. When this task is completed, it is necessary to calibrate the model, conducting a second set of simulations to adjust the parameters of the model in order to adequately reproduce the observed results. The parameters govern the behaviour of the equations that describe the variables over time and space and their interactions. Generally, there is uncertainty about the value of each parameter.

The solution proposed in this dissertation is based on the use of optimization methodologies, for the parameterisation of ecological models, using EcoSimNet Multi-agent simulation platform developed at project ABSES – “Agent Based Simulation of Ecological Systems”. A model is considered to be calibrated, when the results of simulation are similar to the results obtained in the same characteristics in the real world. From that point, the simulator can perform simulations with a greater degree of confidence.

It was developed an application for this task, called Calibration Agent, which makes use of three optimization methods: Hill Climbing, Genetic Algorithms and Simulated Annealing. Despite the use of these three methods, the application was developed in order to be able to add more optimization methods with different characteristics. The agent interacts with the EcoDynamo simulator through messages that follow the ECOLANG format - ecological language of communication.

The work of dissertation ends with a series of tests in two ecological models: predator-prey and Sangoo Bay. The first model, predator-prey allowed to test the methods of optimization, because of the simplicity of the model and the small number of interactions between these two entities. The Sangoo Bay model, is much more complex in entities and relationships, and allowed to study the performance and results of the optimization methods, without the use of learning techniques, as well as to take a set of

conclusions about those results. The major experience conclusions, use of parameter sensibility for understand parameters values boundaries, and its weight into evaluation formula. The use of the traditional optimization methods describe before, permit to achieve a good solution, within an acceptable time. This tool helps human expert into model calibration process.

Aos meus pais,

Agradeço pelo que sou hoje e o que poderei ser amanhã.

Agradecimentos

Na elaboração deste trabalho foram vários os que contribuíram para que fosse possível atingir o fim desta dissertação.

Em primeiro lugar, agradecer ao meu orientador Professor Doutor Luís Paulo Reis pela proposta deste tema, pela paciência e sapiência, que demonstrou ao longo deste período.

Não podia deixar um agradecimento especial ao António Pereira, que desde o início evidenciou grande profissionalismo, empenhamento e companheirismo ao longo do projecto.

Agradeço ao Professor Pedro Duarte, pelo tempo dispendido na preparação do modelo Predador-Presa para o Simulador EcoDynamo, assim como da introdução ao projecto ABSES.

Aos membros do LIACC (NIAD&R), que proporcionaram as condições necessárias para a realização deste projecto.

A todos os meus amigos, Lara, Raquel, Ricardo, Rui, Susana, Paulo, Duarte entre outros, que sempre me apoiaram nos momentos mais atribulados, orientando-me na direcção correcta.

Aos meus pais que sempre foram e são a minha razão de estar aqui neste momento, e que sempre me apoiaram e incentivaram a ultrapassar as barreiras da vida. Um grande Obrigado!

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

1. Introdução

Ao longo das últimas décadas assistiu-se a um aumento da utilização da modelação matemática em todos os campos da ciência, em relação directa com o rápido progresso dos meios informáticos. Os modelos matemáticos são utilizados em Ecologia teórica e aplicada. Recentemente, uma nova área de investigação tem vindo a emergir, resultante da aplicação de técnicas da área da Inteligência Artificial (IA) tais como a Aprendizagem e os Agentes Autónomos às Ciências do Ambiente, conforme referido em diversos “Workshops” internacionais.

Na criação de um novo modelo matemático, o primeiro conjunto de simulações é geralmente estruturado a testar a lógica interna do modelo. Quando esta tarefa está concluída, é necessário calibrar o modelo, realizando um segundo conjunto de simulações, para ajustar os parâmetros do modelo, reproduzindo adequadamente os resultados observados. Os parâmetros regulam o comportamento das equações que descrevem as variáveis ao longo do tempo e do espaço, bem como as suas interacções. Normalmente, existe alguma incerteza quanto ao valor de cada parâmetro. Como consequência, o processo de calibração pode ser longo e trabalhoso, requerendo compreensão dos efeitos dos parâmetros nas variáveis (causa-efeito).

Após a calibração do modelo, é necessário realizar outro conjunto de simulações, validando o modelo, com valores observados que não tenham sido utilizados na calibração. Uma vez validado o modelo, define-se simulações, em função dos objectivos para os quais o modelo foi desenvolvido, sendo a optimização de soluções, um dos exemplos. Este processo pode ser automatizado, ou semi-automatizado, reduzindo-se desta forma, o tempo dispendido na modelação e consequente validação,

fazendo recurso de metodologias de aprendizagem automática, ou a procura de soluções óptimas, no caso da fase de calibração.

Apesar de existirem no mercado soluções de calibração automática, baseados na geração exaustiva de vectores de parâmetros e utilizando diversas técnicas de convergência, no entanto, requerem um grande número de simulações, não sendo aplicáveis a modelos complexos (com maior número de entidades/relações), que exigem tempos de cálculo elevados. Uma alternativa a este cenário, poderá passar pelo desenvolvimento de ferramentas que simulem os processos de aprendizagem de quem implementa e usa os modelos (peritos no modelo). Na utilização dos modelos para a optimizar soluções, pode ser utilizada uma abordagem semelhante.

Em ambos os casos, a utilização de Agentes Autónomos é uma boa alternativa permitindo ainda introduzir no processo de simulação, de forma natural, o elemento humano cujos processos de raciocínio são muito difíceis de modelar através de metodologias de calibração tradicionais.

1.1 Enquadramento e Motivação

Muitos sistemas, em áreas como produção, a gestão financeira e controlo do tráfego, são simplesmente, demasiado complexos para serem modelados analiticamente, existindo ainda a necessidade de analisar o seu comportamento, optimizando o desempenho. As simulações discretas por eventos, têm sido usadas desde há muito tempo, para testar o desempenho de tais sistemas em uma variedade de condições. O uso de simulações está geralmente associado à necessidade de compreensão, de como determinado sistema se comporta, sob a influência de variáveis de ambiente, e caso sejam alterados, verificar se estes melhoram o desempenho.

O processo de calibração, dentro da simulação de modelos, é uma ferramenta importante no estudo de alterações no sistema, sendo muitas vezes, á recombinação de parâmetros, avaliando se o novo conjunto de valores é mais adequado. Para ajudar a resolver este problema, um grande número de métodos de optimização têm sido desenvolvidos. Estes métodos podem ser usados para encontrar o melhor conjunto de parâmetros para uma determinada simulação.

O processo de optimização de parâmetros de simulação, apesar de muito discutido e investigado, mas que ainda é uma área muito activa e com enorme potencial. Vários algoritmos têm sido desenvolvidos e estudados ao longo dos anos, tais como: aproximação estocástica, arrefecimento simulado e pesquisa tabu. No entanto, na maioria das vezes, estes algoritmos só irão optimizar uma simulação para um determinado cenário estático. Caso o cenário evolua com o tempo ou espacialmente, a optimização de parâmetros provavelmente irá mudar.

1.2 Objectivos

O principal objectivo deste trabalho é o desenvolvimento de um agente de calibração que consiga, automaticamente, calibrar um modelo ecológico de simulação. Para isso tem de escolher, em tempo real, os melhores valores para os parâmetros das equações de simulação dos vários objectos, sem ter conhecimento prévio das equações e das classes envolvidas.

Este agente comunica com uma aplicação de simulação biogeoquímica (EcoDynamo), localizada no mesmo computador ou num computador remoto, utilizando a linguagem de simulação ecológica, ECOLANG (Pereira et al., 2005). Os objectivos específicos relativos ao desenvolvimento do agente incluem:

- Escolher o modelo e verificar se a sua base de dados está preenchida;
- Executar o modelo para recolher informação sobre a interacção entre as diferentes classes do modelo;
- Realizar uma análise relativa à sensibilidade intra-classe (sensibilidade de cada variável a cada parâmetro da própria classe) e extra-classe (sensibilidade de cada variável de cada classe às variações dos parâmetros das classes que a influenciam);
- Iniciar o processo de calibração a partir dos dados calculados, seguindo uma estratégia definida para a escolha de valores de parâmetros;
- Permitir a utilização diversos métodos de optimização (incluindo a subida da colina, arrefecimento simulado e algoritmos genéticos) de modo a calibrar o modelo respectivo;

- Visualizar as mensagens trocadas com a aplicação de simulação EcoDynamo;
- Guardar as opções tomadas e os dados que as sustentam;
- Monitorizar o seu próprio processo de resolução do problema.

Além disso, deverá permitir ao utilizador visualizar toda a actividade realizada no âmbito das suas acções.

1.3 Estrutura da Dissertação

Esta dissertação encontra-se estruturada em 6 (seis) capítulos dos quais, o primeiro é composto por esta introdução ao trabalho.

O segundo capítulo faz a contextualização á área da simulação ecológica, fazendo a distinção dos diferentes tipos de modelos de simulação e aplicações desenvolvidas para o efeito. Introduz a simulação ecológica como tema central, focalizando-se na simulação hidrodinâmica. Tem como objectivo definir os conceitos básicos inerentes a simulação, realçando as vantagens/desvantagens do seu uso, como sistema de apoio à decisão. De que forma as meta-heuristicas podem auxiliar o perito em modelos ecológicos a calibrar o modelo, aumentando o grau de confiança nos resultados de simulação. Neste capítulo apresenta-se resumidamente os principais métodos utilizados na calibração de simulações ecológicas, dando realce a: Subida em Colina (*Hill-Climbing*), Arrefecimento Simulado (*Simulated Annealing*) e aos Algoritmos genéticos (*Genetic Algorithms*).

O terceiro capítulo apresenta a plataforma de Simulação EcoSimNet, como plataforma para suportar um sistema multi-agente inteligente (SMA). Caracteriza o simulador EcoDynamo como núcleo de toda a plataforma pelos seus inputs e outputs disponíveis, assim como a linguagem de comunicação ECOLANG entre produtos de software/agentes inteligentes. O capitulo termina explicando como se adiciona um novo agente inteligente á plataforma, realçando as características de reutilização de código assim como a facilidade de integração de novas funcionalidades de simulação.

O quarto capítulo descreve o projecto e implementação do Agente de calibração integrado num sistema multi-agente de simulação de ecossistemas costeiros complexos,

sendo neste caso específico, usada a plataforma de Simulação *EcoSimNet*. Além da definição da arquitectura e tecnologia inerente ao projecto, explica as implementações dos três métodos de calibração descritos no terceiro capítulo, com as respectivas modificações para melhoramento do desempenho.

O quinto capítulo apresenta o ambiente de teste do agente calibrador, explicando de forma sucinta os dois modelos de simulação testados (Predador-Presa e baia de Sangoo), suas características morfológicas, realçando a complexidade inerente em termos de calibração. Para cada modelo foram realizadas baterias de 10 simulações para cada método de calibração, comparando resultados em termos de números de soluções criadas, melhores soluções.

O último capítulo, apresenta as conclusões e perspectivas de desenvolvimento, tecendo os comentários globais de todo o projecto desenvolvido e, projectando caminhos a seguir na optimização de parâmetros de simulação ecológica.

Capítulo 2

2. Simulação Ecológica

Quando nos referimos ao conceito *Simulação Computacional*, este está intrinsecamente ligado ao uso dos computadores para imitar, simular operações e comportamentos do mundo real. Esse processo normalmente designa-se por *Sistema*, estando associado assumpções sobre o modo de funcionamento. Essas assumpções, que se traduzem por relações matemáticas ou lógicas, constituem o *Modelo* que vai ser usado para extrair a compreensão sobre o sistema correspondente.

Sistema e *Modelo* são dois conceitos relacionados com simulação, pelo que convém definir o seu significado e abrangência no domínio da simulação. Segundo António Brito (Brito et al., 2001), sistema é todo o objecto sobre o qual se pretende realizar um determinado estudo, enquanto um seu modelo é uma qualquer representação desse objecto na qual irá, efectivamente, executar tal estudo.

Schmidt e Taylor, em 1970, definem *Sistema* de forma genérica, como sendo uma colecção de entidades (pessoas ou maquinas), que actuam e interagem conjuntamente para realização de algo lógico. Na prática, o Sistema depende dos objectivos de determinado estudo. Os sistemas podem ser classificados em dois tipos, discretos e contínuos. Um sistema discreto é um sistema onde o estado das variáveis muda de estado em pontos separados de tempo. Um banco é um exemplo de sistema discreto, desde que as variáveis de estado – por exemplo, o número de clientes do banco, se altere quando um cliente novo abre conta ou quando fecha as contas nesse banco. Um sistema contínuo é um em que as variáveis de estado mudam continuamente de valor ao longo do tempo. Um avião realizando um voo é um exemplo de sistema contínuo, desde que as variáveis de estado, tais como o posicionamento e a velocidade sejam alteradas continuamente ao longo do tempo. São poucos os sistemas que são exclusivamente

contínuos ou discretos, pelo que são classificados segundo a característica predominante (Hopcroft et al., 2001).

Durante o tempo de vida de um Sistema, é necessário estuda-lo para ganhar conhecimento sobre as interacções entre os vários componentes, ou prever mudanças de performance face alguma alteração introduzida. Assim, a solução que muitas vezes se adopta é a de construir um Modelo que represente adequadamente o Sistema, para que,

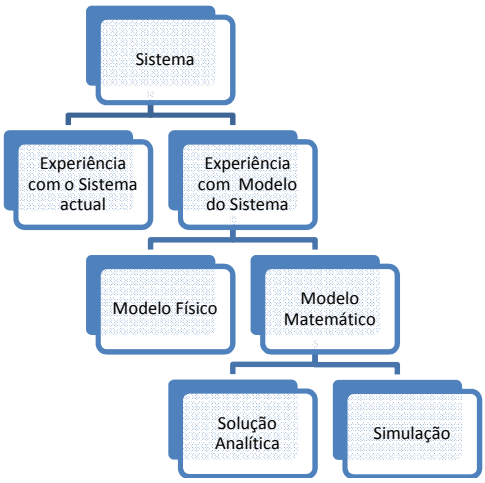

Figura 1. Abordagens de Estudo a um Sistema

sobre ele se possam depois executar os testes. Contudo um modelo, é uma visão de alto nível do sistema, logo não contempla todas as suas características, sendo uma "aproximação" ao sistema real e não o verdadeiro sistema.

A realização de experiencias no sistema real face a experimentação através de uma representação (modelo) do Sistema permite reduzir os custos associados as alterações a realizar fisicamente, assim como ao tempo necessário para a aplicação das alterações (figura 1). É muito dispendioso, e nada prático, construir todas as alternativas possíveis do sistema físico real, até se encontrar uma solução satisfatória. Assim, é possível a elaboração de vários modelos para ser determinado um modelo optimizado.

A modelação do sistema permite controlar melhor o ambiente, assim como torna-se impraticável alterar o sistema actual, sem pôr em causa o seu bom funcionamento. Normalmente os estudos a um sistema existente têm como objectivo a melhoria/optimização do seu desempenho, sendo necessária a criação de um modelo, além da sua compreensão. Por outro lado, os modelos, sendo uma abstracção da

realidade, não garantem fiabilidade completa, sendo a fase de validação, de extrema importância, na modelação, devendo reflectir os objectivos propostos para a tomada de decisão.

Vulgarmente modelo é associado a modelação física, onde podemos ver por exemplo, miniaturas de modelos automóveis em túneis de vento, maquetas de edifícios á escala ou simuladores de avião em cabine de controlo reais. São exemplos de modelos físicos (também designados por modelo icónicos). Contudo a maioria dos modelos são matemáticos, representando o sistema em termos lógicos e quantificando as relações, podendo desta forma fazer as alterações no modelo, observando as suas reacções – caso seja um modelo válido (Eykhoff, 1974).

Após construído o modelo matemático, este deve ser examinado para verificação se os resultados obtidos vão ao encontro das questões estipuladas inicialmente acerca do sistema. Caso o modelo seja simples, é possível trabalhar com as suas relações e quantidades de forma a obter uma solução analítica. Exemplo de um modelo simples matemático, é a determinação da distância ($d = v.t$), em que v é a velocidade e o t é o tempo. Com este modelo, caso tenhamos conhecimento da distância a percorrer e da velocidade, podemos utilizar o modelo para obter o $t = d/v$. Apesar de este modelo ser simples de calcular, existem outros em que as soluções analíticas são complexas, requerendo carga computacionais acrescidas.

Com o aumento das necessidades e complexidade dos sistemas, a validação dos modelos matemáticos torna-se um processo complexo, não podendo as suas soluções ser expressas em termos analíticos. Desta forma, o modelo deve ser simulado, isto é, devem ser observados os resultados (outputs) das fórmulas matemáticas, tendo em conta a performance do sistema, fazendo alterações nas entradas (inputs) do modelo.

Simular, segundo a Webster's Collegiate Dictionary, é "fingir, para obter a essência externa da realidade". De acordo com Schriber (1987), "A simulação envolve a modelação de um processo ou sistema de modo que o modelo imite a resposta do sistema actual para eventos que acontecem com o passar do tempo".

Neste trabalho a simulação será definida como um processo de desenvolver um modelo de um sistema real e experimentação do modelo, com o propósito de conhecer o

comportamento do sistema e avaliar várias estratégias para a sua operação (Pegden et al., 1990).

A simulação é considerada uma ferramenta disponibilizada pela área da investigação operacional que permite a geração de cenários, a partir dos quais se pode: orientar o processo de tomada de decisão, proceder a análises e avaliações de sistemas, propondo melhorias de performance. Normalmente estes procedimentos têm como base parâmetros técnicos e/ou económicos.

Este trabalho irá incidir na simulação de modelos matemáticos, pelo que a partir de agora designaremos simulação de modelos matemáticos por modelos de simulação. Devido ao domínio em estudo serem as simulações ecológicas, a partir deste ponto designaremos por ecossistema ao Sistema em modelação. O próximo capítulo descreve uma classificação de simuladores de modelos baseado em três dimensões: Estatísticos ou Dinâmicos, Determinísticos ou Estocásticos e Contínuos ou Discretos.

2.1 Simulação de Modelos

Apesar do reconhecimento das potencialidades da modelação enquanto ferramenta de suporte à decisão, é necessário ter em conta que, como modelo, é uma representação abstracta da realidade, logo não contém todos os aspectos do sistema. O modelador tem de ter a noção dessa característica, na análise dos resultados da simulação. Os resultados devem ser observados com a mesma abstracção da realidade. Na formulação matemática, um modelo num domínio científico tem cinco componentes (Jorgensen et al., 2001):

- **Variáveis externas ou funções fixas**: variáveis ou funções que são de natureza externa ao sistema, mas que interagem com este, influenciando o estado do ecossistema (sistema);
- **Variáveis de sistema**: como o nome indica, são variáveis que descrevem o estado do ecossistema. A selecção das mesmas são cruciais na estrutura do modelo, mas normalmente a sua escolha é óbvia. Normalmente os resultados da simulação são expressos segundo valores para as variáveis de sistema (outputs);

- **Equações matemáticas**: são usadas para representar os processos biológicos, químicos e físicos. Descrevem as relações entre as funções fixas e as variáveis de sistema. O mesmo tipo de processos pode ser encontrado em diferentes contextos, o que implica que a mesma equação pode ser usada em diferentes modelos. Exemplo disso, são equações para representação das marés.
- **Parâmetros**: são coeficientes na representação matemática dos processos. Podem ser considerados constantes para ecossistemas específicos ou parte de ecossistemas. Um exemplo de parâmetro pode ser a taxa de crescimento de um animal, sendo este valor configurado pelos especialistas do modelo para condizer com os valores reais.
- **Constantes**: são parâmetros de sistema cujos valores não mudam. Por exemplo, o valor da gravidade.

Durante a simulação os valores dos parâmetros são constantes, sendo o seu valor constante durante toda a simulação. Este tipo de abordagem tem sido discutido devido ao facto que no mundo real, em alguns casos, os parâmetros são influenciados ao longo do tempo, pelo que o modelo deverá reflectir tais actualizações durante as simulações (Jorgensen et al., 2001).

Como já foi referido anteriormente, os modelos são definidos em termos matemáticos, formalizando expressões dos elementos essenciais de um determinado problema. O primeiro reconhecimento do problema costuma ser verbal, destacando o essencial a modelar assim como questões pelas quais o simulador deverá responder. Existem três fases a ter em conta aquando da criação de um novo modelo, a saber:

- **Verificação**: teste da lógica interna do modelo. Questões como - o modelo reage de forma esperada? É o modelo estável ao longo do tempo? - são aqui levantadas. A verificação permite validar o comportamento do modelo, ao longo das simulações, sendo a sua duração, em modelos mais complexos, até a próxima fase: calibração;
- **Calibração**: esta fase é dedicada a encontrar as melhores combinações de valores dos parâmetros de forma a diminuir a variação entre os resultados simulados e os observados no mundo real. Este processo costuma ser por tentativa/erro, realizado por especialistas em modelação e do domínio de

aplicação (neste caso simulações ecológicas). É um processo moroso em termos de tempo, pelo que cada modelo deve ser calibrado afim de se poder extrair algum tipo de conhecimento, aumentando o grau de confiança nos resultados produzidos. É nesta fase que este projecto recai, tentando automatizar o processo de calibração, tendo como base alguns dados do modelo a simular.

- **Validação:** deve ser distinguida da verificação. Enquanto a verificação estuda os comportamentos das diversas entidades representadas no modelo, a validação preocupa-se com a qualidade dos resultados obtidos da simulação. Um modelo diz-se estruturalmente valido, quando se consiga estabelecer a mesma relação causa-efeito do sistema real, com um grau de exactidão considerável. A escolha de funções de avaliação depende dos objectivos para o modelo em simulação, mas normalmente realizam-se os desvios entre os resultados simulados e os observados no sistema real.

Os modelos podem ser classificados de diversas formas, quanto a sua evolução ao longo do tempo – estático ou dinâmico, quanto a exactidão dos seus resultados – determinístico ou estocástico ou quanto aos resultados produzidos – discretos ou contínuos. Cada tipo de modelo é organizado segundo os objectivos que se propõe alcançar, existindo em alguns casos, a necessidade de tratar cada entidade como um indivíduo, que é descrito no modelo com características próprias, em vez de pertencer a um grupo de indivíduos. Este tipo de modelos tem o nome de modelo baseado em indivíduos (IBM – Individual Based Models), sendo uma aproximação aos sistemas multi-agentes inteligentes, da área Inteligência Artificial. Permite ter uma granularidade mais fina quanto aos resultados obtidos, contrapondo com o aumento de relacionamentos entre indivíduos, computacionalmente mais exigente.

2.1.1 Estáticos ou dinâmicos

Denominam-se como modelos estáticos os que visam representar o estado de um sistema em um instante ou que em suas formulações não se leva em conta a variável tempo, enquanto os modelos dinâmicos são formulados para representarem as alterações de estado do sistema ao longo da contagem do tempo de simulação.

O algoritmo de *Monte Carlo* é um exemplo de modelo estático, tendo as suas origens e primeiras simulações, durante a segunda guerra mundial (Halton, 1970), onde foi aplicado a problemas relacionados com o desenvolvimento da bomba atómica. Este modelo consegue percorrer milhares de cenários e gerar hipóteses tendo em conta a aleatoriedade apenas.

2.1.2 Determinísticos ou Estocásticos

Se o modelo de simulação não contém nenhuma componente probabilística associada, é chamado de modelo determinístico. Um modelo que descreva uma reacção química pode ser considerado como modelo determinístico, visto ser constituído por um conjunto de equações diferenciais matemáticas, cujos resultados não dependem de probabilidades. Nos modelos determinísticos, os resultados são determinados pelo conjunto de valores nos inputs e pelos relacionamentos que estão especificados no modelo, apesar de consumirem bastante tempo de computação na avaliação do modelo. Alguns sistemas, contudo, podem ser modelados com componentes de aleatoriedade, sendo considerados como modelos estocásticos.

Como acontece no mundo real, certos parâmetros de um sistema, só são conhecidos dentro de uma gama de valores e não de forma exacta, podendo por isso apresentar valores diferentes em diferentes instantes do tempo. Exemplo disso é o tráfego de uma dada avenida, o número de carros a circular varia por unidade de tempo, sendo num modelo determinístico considerado um valor médio, que muitas vezes, não permite fiabilidade nas respostas do modelo. Seria necessário substituir esse valor médio por uma distribuição estatística do fluxo de automóveis. Sempre que um modelo entra em conta com este aspecto de flutuação, deixa de ser considerado determinístico para tomar a designação de estocástico.

Apesar desta divergência de definição entre os dois modelos, em muitos casos o modelo é considerado determinístico, pois as suas regras internas são bem determinadas e recorrentes de uma matemática que é determinística. Por isso é mais correcto afirmar que o processo de simulação é estocástico, em vez de apelidar como modelo estocástico.

Mas como introduzir num modelo determinístico o comportamento estocástico? Pode ser aplicado distribuição de valores (realizada por meio de experimentação no sistema) ou através de funções adaptadas ao conjunto dos dados experimentais, tais como funções de probabilidade típicas: distribuição *Normal*, distribuição de *Poisson*, entre outras.

Modelos de simulações estocásticas produzem resultados que por si só, são aleatórios, e devem ser considerados como estimativas das verdadeiras características do modelo. É uma das desvantagens deste tipo de simulação.

2.1.3 Contínuos ou Discretos

São modelos discretos em que o avanço da contagem de tempo na simulação se dá na forma de incrementos cujos valores podem ser definidos em função da ocorrência dos eventos ou pela determinação de um valor fixo, nesses casos só é possível determinar os valores das variáveis de estado do sistema nos instantes de actualização da contagem de tempo; enquanto para os modelos contínuos o avanço da contagem de tempo na simulação dá-se de forma contínua, o que possibilita determinar os valores das variáveis de estado a qualquer instante.

Podemos considerar novamente o exemplo do tráfego automóvel. A decisão de usar modelos discretos ou contínuos depende dos objectivos estipulados para o estudo do sistema. Caso estejamos a falar numa auto-estrada, devemos considerar o modelo discreto, caso as características e os movimentos dos automóveis individuais serem importantes. Alternativamente, se os automóveis podem ser tratados como um conjunto, a orientação dos mesmos, pode ser descrita por equações diferenciais no modelo contínuo.

2.1.4 Modelos baseados nos Indivíduos (IBM)

Modelos baseados nos indivíduos (IBM - Individual-based models) são simulações baseadas nas consequências globais das interacções locais dos indivíduos de uma população. Esses indivíduos podem representar plantas e animais de um ecossistema, veículos automóveis no trânsito, pessoas em multidões, ou personagens autónomas em

animações e jogos. Estes modelos tipicamente consistem num ambiente ou Framework no qual as interacções ocorrem, sendo os seus indivíduos definidos em termos de comportamentos, procedimentos e parâmetros (Grimm et al., 2005).

Neste tipo de modelos, as características de cada indivíduo é registado ao longo do tempo. Esta característica contrapõe os pressupostos das técnicas de modelação, em que as características da população é uma média das características dos indivíduos e o modelo tenta simular as alterações dessas características médias para toda a população. Os modelos baseados nos indivíduos são também conhecidos como entidades ou modelos baseados em agentes, assim como simulações baseadas em indivíduos/entidades/agentes.

Alguns modelos baseados em indivíduos contêm também conhecimento explícito sobre a localização no espaço geométrico dos indivíduos. Esse conhecimento individual confere aos modelos deste tipo, a mobilidade, onde os indivíduos podem mover-se dentro do ambiente. Podemos considerar um modelo natural, por exemplo, de um animal numa simulação ecológica. Onde na mesma simulação as plantas não possuem a mobilidade dos animais. Alguns modelos baseados em indivíduos não carecem de conhecimento espacial, por exemplo, uma simulação de uma rede de computadores pode ser baseada em modelos individuais de redes de computadores, sendo a sua localização irrelevante para os propósitos da simulação. Os modelos com conhecimento espacial usam domínios de valores contínuos (valores reais) ou discretos (valores inteiros em forma de grelha).

Os sistemas de simulação baseados em indivíduos podem ser associados a sistemas multi-agente onde os agentes neles contidos não contemplam estruturas simbólicas complexas de representação do ambiente e das entidades nele representadas. Estes sistemas tiveram a sua origem nos chamados autómatos celulares cuja componente fundamental era a célula posicionada numa determinada localização espacial e que evoluiu para o conceito corrente de indivíduo posicionado no espaço, num determinado ambiente de simulação.

2.2 Simulação Hidrodinâmica

A hidroinformática desempenha um papel cada vez mais importante na optimização de recursos e gestão de ecossistemas aquáticos, devendo a sua utilização, ser baseado no conhecimento dos ecossistemas em termos de funcionamento.

A aquisição deste conhecimento tem de ter por base uma observação sistemática dos processos considerados relevantes, a qual, realisticamente, só poderá ser efectuada num pequeno número de pontos. O recurso aos modelos matemáticos permite, por um lado, integrar a informação recolhida num número reduzido de pontos, extrapolando essa informação para todo o sistema e, por outro lado, efectuar previsões sobre possíveis comportamentos do sistema em função de eventuais alterações das condições ambientais.

Nesta perspectiva, a optimização da gestão de ecossistemas aquáticos deve assentar num sistema de monitorização o qual deverá englobar três componentes: medidas, modelação, publicação de dados/resultados.

O estabelecimento de um programa de medidas é fundamental para, por um lado, fornecer directamente dados sobre parâmetros importantes do ecossistema e, por outro lado, calibrar e validar os modelos. Este processo de calibração e validação resulta assim num processo dinâmico que vai sendo enriquecido à medida que vai existindo mais informação disponível.

Os modelos desempenham então um papel importante, tanto no que respeita ao estabelecimento de diagnósticos sobre os problemas do ecossistema, através da integração e da correlação dos diversos parâmetros envolvidos, como no que respeita à execução de prognósticos quer sobre eventuais medidas remediadoras quer sobre os possíveis efeitos da alteração das variáveis no respectivo funcionamento.

A garantia de um uso eficaz de toda esta informação passa finalmente pela disponibilização de meios eficientes de publicação que tornem fácil e atractivo o respectivo uso.

A rápida evolução, quer ao nível do preço quer das capacidades de processamento, que se tem vindo a verificar ao nível das tecnologias de computação e de aquisição de

dados, permite estabelecer actualmente sistemas de monitorização em tempo real, em que conjuntos de sensores fornecem informação a modelos matemáticos que, por sua vez, simulam o sistema em tempo real, garantindo assim uma observação em contínuo e a possibilidade de actuação imediata sobre eventuais anomalias que sejam detectadas.

2.3 Simuladores Ecológicos Aquáticos

A área costeira sempre desempenhou um papel importante na vida dos seres humanos, sendo geograficamente delimitadora, entre o mar e a terra, providenciando possibilidades de lazer, comércio e serviços básicos para a humanidade.

No último século, assistiu-se a uma migração das populações do interior para o litoral, estando perto de 60% da população mundial localizada até 60km do litoral (Watson et al., 1996). Nas últimas décadas, depois da observação de alguns desastres ambientais, cientistas, políticos, ambientalistas e outros intervenientes, tomaram consciência da necessidade de unir esforços para assegurar uma gestão sustentável da orla marítima, mantendo os níveis de qualidade dos ecossistemas compatíveis com as estratégias de desenvolvimento – normalmente apelidado de gestão para o desenvolvimento sustentável de ecossistemas.

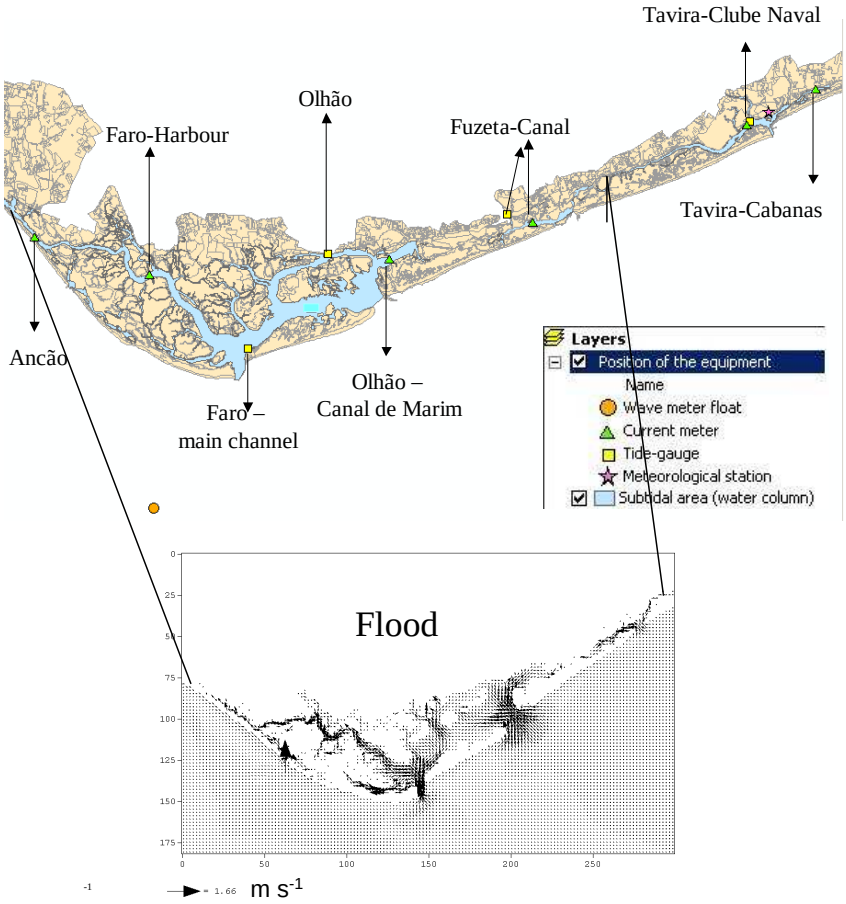

Figura 2.Esquema modelo Ria Formosa (Algarve)

Com a sedimentação populacional no litoral, a aquacultura, o turismo e desenvolvimento urbano interagem com o ecossistema aquático, forçando a movimentação de águas ricas em nutrientes orgânicos e minerais, derivado da agricultura, dos afluentes urbanos e industriais assim como dos esgotos domésticos (Duarte et al. 2007b).

As estratégias levadas a cabo para o desenvolvimento sustentado dos ecossistemas, devem incluir todos os interesses de cada região e devem explicar, com a maior clareza possível, a tomada das decisões e os benefícios que irão cumprir a médio e a longo prazo, para cada ecossistema. Essas estratégias podem incluir objectivos a curto prazo (quando está em causa a reposição do equilíbrio ambiental), mas devem ser elaboradas tendo em conta gerações futuras.

Modelos de ecossistemas aquáticos incluem processos bioquímicos, tais como a fotossíntese, o ciclo de nutrientes assim como o processo de transporte, tornando os

modelos complexos, quer em representação matemática, assim como no processo de calibração dos seus parâmetros. Muitos modelos matemáticos usados na ecologia são baseados em simplificações implícitas ou ambíguas, que nem sempre vão ao encontro das teorias aceites cientificamente. Isto origina resultados incertos, devido à incerteza associada aos parâmetros do modelo, aos valores de entrada e em alguns casos, até na estrutura do modelo (Scholten et al., 1998).

No domínio da modelação de ecossistemas ecológicos, diferentes modelos podem incluir diferentes processos, ou o mesmo processo descrito em diferentes camadas de detalhe. O grau de detalhe é determinado pela importância assumida pelos diferentes processos num determinado ecossistema, assim como pelo conhecimento existente sobre este. Por exemplo, alguns modelos matemáticos usam uma descrição simplificada dos processos de transporte hidrodinâmicos e detalhada sobre os processos ecológicos bênticos (o termo aplica-se ao fundo do mar ou as espécies ai existentes) (Barreta et al., 1988), e outros modelos o oposto – descrição detalhada dos processos hidrodinâmicos e processos ecológicos simplificados (Luyten et al., 1999).

Quando um modelo ecológico é construído, as incertezas e variabilidade mencionadas, são reflectidas na fase de implementação, razão pela qual existe tanto desenvolvimento de modelos, criados por diferentes equipas de investigadores, em todo o mundo. Cada equipa de investigadores, adopta diferentes técnicas de modelação, tal como é visível nas aplicações de simulação ecológica: modelo EMS Dollard descrito por Baretta e Ruardij, 1988; modelo COHERENS descrito por Luyten et al., 1999; ou software orientado a objectos: EcoWin – desenvolvido por Ferreira (1995), MOHID (em linha).

Os modelos baseados em linguagens de programação estruturadas, são constituídos pelo programa principal, onde algumas variáveis de estado descrevem o ecossistema dentro da simulação, fazendo a chamada para as diversas subrotinas, a cada passo de tempo do modelo. As subrotinas representam as entidades e o relacionamento entre elas, em termos de processos, representando o fluxo que influencia cada variável de estado. No fim de cada ciclo de simulação, todas as variáveis de estado são actualizadas como uma função dos fluxos mencionados.

2.4 Métodos de Optimização

No capítulo anterior fez-se referência a importância da existência de algoritmos capazes de estimar valores para os parâmetros dos modelos, de forma a garantir que a simulação seja de acordo com o estipulado para o modelo e para os fins que se propõe.

Os métodos de optimização de simulações são utilizados quando a função objectivo pode ser apenas avaliada fazendo recurso a simulações de computador. Isto acontece porque não existe uma expressão analítica para a função objectivo que consiga descrever todo o modelo, sendo este o resultado da interacção de várias entidades nele representadas, que evoluem ao longo do tempo.

Jorgensen (Jorgensen, 2003) defende que além de bons métodos de optimização de parâmetros dentro do domínio da simulação ecológica, é necessário especial atenção a sensibilidade dos parâmetros, o que quer dizer, que é necessário ter conhecimento prévio de valores estimados de referência. Outra ideia subjacente, é a visão do modelo ecológico como uma abstracção da realidade, logo não representa a 100% o mundo real.

2.4.1 Subida de Colina (Hill-Climbing)

O método de optimização Subida de Colina é uma técnica que pertence à família de métodos de pesquisa local. É relativamente simples de implementar e conduz a soluções de modo extremamente rápido, tornando-se, por estes motivos uma primeira escolha popular. Embora existam algoritmos com maior complexidade, podendo ter melhores resultados, na maior parte das situações este método funciona bem.

O método apresenta problemas relativamente a mínimos locais pois não inclui qualquer método que lhe permita escapar deste tipo de mínimos. Deste modo, a solução encontrada pela subida de colina é tipicamente um mínimo local cuja qualidade pode estar muito distante do óptimo global.

```
Início
  s = GerarSolucaoInicial;
  T = Tinicial;
  Enquanto condição de fim fazer
     s' = NovaSolucao(N(s));
     Se Avaliacao(s') < Avaliacao(s) Entao
          s = s';
Fim
```

Algoritmo 1. Subida de Colina - Genérico

O método Subida de Colina pode ser usado para resolver problemas que têm muitas soluções, algumas das quais são melhores do que outras. Tudo começa com uma solução tipicamente calculada de modo aleatório (potencialmente pobre), e iterativamente faz pequenas alterações para a solução cada vez melhorar um pouco (procura na vizinhança da solução). Quando o algoritmo não consegue encontrar qualquer melhoria, termina. Idealmente, a esse ponto, a solução actual é próxima do ideal, mas não é garantido que a subida da colina chegaria perto da solução óptima.

2.4.2 Arrefecimento Simulado

Arrefecimento simulado ou *Simulated Annealing* é uma meta heurística para optimização que consiste numa técnica de pesquisa local probabilística, e se fundamenta numa analogia com a termodinâmica.

Esta meta heurística é uma metáfora de um processo térmico, dito *annealing* ou recozimento, utilizado em metalurgia para obtenção de estados de baixa energia num sólido. O processo consiste de duas etapas: na primeira a temperatura do sólido é aumentada para um valor máximo no qual ele se funde; na segunda o resfriamento deve ser realizado lentamente até que o material se solidifique, sendo acompanhado e controlado esse arrefecimento. Nesta segunda fase, executada lentamente, os átomos que compõem o material organizam-se numa estrutura uniforme com energia mínima. Isto provoca que os átomos desse material ganhem energia para se movimentarem livremente e, ao arrefecer de forma controlada, dar-lhes uma melhor hipótese de se organizarem numa configuração com menor energia interna, para ter, como resultado prático, uma redução dos defeitos do material.

```
Início
  s = GerarSolucaoInicial;
  T = Tinicial;
  Enquanto condição de fim fazer
      s' = NovaSolucao(N(s));
      Se Avaliacao(s') < Avaliacao(s)Entao
            s = s';
      Senao
            s = CriterioAceitacao(s,s',T);
      ActualizarTemperatura(T);
Fim
```

Algoritmo 2. Arrefecimento Simulado – Genérico

De forma análoga, o algoritmo de arrefecimento simulado substitui a solução actual por uma solução próxima (i.e., na sua vizinhança no espaço de soluções), escolhida de acordo com a função objectivo e tendo em consideração a variável T (dita Temperatura, por analogia). Quanto maior for T, maior a componente aleatória que será incluída na próxima solução escolhida. À medida que o algoritmo progride, o valor de T é decrementado, começando o algoritmo a convergir para uma solução óptima, necessariamente local.

Uma das principais vantagens deste algoritmo é permitir testar soluções mais distantes da solução actual e dar mais independência do ponto inicial da pesquisa. Consegue escapar de mínimos locais dado que permite a movimentação para soluções vizinhas de pior qualidade enquanto a temperatura for elevada.

2.4.3 Algoritmos Genéticos

Um algoritmo genético (AG) é uma técnica de optimização baseada na teoria da evolução. Algoritmos genéticos são uma classe particular dos algoritmos evolutivos que utilizam técnicas inspiradas pela biologia evolutiva tais como a hereditariedade, mutação, selecção natural e recombinação (ou crossing over).

```
Início
  P = GerarPopulacaoInicial;
  Avaliar(P);
  Enquanto condição de fim fazer
      P' = Recombinar(P);
      P" = Mutacao(P');
      Avaliar(P");
      P = Escolha(P" ∪ P);
Fim
```

Algoritmo 3. Algoritmo Genético - Genérico

Os algoritmos genéticos são implementados como uma simulação de computador em que, uma população de representações abstractas da solução é seleccionada para efectuar uma pesquisa de soluções melhores. A evolução geralmente inicia-se a partir de um conjunto de soluções criado aleatoriamente e é realizada por meio de gerações. A cada geração, a adaptação de cada solução na população é avaliada, alguns indivíduos são seleccionados para a próxima geração, e recombinados ou mutados para formar uma nova população. A nova população então é utilizada como entrada para a próxima iteração do algoritmo.

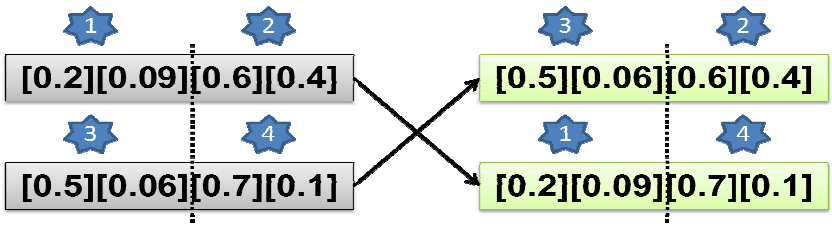

Figura 3. Exemplo Mutação Genética – Algoritmo Genético

Os algoritmos genéticos diferem dos algoritmos tradicionais de optimização basicamente em quatro aspectos:

- Baseiam-se numa codificação do conjunto das soluções possíveis, e não nos parâmetros da optimização em si;
- Os resultados são apresentados como uma população de soluções e não como uma solução única;
- Não necessitam de nenhum conhecimento derivado do problema, apenas de uma forma de avaliação do resultado;

- Usam transições probabilísticas e não regras determinísticas.

Os algoritmos genéticos tipicamente conseguem atingir soluções de melhor qualidade que a subida da colina e arrefecimento simulado. No entanto, são um método computacionalmente mais pesado e cuja implementação exige um maior esforço.

2.5 Conclusões

Neste capítulo foi apresentado o conceito de simulação ecológica sendo apresentados os principais tipos de modelos utilizados neste âmbito com ênfase para a simulação hidrodinâmica e simuladores ecológicos. Foram também apresentados os principais métodos de optimização tendo em vista a sua utilização num agente de calibração de simulações ecológicas.

Capítulo 3

3. Rede de Simulação *EcoSimNet*

O EcoSimNet é acrónimo para *Ecologic Simulator Network,* sendo um sistema multi-agente para simulações Ecológicas. O seu desenvolvimento é integrado no projecto ABSES: Agent Based Simulation of Ecological Systems (FCT/POSC/EIA/57671/2004). O projecto ABSES pretende dar continuidade ao sistema de simulação ecológico desenvolvido no projecto DITTY (Development of an Information Technology Tool for the Management of European Southern Lagoons under the influence of the river-basin runoff), projecto de investigação e desenvolvimento europeu (EVK3-2002-00084) onde foi desenvolvido um Sistema de Apoio à Decisão (SAD) para a gestão de ecossistemas costeiros.

O projecto ABSES pretende adicionar algumas funcionalidades de sistemas multi-agente que irão actuar sobre o simulador ecológico. A utilização de sistemas multi-agente em ambientes de simulação ecológica é uma característica não muito explorada em sistemas de simulação ecológica o que poderá trazer algumas descobertas e avanços na área.

Pretende-se representar no sistema de simulação uma maior complexidade de influências externas ao desenvolvimento habitual do ecossistema, normalmente introduzidas pelo factor humano: cultura de bivalves, exploração de locais de turismo, zonas de despejo de ETAR's, etc. (Pereira et al., 2005).

O sistema é composto por várias aplicações de software, tendo cada uma, funcionalidades bem definidas, tendo em conta os objectivos traçados: plataforma com capacidade de adaptação a novos modelos e incorporação de vários actores que podem influenciar o ecossistema costeiro.

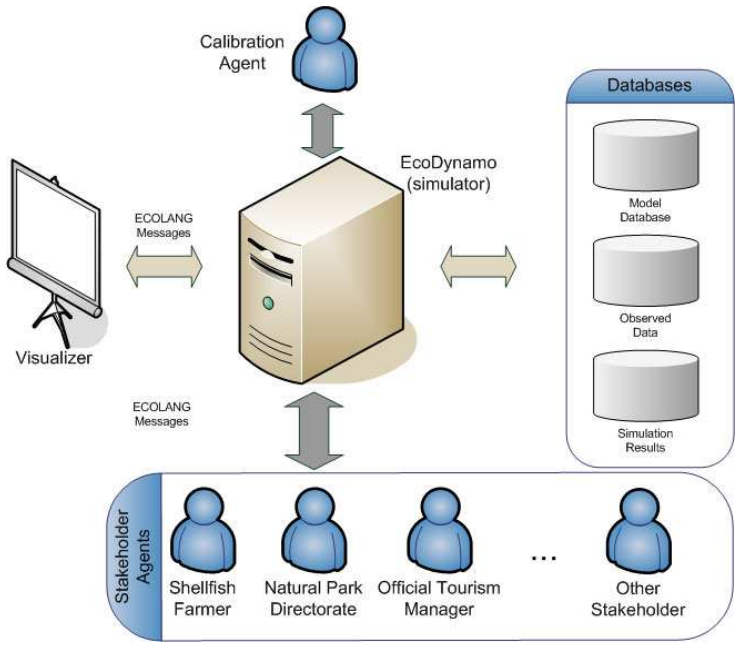

Figura 4. Esquema de interacção entre os componentes de Sistema de Simulação

Como a figura apresenta, toda a plataforma se baseia no simulador EcoDynamo, responsável pela troca de mensagens assim como pela simulação do modelo. À volta do simulador foram desenvolvidas aplicações, cada uma com objectivos específicos, interrogando o simulador através de mensagens ECOLANG, permitindo ao sistema a sua escalabilidade em termos de ferramentas de simulação e extracção de conhecimento à volta da mesma. Exemplo de uma das aplicações desenvolvidas, o agente aquicultor, que tem como principal objectivo maximizar a produção da espécie animal, bivalves, encontrando as melhores zonas para cultivo, disponíveis no modelo. O utilizador define uma área onde pretende lançar as suas colheitas, e o agente encontra as sub-zonas com maior potencial de produção, realizando simulações com combinações diferentes de locais (boxes).

3.1 Sistema Multi-Agente

A arquitectura proposta para o sistema de simulação (figuras 4 e 5) é baseada na utilização de agentes inteligentes [Weiss, 1999; Wooldridge, 2002; Norvig and Russel, 2003; Reis, 2003], representado as entidades do sistema no contexto de um sistema

multi-agente. A arquitectura multi-agente torna o sistema mais flexível a actualizações e exigências do modelo assim como requisitos dos intervenientes.

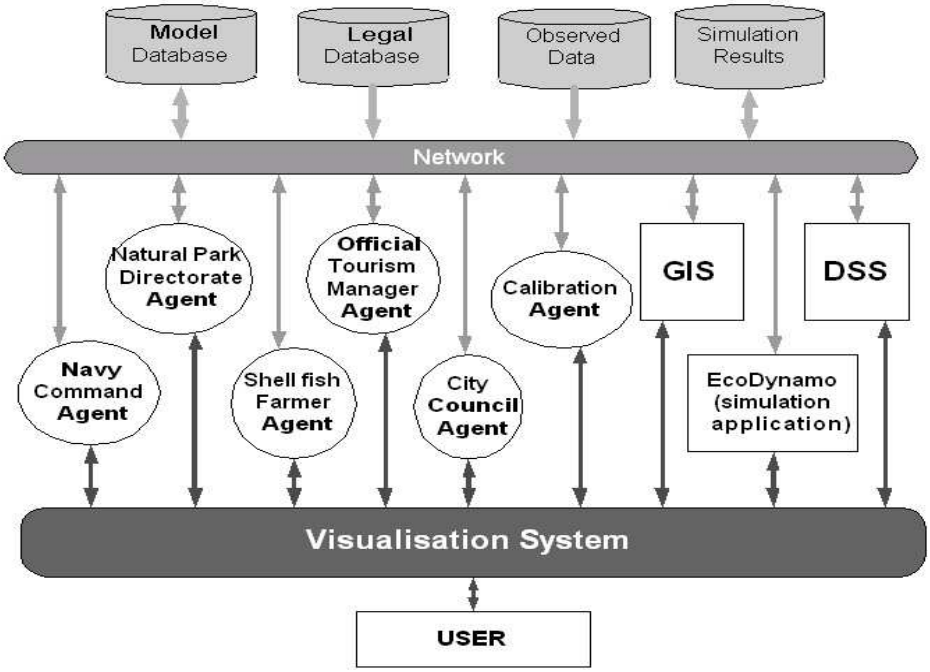

Figura 5. Arquitectura Sistema (adaptada de Pereira et al., 2004)

Cada interveniente no sistema ecológico é incluído no sistema de simulação como um agente inteligente (agente aquicultor, agente gestor parque natural, agente promotor turístico, etc.). Os agentes têm autonomia para planear acções, que são transmitidas para o simulador, influenciando o comportamento do ecossistema; o sistema de simulação processa a simulação, sendo os resultados apresentados aos decisores, num formato adequado, ajudando no processo de tomada de decisão. O simulador comporta-se como um Sistema de Apoio a Decisão, com capacidade de prever, para um dado modelo, com um grau de confiança aceitável, alterações no ecossistema.

Todas as comunicações entre as entidades são realizadas através de mensagens ECOLANG (Pereira et al., 2005) – linguagem de comunicação desenvolvida especialmente para o projecto, actuando como plataforma universal de comunicação entre as aplicações de software. Utilizando uma linguagem de comunicação e mensagens pré-definidas, o sistema permite a inclusão de novas mensagens, sem a necessidade de alterar a arquitectura das aplicações de software e/ou agentes inteligentes.

Apesar de cada aplicação possuir interface própria, o utilizador pode aceder às diferentes informações dos componentes, através do sistema de visualização, que é comum a todas as aplicações (módulo visualizador).

Com a divisão do sistema em agentes, e a existência de uma linguagem partilhada de comunicação, é possível representar diferentes objectivos para o mesmo modelo de simulação, cada um extraindo a informação que lhe interessa, convertendo-a em conhecimento, que poderá alimentar novamente a simulação, com os novos dados.

3.2 Simulador *EcoDynamo*

Um dos componentes mais importantes do projecto ABSES, o simulador ecológico, tem como principal objectivo a fácil utilização e manuseio, permitindo aos intervenientes do processo de decisão a sua utilização, sem para tal necessitarem de grandes conhecimentos de informática (óptica do utilizador). Este pedaço de Software (inicialmente apelidado de EcoDyn, sendo mais tarde alterado para EcoDynamo – **Eco**logical **Dyn**amics **Mo**del) foi parcialmente desenvolvido no âmbito do projecto europeu DITTY (www.dittyproject.org).

O EcoDynamo foi idealizado para ter várias funcionalidades, tais como, possuir uma plataforma de comunicação para com agentes inteligentes, usando a linguagem ECOLANG – linguagem de comunicação para simulação de sistemas ecológicos complexos (Pereira et al., 2005).

A simulação de sistemas ecológicos no EcoDynamo requer a segmentação de um dado modelo a simular, em áreas quadradas de *n* metros. Sendo as várias fórmulas matemáticas de simulação ecológicas aplicadas a estas regiões tendo em conta a ligação física entre elas.

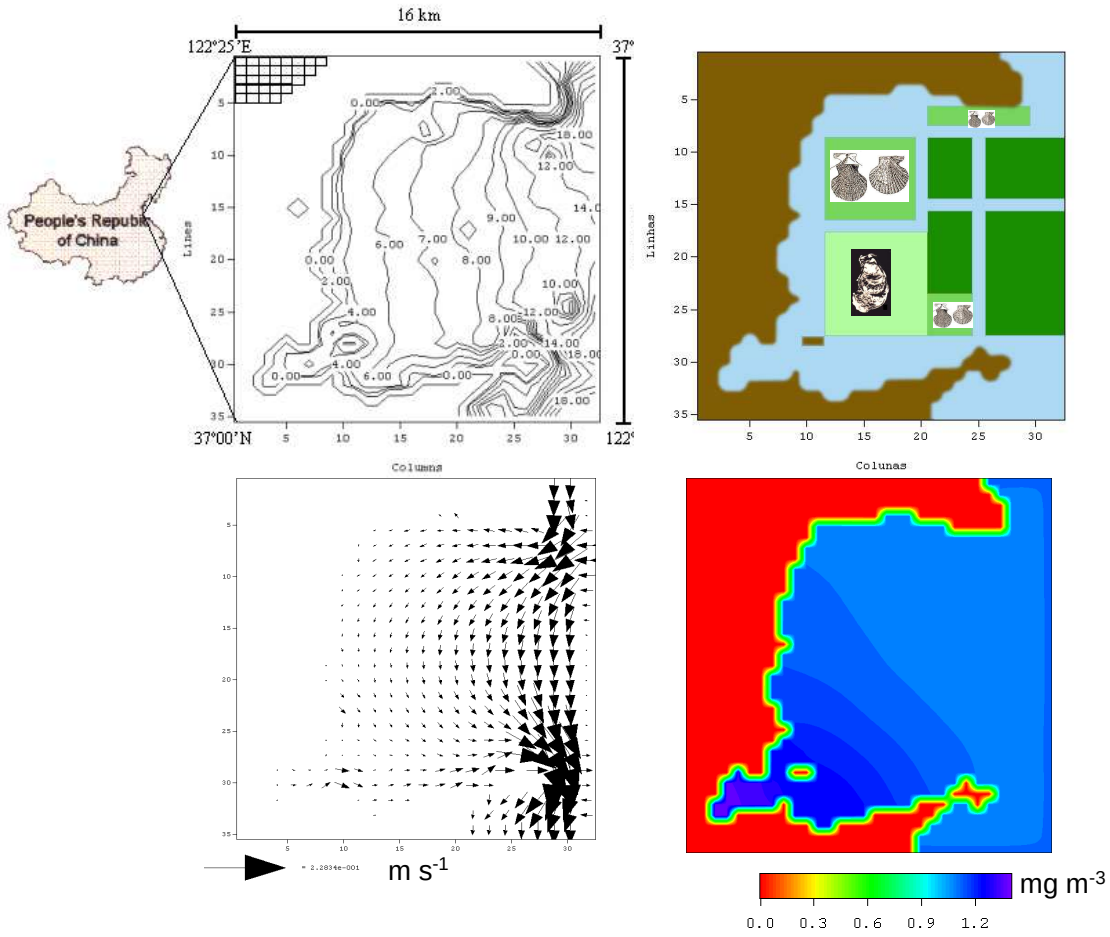

Figura 6. Esquema da Baía de Sangoo

Os modelos ecológicos simulados pelo EcoDynamo foram construídos baseado em modelos ecológicos reais. Exemplos de alguns modelos ecológicos que podem ser simulados realisticamente com o EcoDynamo incluem a Ria Formosa em Portugal e a Baía de Sangoo na China. Esquemas destes dois modelos podem ser vistos na Figura 2 – Ria Formosa e Figura 6 – Baía Sangoo. (Duarte et al., 2003)

O EcoDynamo consegue simular componentes físicas e biológicas dos modelos representados ao longo do tempo. Alguns dos aspectos que o EcoDynamo simula incluem: a subida e descida das marés, sedimentação de bioelementos, densidade e distribuição ao longo do tempo e espaço de matéria orgânica e inorgânica, processo de

crescimento, reprodução e morte de várias espécies de *fitoplancton* e *zooplancton*.

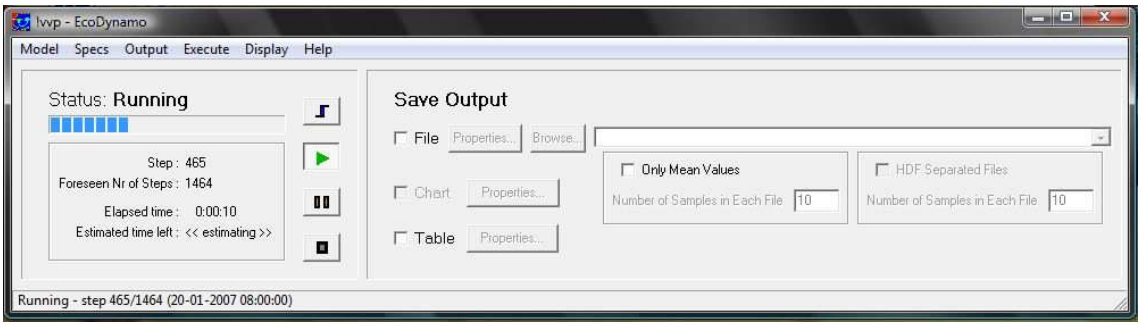

Figura 7. Interface Simulador EcoDynamo

A aplicação foi desenvolvida usando a linguagem de programação C++, segundo o paradigma Orientado a Objectos, tendo capacidades de modelação para ecossistemas aquáticos. O simulador apresenta-se com uma interface gráfica dividida em 2 blocos: correr a simulação e opções para guardar os resultados de simulação (figura 7). Permite ao utilizador seleccionar o modelo e configurar o ambiente de simulação, assim como assegurar a comunicação entre os vários objectos (ou classes) e componentes que dependam dos seus resultados.

Seguindo a associação com a metodologia Orientada a Objectos, cada entidade é representada no simulador, por uma classe de modelo, com as variáveis e comportamentos associados, tal e qual se comportam no mundo real. Como as classes interagem entre si, os resultados (variáveis) não estão unicamente dependentes das equações matemáticas dos processos intrínsecos, mas também dos valores (parâmetros) que constituem a equação.

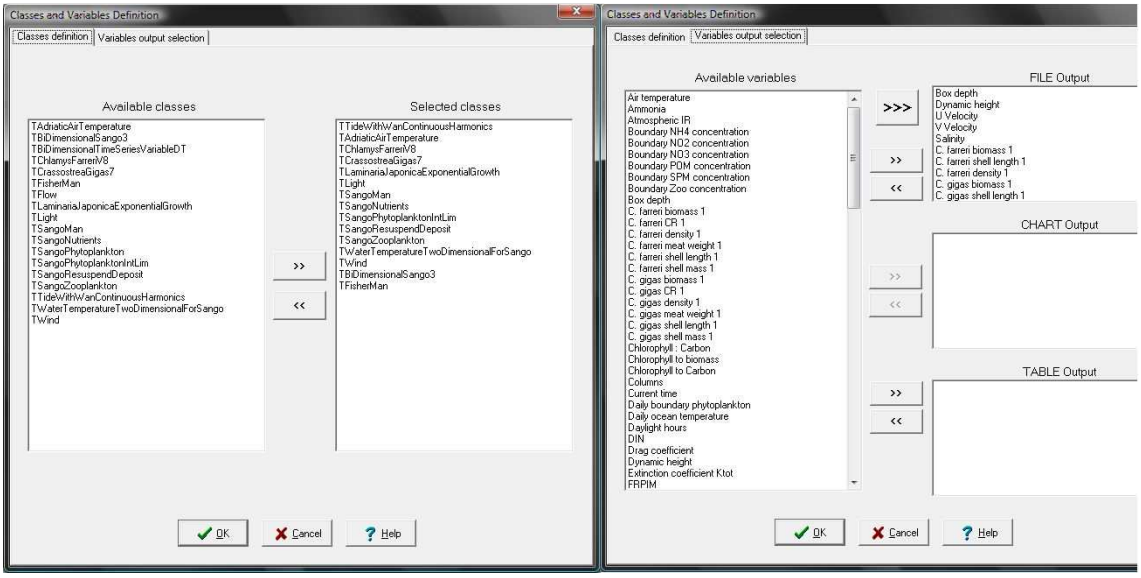

Figura 8. Interface - Seleccionar Classes e Variáveis para Simulação

Esses parâmetros podem ser influenciados pelo resultado das variáveis de outra classe e assim sucessivamente (tabela 1). De uma forma simplificada, cada classe comporta-se como uma caixa negra, que reage mediante a entrada de dados, produzindo resultados de saída, que podem vir a alimentar a entrada de dados de outras classes, a modelar pelo sistema.

Tabela 1. Exemplo Nomes de Classes e Variáveis do EcoDynamo

Class Type	**Class Name**	**Class outputs**
Classes providing forcing functions	Wind	Wind speed
	Air temperature	Air temperature
	Water temperature	Irradiative fluxes and balance between water, atmosphere and water temperature
	Light intensity	Total and photo synthetically active radiation (PAR)
	Tide object	Tidal height
Classes providing state variables	Hydrodynamics 2D	Sea level, current speed and direction
	Sediment biogeochemistry	Inorganic nitrogen, phosphorus and oxygen, sediment adsorbed inorganic phosphorus, organic phosphorus, nitrogen and carbon
	Dissolved substances	Ammonia, nitrate and nitrite, inorganic phosphorus and oxygen
	Suspended matter	TPM, POM, POC, PON, POP and extinction coefficient
	Phytoplankton	Phytoplankton biomass, chlorophyll productivity and cell nutrient quotas
	Enteromorpha sp. and *Ulva sp.*	Macroalgae biomass, productivity and cell nutrient quotas
	Zostera noltii	Macrophyte biomass and numbers, cell nutrient quotas and demographic fluxes
	Clams (*Ruditapes decussatus*)	Clam size, biomass, density, filtration, feeding, assimilation and scope for growth

A aplicação permite ao utilizador seleccionar as classes que deseja simular, através da interface, aumentando assim as possibilidades de teste de simulação (figura 8).

Os processos de simulação incluem:

- **Hidrodinâmica dos Sistemas Aquáticos**: correntes marítimas e suas velocidades;
- **Termodinâmica**: equilíbrio entre a atmosfera e a superfície aquática, e sua temperatura;
- **Bioquímica**: interacção entre as espécies biológicas e os nutrientes;
- **Pressões antropogénicas**, tal como a recolha de biomassa.

As propriedades das características do ecossistema estão descritas na base de dados do modelo, que não são mais do que ficheiros de configuração, podendo encontrar a

informação: representação morfológica e geométrica do modelo, dimensão do modelo, número de células – classes, variáveis de saída, valores iniciais dos parâmetros e seus limites.

A comunicação entre os diferentes objectos que representam variáveis e processos, respectivamente, pode ser realizada através da consola. Permite obter um histórico das interacções entre os diferentes objectos, sendo uma ferramenta importante para o processo de aprendizagem, introduzido anteriormente.

O utilizador pode escolher diferentes formatos de recolha de dados de simulação: ficheiro, gráfico ou tabela. Estes formatos de recolha de dados são compatíveis com algumas das aplicações comerciais, tais como o Software MatLab®, permitindo o tratamento dos dados a *posteriori*.

O EcoDynamo, possui uma interface (onde implementa mensagens ECOLANG), permitindo a outros módulos/programas a sua comunicação e controlo das acções principais de simulação. Por exemplo, as simulações podem ser controladas fora do interface gráfica do EcoDynamo, através dos comandos start / stop / pause / step.

O simulador ecológico EcoDynamo é uma ferramenta com capacidade de realizar simulações realísticas de modelos ecológicos complexos em vários níveis de abstracção. O simulador permite comunicar com um sistema de visualização e com agentes com diferentes objectivos e capacidades, tais como o agente calibrador de modelos, que este trabalho retrata, e um agente aquicultor, cujo objectivo é a maximização da produção de espécies com valor económico (bivalves).

3.3 ECOLANG - Linguagem de Comunicação para Redes de Simulação Ecológicas

A linguagem ECOLANG (Pereira et al., 2005; Pereira, 2008) foi desenvolvida com o objectivo de facilitar a troca de informações entre a aplicação de simulação ecológicas e os agentes externos. É uma linguagem de alto nível com capacidade de descrição do sistema ecológico em termos de características regionais, percepções dos agentes e suas acções, independentemente de qualquer plataforma de *hardware* ou *software*.

Este novo projecto, foi baseado no trabalho de Reis e Lau, a COACH UNILANG (Reis and Lau, 2002). A linguagem - COACH UNILANG inserida no contexto do futebol, permite a um agente treinador, comunicar com os agentes Jogadores, os quais necessitam de coordenação para a formação da equipa e estratégias eficazes para o período de jogo. Esta linguagem, associada a um protocolo de comunicação, permite aos agentes num ambiente multi-agente, compreender as suas interacções no domínio ecológico. Alguns dos pré-requisitos levantados aquando da criação da nova linguagem:

- Linguagem de alto-nível com capacidades de compreensão entre agentes de software e utilizadores humanos;
- Deve ter uma validação sintáctica simples;
- A Ontologia deve ser orientada para Sistemas Aquáticos;
- Facilmente adaptável a novos actores no Sistema;
- Deve ser independente de qualquer plataforma de Hardware ou Software, assim como do Sistema Operativo e Linguagem de desenvolvimento.

As mensagens ECOLANG descrevem as características regionais dos sistemas ecológicos, acções e percepções dos agentes, possibilitando várias camadas de comunicação.

Tabela 2. Lista de Tipos de Acções ECOLANG

Acção	Descrição
Execução	Comandos relativos à simulação do modelo (run, stop, pause, etc.)
Configuração	Escolha de um subdomínio para simulação, classes, alteração de variáveis e valores iniciais para os parâmetros, alteração áreas de cultivo de aquacultura, escolha variáveis, períodos de simulação e intervalos para observação, antes do modelo correr.
Definições	Recepção da morfologia e áreas de cultivo definidas pelo modelo, agregação das células em regiões de acordo com algumas propriedades, definição de subdomínios baseados nessas novas regiões.
Estatísticas	Recolha de resultados das experiências de simulação e comparação com resultados anteriores ou dados reais, auxiliando o módulo de configuração para as melhores acções a executar.

Eventos	Mensagens espontâneas que os agentes ou aplicações geram para informar sobre eventos ou resultados importantes.

As mensagens ECOLANG regem-se pelo formalismo BNF. Backus-Naur Form (BNF) é uma meta-linguagem (linguagem usada para descrever linguagens) com provas dadas no campo da ciência dos computadores. BNF é amplamente usada como notação para a gramática de linguagens de programação, conjuntos de instruções e protocolos de comunicação, assim como, para representações parciais das gramáticas de linguagem natural (Naur 1960). John Backus e Peter Naur foram os precursores, descrevendo a sintaxe da linguagem Algol 60 de forma inequívoca. A notação ECOLANG é uma extensão ao formalismo original da BNF, adicionando-se as seguintes extensões (meta-símbolos):

- { } para itens repetidos (uma ou mais vezes);
- [ ] delimita tipos de valores;
- Símbolos terminais usam a formatação Bold (carregado) para as letras.

A definição completa da sintaxe da ECOLANG e exemplos pode ser consultada em (Pereira et al., 2005) com a recente actualização em (Pereira, 2008).

A sintaxe base de cada mensagem pode ser descrita por:

```
<MESSAGE> ::= message (<ID> <SENDER> <RECEIVER> <MSG_CONTENT>)
<ID> ::= [integer]
<SENDER> ::= [string]
<RECEIVER> ::= [string]
```

Figura 9. Exemplo abstracto mensagem ECOLANG

<ID> identificador da mensagem – é um número inteiro sequencial controlado por cada emissor (valor inicial é 1).

<SENDER> nome agente emissor da mensagem.

<RECEIVER> nome agente destinatário da mensagem.

<MSG_CONTENT> conteúdo da mensagem.

Cada mensagem é representada por uma referência numérica para facilitar a sua identificação.

As mensagens trocadas pelas aplicações e agentes podem ser de quatro tipos: ligação, definição, acções e percepções, tal como já foi referido na tabela 2.

As mensagens de ligação estabelecem o canal de comunicação entre agentes e/ou aplicações de Software e especificam o computador onde cada agente pertence, sua porta de comunicação, onde cada agente fica à "escuta" das mensagens para ele dirigidas - <sender>.

Apesar de existirem acções e percepções dedicadas a tipos de agentes do Sistema (devido a especificidade e contexto associado), não existem restrições quanto à sua utilização por parte das aplicações.

```
<MSG_CONTENT> ::= <CONNECTION_MSG> | <DEFINITION_MSG> | <ACTION_MSG> | <PERCEPTION_MSG>
```

Figura 10. Tipos de Mensagem ECOLANG

Mensagens de ligação definem o início e o fim da sessão de comunicação entre aplicações. Neste grupo encontram-se também, as mensagens de validação da outra parte da sessão de comunicação estabelecida. Isto permite a criação de ligações entre múltiplas aplicações, facilitando a expansão das comunicações e da rede de conhecimento.

Desde a versão 1.3 deste protocolo, as mensagens permitem a definição das regiões e informação acerca do tipo de modelo em uso pelo simulador, a sua dimensão, morfologia e espécies animais em simulação. Este tipo de mensagens tem sofrido mais alterações devido às necessidades de cada agente/aplicação face ao simulador.

As mensagens de acções estão intrinsecamente ligadas a cada tipo de agentes envolvido no sistema, pelas suas especificidades e objectivos a concretizar. Como exemplo, um agente/aplicação que tenha interesses na produção de moluscos as suas acções passam pelo depósito, inspecção e recolha das espécies.

As mensagens de percepção actuam de igual forma das de acções, ao estarem dependentes do tipo de agente envolvido, e das acções realizadas por cada um sobre o simulador. Retomando o exemplo do agente produtor de moluscos, cuja intenção é a produção da dita espécie, as suas percepções são o resultado das acções desenroladas, que podem ser expressas em unidades de medida.

3.4 Desenvolvimento de Agentes Inteligentes

A comunicação entre o simulador (EcoDynamo) e os outros agentes/aplicações de software presentes no sistema de simulação é usualmente do tipo *handshake* – expressão utilizada nas áreas das telecomunicações e tecnologias de informação, sendo um processo automático de negociação que dinamicamente estabelece os parâmetros do canal de comunicações entre duas entidades antes da comunicação propriamente dita. É seguida do estabelecimento da ligação física do canal e transferência normal da informação. Neste caso específico, uma mensagem do tipo acção, espera receber uma resposta da aplicação destinatária; essa resposta vem na forma de uma percepção. Apenas as mensagens espontâneas e de registo, enviadas pelo simulador, não necessitam de uma mensagem de retorno.

A primeira mensagem que cada agente realiza é a intenção de se ligar ao simulador, com a mensagem *connect*, tal como demonstrado pela figura 11, ponto 1. O agente "Calibration" dá-se a conhecer ao servidor (EcoDynamo), indicando os valores da sua localização (nome da máquina, IP e porta de comunicação). A resposta é uma mensagem de aceitação (*to accept ok result*), demonstrado pela figura 11, com o segundo valor.

Figura 11. Inicialização Agente no EcoDynamo – troca de mensagens

Cada mensagem possui um identificador, gerado pela aplicação servidora, o qual permite ter comunicações assíncronas - transmissão de dados sem recorrer à utilização de um sinal de sincronismo (chamado de relógio). Desta forma cabe aos receptores gerirem as mensagens e criar mensagens de resposta, com o mesmo identificador da mensagem recebida, para o mesmo agente. Cabe às aplicações gerir a interpretação das mensagens recebidas, de forma, a esperar por uma determinada mensagem de retorno ou não. O protocolo de comunicação não tem definido as dependências para tipos de mensagens.

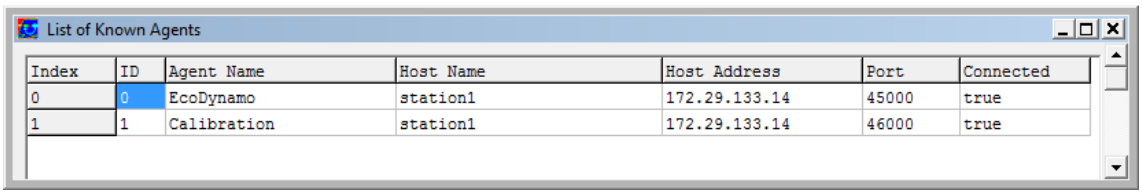

Figura 12. Lista de Agentes registados pelo servidor EcoDynamo

Após o agente receber a mensagem de *accept*, este está registado junto do simulador (servidor) como um agente com interesse em obter informações ou indicar comandos para a simulação (figura 12). Quando um agente sai do sistema, deve enviar uma mensagem de desligar do simulador.

3.5 Conclusões

Neste capítulo foi apresentado a plataforma de simulação EcoSimNet, onde o simulador EcoDynamo tem um papel predominante, na representação das alterações levadas a cabo nos modelos ecológicos aquáticos. Devido à plataforma possuir um protocolo de comunicações, baseando-se no vocabulário ECOLANG, permite o desenvolvimento de aplicações em seu redor, permitindo aumentar as funcionalidades da plataforma, assim como a partilha de código fonte, através das bibliotecas de funções padronizadas.

Estas características permitem aos diversos intervenientes do processo ecológico ter presente o conjunto de ferramentas que satisfaça as suas pesquisas, retornando informação pertinente no formato mais adequado. É neste propósito que o próximo capítulo apresenta o desenvolvimento de um novo agente, cuja finalidade, é a pesquisa dos melhores valores para a calibração de modelos de simulação.

Capítulo 4

4. Projecto e Implementação

O agente de calibração é um agente inteligente que comunica, através do protocolo de comunicações ECOLANG, com a aplicação de simulação EcoDynamo, assumindo o controlo sobre as tarefas primárias sobre a compreensão do modelo (por exemplo, ler/alterar os valores dos parâmetros, correr a simulação, recolher resultados). Seu objectivo é encontrar o melhor conjunto de valores de parâmetros, permitindo desta forma que os resultados de simulação sejam similares com os obtidos pelo sistema, através da calibração do modelo e sua validação.

A especificação e desenvolvimento do agente permite:

- Selecção de um modelo ecológico para teste;
- Correr o modelo para recolher informação sobre a interacção entre as diferentes classes;
- Definição/alteração de valores de parâmetros em tempo real;
- Monitorização, em tempo real, dos valores das variáveis e analisar as suas sensibilidades à variação dos valores dos parâmetros;
- Visualização das mensagens trocadas com a aplicação de simulação EcoDynamo;
- Definição de estratégias de calibração para aumentar a sua rapidez de convergência;
- Definição de critérios de análise para comparar as várias estratégias de calibração testadas;
- Implementação de um algoritmo de aprendizagem que permita, automaticamente, adoptar uma estratégia de alteração de parâmetros;
- A monitorização e registo do seu processo de aprendizagem;

- Correr o modelo ciclicamente, alterando os valores dos parâmetros, recolhendo os valores das variáveis e comparando-os com os valores referência utilizados para a calibração, tentando encontrar os valores ideais para obter a convergência de resultados;
- Verificar a calibração do modelo, comparando os resultados obtidos com outro conjunto de dados diferentes dos utilizados para a calibração.

4.1 Arquitectura

Todas as comunicações realizadas com o simulador, usam mensagens ECOLANG para realizar os inputs/outputs com o modelo carregado pela aplicação de simulação.

O Agente Calibrador adquire conhecimento sobre o comportamento dos processos do sistema em 5 passos (ver figura 13): 1) simulador carrega da base de dados do modelo, o esquema e os valores dos parâmetros iniciais para as equações do modelo, 2) o Agente inquire o simulador sobre a lista de parâmetros e seus valores, 3) faz alterações aos valores dos parâmetros, usando técnicas baseadas em conhecimento prévio do modelo, 4) corre a simulação e 5) compara a diferença dos resultados das variáveis do modelo com os dados reais.

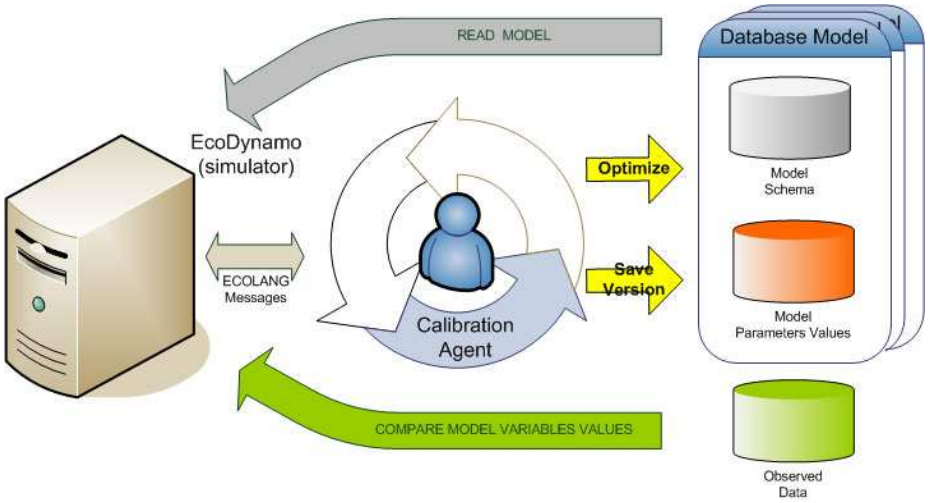

Figura 13. Esquema Agente Calibrador

O processo termina quando os critérios de convergência são atingidos, ou o utilizador obriga, através da interface gráfica para o processo de optimização. O utilizador pode

guardar a solução de parâmetros encontrada até ao momento, para utilização em simulações futuras, ou iniciar o processo de optimização de parâmetros a partir desse ponto.

O processo é iterativo, e o seu sucesso depende, exclusivamente, na escolha dos correctos parâmetros e seu valor. Outro ponto a ter em conta é a análise de sensibilidade dos valores dos parâmetros, visto estes poderem ser de grandezas numéricas diferentes, sendo necessário, o seu estudo antes do processo de calibração. O uso de agentes inteligentes, pode fazer toda a diferença, devido à sua capacidade de aprendizagem e de alteração de estratégia em qualquer altura do ciclo de computação.

4.2 Tecnologia

Toda a plataforma do Agente Calibrador foi idealizada segundo os conceitos OO, para facilidade de reutilização e capacidade de adição de novas funcionalidades. Como é possível observar no esquema da figura3, definiram-se vários blocos de classes, tendo cada um, objectivos bem definidos. A comunicação entre classes é realizada através do mecanismo de herança ou instanciação.

O simulador EcoDynamo foi programado por diversos profissionais, os quais utilizaram as linguagens de programação que melhor dominavam, o que leva a migração de código, e por consequente a não utilização do potencial da linguagem orientado a objectos. Um exemplo disso, a representação interna do modelo, seus parâmetros e variáveis, que no simulador, é realizado em estruturas. Para normalizar os conceitos de programação, com a linguagem C++, criou-se classes específicas para armazenar os dados do modelo (Data Manipulation), para os parâmetros, variáveis, modelo e simulação do modelo ao longo do tempo-espaço. Desta forma é possível criar estruturas de dados (por exemplo, vectores ou arrays) do tipo da classe, sendo o seu manuseio mais fácil.

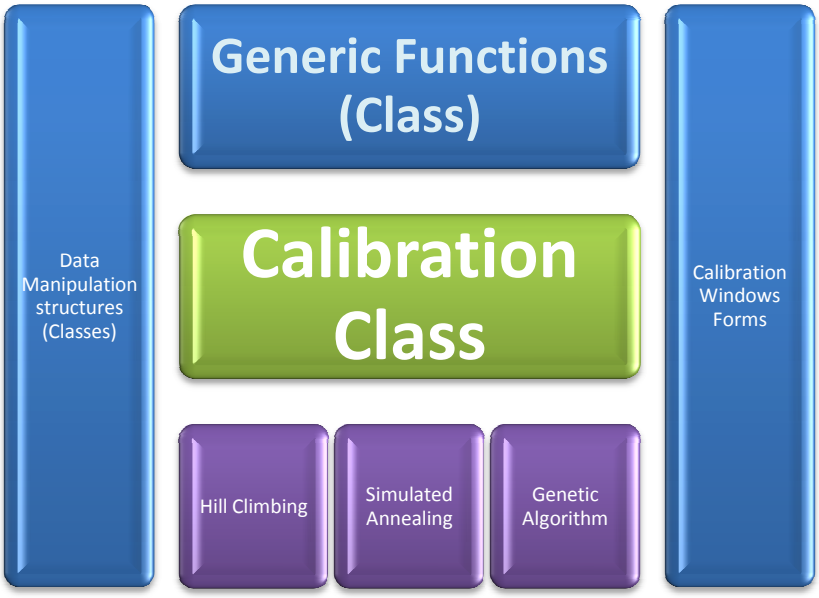

Figura 14 - Estrutura de Classes Agente Calibrador

A aplicação possui interface gráfica (window Forms), na qual o utilizador final interage com o Agente Calibrador. Quando a aplicação inicia o seu processo, é instanciada um objecto do tipo "Generic Functions", permitindo a todos os formulários gráficos da interface comunicarem entre si, partilhando os dados. Decidiu-se separar a componente

Figura 15. Representação agente calibrador em três camadas

gráfica das funções genéricas, seguindo a arquitectura de software em 3 camadas (Eckerson, 1995), possibilitando no futuro a migração para outro tipo de interface, sem necessidade de adaptação para as funções relativas ao comportamento.

As classes das interfaces apenas contêm variáveis locais necessárias para comunicarem com os objectos do formulário (por exemplo, caixas de texto ou *checkbox*), passando os seus valores como parâmetro de função para dentro da instância "Generic Functions".

No entanto podem ter associadas algumas funções de verificação de estado e semântica. Resumindo, a sua função é meramente de *input-output*.

Devido ao facto de os algoritmos de optimização partilharem um conjunto de funções de manuseio dos dados e comunicação com o simulador, criou-se uma classe base, onde esses métodos estão representados, e em alguns casos, definem-se as funções membro como virtuais, cabendo depois às classes derivadas, a sua implementação.

As classes associadas a cada algoritmo de optimização basicamente contêm as variáveis que são importantes para o manuseio do algoritmo, e o algoritmo propriamente dito. O output dos resultados, quer seja para o ecrã ou para ficheiro, é da responsabilidade da classe “Generic Functions”.

4.3 Módulos

O agente calibrador é constituído internamente pelas seguintes classes:

- Data Manipulation - Ficheiro responsável pela definição das estruturas de armazenamento dos dados provenientes do Simulador. Converte as estruturas de dados do Simulador para classes e vectores de classes. Permite uma maior facilidade de reutilização e gestão. Uma das primeiras acções do Agente Calibrador é o seu preenchimento com os valores retornados do Simulador.
- Generic Function - As funções genéricas a utilizar pelo agente calibrador são definidas dentro de uma classe com o mesmo nome. Uma das razões pela utilização de métodos de classe em vez de funções prende-se com a selecção dos algoritmos de calibração, que por sua vez, são classes que herdam todos os métodos desta.
- Calibration Module - Este módulo vai ser o cérebro do agente calibrador. Tem como função a inicialização da função objectivo e a chamada dos diferentes algoritmos de optimização. Pode ter associado um formulário de manipulação para personalização de alguns campos.

O sucesso do agente calibrador reside na qualidade com que os algoritmos de optimização conseguem encontrar os valores dos parâmetros mais indicados para as

características da simulação, sendo a medida de comparação, os valores das variáveis simuladas, com o mesmo valor observado na realidade. A ideia subjacente ao problema parece simples, atribuindo valores aos parâmetros, correr a simulação e comparar o valor das variáveis com amostras reais. O procedimento de arranque é muito similar em todos os métodos de optimização. Define-se de forma aleatória uma solução de valores de parâmetros, para começar a simulação. Após esta primeira interacção, cada algoritmo aplica as suas estratégicas.

Os problemas de optimização são baseados em três pontos principais: codificação do problema, a função objectivo, que se deseja maximizar ou minimizar e o espaço de soluções associado. Cada algoritmo comporta-se como uma caixa negra, onde gera uma solução de parâmetros, e verifica com a função objectivo se essa solução é boa ou não para resolver o problema.

4.3.1 Função de Avaliação

A função de avaliação é muito importante, por determinar a proximidade ou não da solução criada com a que pretendemos encontrar. Neste domínio de calibração de modelos ecológicos, o ponto de comparação, ou melhor dizendo, o grau de confiança de um simulador, é a sua capacidade de recriar com os mesmos resultados (ou muito similares) um cenário pelos quais já foram recolhidos os valores, em campo. A verificação do estado de uma solução assume-se como a comparação das diferenças entre os dados reais e os obtidos da simulação.

$$f(\chi) = \sum \textbf{weight} * |\mathbf{M}_{\textbf{original}} - \mathbf{M}_{\textbf{simulated}}|$$

A variável *weight* permite estabelecer escalas de correspondências entre variáveis. $M_{original}$ corresponde ao valor da variável real e $M_{simulated}$ ao valor simulado. Esse valor é guardado juntamente com o cenário (parâmetros + variáveis). Neste caso pretende-se minimizar o resultado da função objectivo, para valores próximos do zero.

Tabela 3. Exemplo Resultados de Simulação - Variáveis

Model Class	Parameter Name	Value
TPrey	IntrinsicRateOfIncrease	0.3
TPredator	FoodAbsorption	-0.0076
TPredator	FeedRate	0.064
TPredator	DeathRate	0.064

Step	Time	Prey abundance	Predator abundance
1	1167469200	10	10
2	1167472800	9.858333	9.971307
3	1167476400	9.719428	9.942724
4	1167480000	9.583221	9.914252
5	1167483600	9.44965	9.885888
6	1167487200	9.318655	9.857633
7	1167490800	9.190179	9.829484
8	1167494400	9.064163	9.801441
9	1167498000	8.940554	9.773503
10	1167501600	8.819296	9.74567
11	1167505200	8.700337	9.71794
12	1167508800	8.583626	9.690311
13	1167512400	8.469114	9.662785
…	…	…	…

A tabela anterior é exemplo de um resultado de simulação para uma solução gerada aleatoriamente. São armazenados os valores para cada variável do modelo (considerado como output), para cada unidade de tempo. Com base nesses resultados é calculado o valor do erro.

A função *NewSolution(int numRandom)*, responsável por criar uma nova solução (conjunto de valores de parâmetros), é genérica aos algoritmos de optimização, dado que é possível indicar o número de parâmetros a gerar novos valores. São escolhidos aleatoriamente os parâmetros a alterar, e verifica-se na estrutura do modelo, se o parâmetro possuía dados sobre valores mínimos e máximos, assim como a variação (*step*), e com base nessa informação gera-se um novo valor.

4.3.2 Implementação Hill-Climbing

Herda da classe *Calibration*. Aplica o algoritmo Subida de Colina. O conceito subjacente a este método é que a solução seguinte deverá ser melhor que as anteriores.

A definição da nova solução para teste tem associada uma probabilidade de 50% para alteração de 2 parâmetros ou 1 parâmetro face a melhor solução encontrada até ao momento. Para cada nova solução os parâmetros são escolhidos aleatoriamente, assim como os seus valores.

```
Início
   s = GerarSolucaoInicial2;
   T = Tinicial;
   Enquanto condição de fim fazer
       Se aleatorio() Entao
              s' = NovaSolucao(N(s,1));
       Senao
              s' = NovaSolucao(N(s,2));

       Se Avaliacao(s') < Avaliacao(s) Entao
              s = s';
Fim
```

Algoritmo 4. Hill-Climbing – Agente Calibrador

De forma a tornar a escolha mais "inteligente", dotou-se o algoritmo responsável pela criação de novas soluções, de direcção do novo valor (novo valor será mais baixo ou mais elevado), tendo como base a última solução encontrada. Isto permite que se determinado caminho tem bons resultados deve-se seguir nessa direcção. Estas alterações ao algoritmo base, permitem tirar proveito das qualidades do mesmo, ultrapassando as limitações do algoritmo.

Com esta forma de atribuição de novas soluções, o espaço de pesquisa deixa de ser fulcral para o desempenho do algoritmo, visto este seguir um caminho no sentido da melhor solução encontrada até ao momento, em vez de atribuir valores aleatórios dentro da gama de valores disponíveis para os parâmetros.

4.3.3 Implementação Arrefecimento Simulado

Herda da classe *Calibration*. Aplica o algoritmo Arrefecimento Simulado. Arrefecimento simulado ou *simulated annealing* é uma meta heurística para optimização que consiste numa técnica de pesquisa local probabilística, e fundamenta-se numa analogia com a termodinâmica. De forma análoga, o algoritmo de arrefecimento simulado substitui a solução actual por uma solução próxima (i.e., na sua

vizinhança no espaço de soluções), escolhida de acordo com a função de avaliação e com uma variável *T* (dita *Temperatura*, por analogia). Quanto maior for *T*, maior a componente aleatória que será incluída na próxima solução escolhida. À medida que o algoritmo progride, o valor de *T* é diminuído, começando o algoritmo a convergir para uma solução óptima, necessariamente local.

```
Início
  s = GerarSolucaoInicial;
  T = Tinicial;
  Enquanto condição de fim fazer
      Se aleatorio() Entao
            s' = NovaSolucao(N(s,1));
      Senao
            s' = NovaSolucao(N(s,2));
      Se Avaliacao(s') < Avaliacao(s)Entao
            s = s';
      Senao
            s = CriterioAceitacao(s,s',T);
      ActualizarTemperatura(T);
Fim
```

Algoritmo 5. Arrefecimento Simulado – Agente Calibrador

Para melhorar o desempenho do algoritmo, este simula algumas soluções dentro do mesmo valor de temperatura para estabilizar.

4.3.4 Implementação Algoritmo Genético

O algoritmo genético implementado não sofreu muitas alterações face ao tradicional. Para a componente da mutação genética, criou-se duas versões, uma cuja mutação da solução apenas difere do valor do passo. Na segunda adaptação, o valor do parâmetro é escolhido aleatoriamente dentro do intervalo de valores definido inicialmente. Tal como nos outros algoritmos, a componente de aleatoriedade é bastante utilizada para a criação da primeira geração, assim como da definição da solução a sofrer mutação.

4.4 Conclusões

Neste capítulo foi apresentado o agente calibrador e seus componentes, dando enfoque, nos métodos de optimização, e variações implementadas aos algoritmos padrão, colmatando algumas desvantagens, devidamente reconhecidas pela comunidade científica. A estrutura do agente calibrador foi pensada para ser capaz de adicionar mais algoritmos de optimização, tendo como base um conjunto de funções comuns a todos os algoritmos tendo sido organizado em hierarquia de classes (linguagem de programação).

Capítulo 5

5. Resultados e Análise

Tal como noutro processo de criação de software, a fase de testes é crucial para verificação da qualidade do software, sendo neste caso específico, a validade da robustez dos algoritmos calibradores implementados, assim como se estes eram capazes de convergir para a solução óptima de valores de parâmetros, sem estudo prévio da sensibilidade dos parâmetros, tal como defendem alguns autores (Jørgensen & Bendoricchio, 2001). Foram realizados estudos em dois modelos ecológicos com características diferentes, o primeiro, Modelo Predador-Presa, e o segundo, Modelo da Baía de Sangoo.

O modelo Predador-Presa foi adaptado para o simulador para validação do agente calibrador, pela sua simplicidade de interacções, visto representar apenas duas entidades, e pelo número reduzido de parâmetros a calibrar. O modelo da baía de Sangoo já apresenta maior complexidade em termos de interacções entre entidades, sendo um dos modelos em estudo pelo simulador EcoDynamo. O número de parâmetros e de variáveis a calibrar é em maior número.

Através da realização das baterias de testes e análise dos resultados obtidos, podemos tirar conclusões sobre o uso da calibração automática, em vez da tradicional, realizada por peritos.

5.1 Modelo Predador-Presa

Os matemáticos e ecologistas pioneiros, que descobriram as propriedades deste modelo, sugeriram que este tipo de relacionamento poderia explicar as oscilações observadas entre conjuntos de animais, tais como a lebre de neve e seu predador, o lince. As

oscilações regulares destas populações foram medidas por contagem de peles (cruas) no Canadá pela empresa Hudson Bay Company, de 1845-1935.

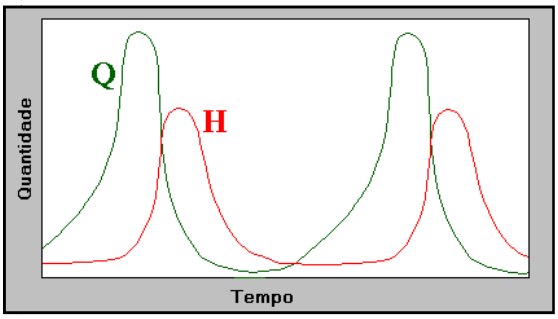

Gráfico 1. Distribuição Predador(h) e Presa(q) vs tempo

Quando a população de presas começa a crescer exponencialmente, a população de predadores cresce rapidamente fazendo que a população de presas se reduza novamente. Com menos comida disponível a população de predadores diminui. O gráfico das duas populações versus tempo mostra-se no Gráfico 1.

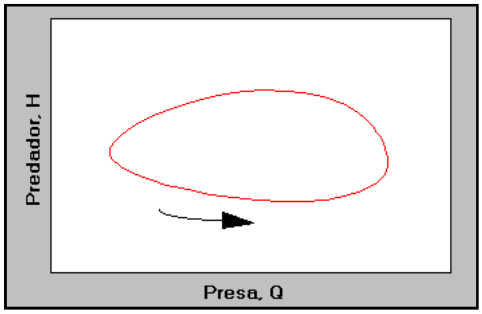

Gráfico 2. Relação Predador-Presa

Em lugar de desenhar as duas populações versus tempo, como no Gráfico1, pode-se fazer um gráfico com a quantidade de uma população sobre o eixo horizontal e a quantidade da outra população sobre o eixo vertical. Como resultado do processo de oscilação, obtém-se um gráfico circular de acordo com o Gráfico 2, mostrando que a oscilação é repetitiva.

5.1.1 Resultados

Tendo em conta a simplicidade do modelo Predador-Presa, este permitiu testar os algoritmos de optimização, permitindo ter uma visão crítica sobre os mesmos, realizando as modificações necessárias para acelerar o processo de calibração. Tendo

em conta que esta experiência serve de padrão para a experiência seguinte, com um modelo mais complexo usado pelo simulador, adoptou-se que os resultados da primeira simulação seriam considerados como a solução real para cálculo dos desvios, entre cada simulação. Devido ao comportamento aleatório destes métodos de optimização, correram para cada um, dez calibrações, tendo-se registado os valores do desvio para as primeiras 50 soluções encontradas.

O período de simulação foi de 2 meses, tendo sido aplicado um passo de simulação de 1 hora. Foi definido como critério de paragem, uma taxa de erro inferior a uma unidade.

Tabela 4. Lista de Parâmetros e Valores Óptimos: Predador-Presa

Classes do Modelo	Parâmetros	Valores
TPrey	IntrinsicRateOfIncrease	0.3
TPredator	FoodAbsorption	0.01
TPredator	FeedRate	0.1
TPredator	DeathRate	0.1

A tabela 4 mostra os parâmetros para o modelo predador-presa, indicando a solução considerada óptima, servindo de base para o cálculo da taxa de erro (função objecto).

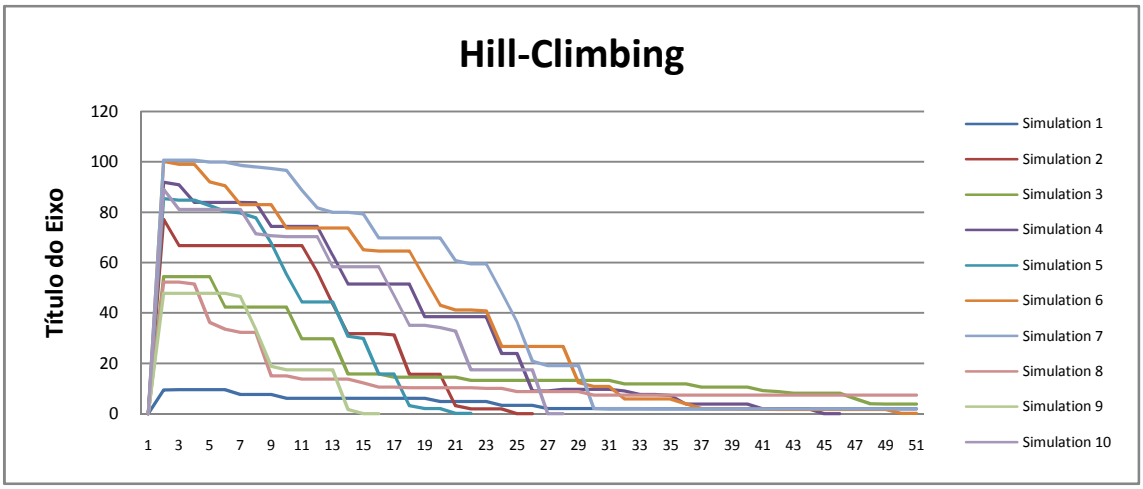

Gráfico 3. Resultados calibração Hill-Climbing para modelo Predador-Presa

O Gráfico 3 mostra os valores do desvio entre as soluções de parâmetros encontradas e os valores de comparação. Como já foi dito anteriormente a primeira simulação define os valores das variáveis para comparação, devendo no final dar o mesmo resultado em termos dos valores dos parâmetros. Na segunda simulação, é gerado uma solução aleatória dos valores dos parâmetros, daí se poder observar no gráfico, que a segunda

simulação contém os valores mais elevados de erro (*lack of fitness*). Apesar do algoritmo Hill-Climbing explorar as soluções vizinhas, mudando 1 ou 2 valores de parâmetros, da última melhor solução encontrada, foi adicionada uma variante, a direcção na determinação dos valores dos parâmetros.

Caso a diferença dos valores de parâmetros dos últimos melhores resultados seja positiva, o valor a atribuir deverá seguir na mesma direcção dos seus antecessores até que se obtenha piores resultados. Nesse caso, funciona o factor de aleatoriedade, dentro dos limites estabelecidos para cada parâmetro. Tendo em conta a direcção das últimas soluções encontradas, podemos constatar que quase todas as simulações seguem o mesmo padrão de comportamento, menos a primeira experiência, cuja solução aleatória (2.º ciclo de calibração) retornou um valor baixo de erro, levando a que as soluções vizinhas fossem piores que a anterior.

Tabela 5. modelo Predador-Presa com Hill-Climbing

Iteration	Simulation 1	Simulation 2	Simulation 3	Simulation 4	Simulation 5	Simulation 6	Simulation 7	Simulation 8	Simulation 9	Simulation 10
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	9.3821	77.2118	54.5079	91.9313	85.4325	100.1650	100.7340	52.3369	47.8699	89.1603
2	9.5157	66.7316	54.5079	90.9401	84.8454	99.0119	100.7340	52.3369	47.8699	81.0848
3	9.5157	66.7316	54.5079	83.9346	84.8454	99.0119	100.7340	51.5389	47.8699	81.0848
4	9.5157	66.7316	54.5079	83.9346	82.6190	92.0674	100.0180	36.3446	47.8699	81.0848
5	9.5157	66.7316	42.3526	83.9346	80.3701	90.5196	100.0180	33.5698	47.8699	81.0848
6	7.5494	66.7316	42.3526	83.9346	79.6830	82.9564	98.6714	32.3061	46.5970	81.0848
7	7.5494	66.7316	42.3526	83.7894	77.7489	82.9564	98.0344	32.3061	33.6722	71.4399
8	7.5494	66.7316	42.3526	74.3295	67.8005	82.9564	97.4136	15.0128	18.8583	70.6816
9	6.1185	66.7316	42.3526	74.3295	55.4341	73.7427	96.6721	15.0128	17.4450	70.3943
10	6.1185	66.7316	29.7917	74.3295	44.4303	73.7427	88.7408	13.7305	17.4450	70.3943
11	6.1185	56.1700	29.7917	74.3295	44.4303	73.7427	81.8435	13.7305	17.4450	70.3943
12	6.1185	43.8019	29.7917	63.0025	44.4303	73.7427	79.9773	13.7305	17.4450	58.4274
13	6.1185	31.7855	15.8455	51.4360	30.7547	73.7427	79.9773	13.7305	1.5940	58.4274
14	6.1185	31.7855	15.8455	51.4360	29.9006	65.1438	79.3988	12.1851	0.0000	58.4274
15	6.1185	31.7855	15.8455	51.4360	15.8252	64.6209	69.8151	10.6121	0.0000	58.4274
16	6.1185	31.3035	14.5463	51.4360	15.8063	64.6209	69.8151	10.6121		46.9655
17	6.1185	15.6785	14.5463	51.4360	3.2380	64.6209	69.8151	10.3631		35.1448
18	6.1185	15.6785	14.5463	38.4837	2.0427	53.8297	69.8151	10.3631		35.1448
19	4.8082	15.6785	14.5463	38.4837	2.0427	43.0874	69.8151	10.3631		34.2183
20	4.8082	3.2380	14.5463	38.4837	0.1665	41.1746	60.7605	10.3631		32.7969
21	4.8082	1.8769	13.2268	38.4837	0.1665	41.1746	59.5844	10.3631		17.4609

22	4.8082	1.8769	13.2268	38.4837		40.8662	59.5844	10.1135		17.4609
23	3.3405	1.8769	13.2268	23.8847		26.7540	48.2212	10.1135		17.4609
24	3.3405	0.0000	13.2268	23.8847		26.7540	36.5566	8.7572		17.4609
25	3.3405	0.0000	13.2268	9.0590		26.7540	20.9062	8.7572		17.4609
26	1.9923		13.2268	9.0590		26.7540	19.0752	8.7572		0.0000
27	1.9923		13.2268	9.6590		26.7540	19.0752	8.7572		0.0000
28	1.9923		13.2268	9.6590		12.3002	19.0622	8.7572		
29	1.9923		13.2268	9.6590		10.7569	1.9923	7.3790		
30	1.8506		13.2268	9.6590		10.7569	1.9923	7.3790		
31	1.8506		11.8865	9.0590		5.8763	1.9923	7.3790		
32	1.8506		11.8865	7.6711		5.8763	1.9923	7.3790		
33	1.8506		11.8865	7.6711		5.8763	1.8506	7.3790		
34	1.8506		11.8865	7.2113		5.8763	1.8506	7.3790		
35	1.8506		11.8865	3.7807		3.9458	1.8506	7.3790		
36	1.8506		10.5248	3.7807		2.0427	1.8506	7.3790		
37	1.8506		10.5248	3.7807		1.8769	1.8506	7.3790		
38	1.8506		10.5248	3.7807		1.8769	1.8506	7.3790		
39	1.8506		10.5248	3.7807		1.7628	1.8506	7.3790		
40	1.8506		9.1413	1.8769		1.7628	1.8506	7.3790		
41	1.8506		8.7304	1.8769		1.5940	1.8506	7.3790		
42	1.8506		8.1079	1.8769		1.5940	1.8506	7.3790		
43	1.8506		8.1079	1.8769		1.5940	1.8506	7.3790		
44	1.8506		8.1079	0.0000		1.5940	1.8506	7.3790		
45	1.8506		8.1079	0.0000		1.5940	1.8506	7.3790		
46	1.8506		6.0015			1.5940	1.8506	7.3790		
47	1.8506		3.9247			1.5940	1.8506	7.3790		
48	1.8506		3.7807			1.5940	1.8506	7.3790		
49	1.8506		3.7807			0.1665	1.8506	7.3790		
50	1.8506		3.7807			0.1665	1.8506	7.3790		

Apesar do registo da taxa de erro de todas as soluções testadas, podemos observar na tabela anterior os pontos onde o calibrador obtém melhores soluções. Quando aparecem na tabela valores de erro repetidos, isso indica que a solução encontrada é a melhor até ao momento, servindo de base na exploração de soluções vizinhas.

Apesar da taxa de erro estipulada, como sendo inferior a um, algumas experiências terminaram com taxa de erro igual a zero, o que quer dizer que encontraram os mesmos valores de parâmetros, inicialmente estabelecidos. Outras experiências, tais como a primeira, a sétima e a oitava, encontraram um mínimo local, onde podemos observar

que os valores dos erros entram em ciclo ou são muito similares. A simulação 3 conseguiu convergir para a solução inicial em 64 iterações.

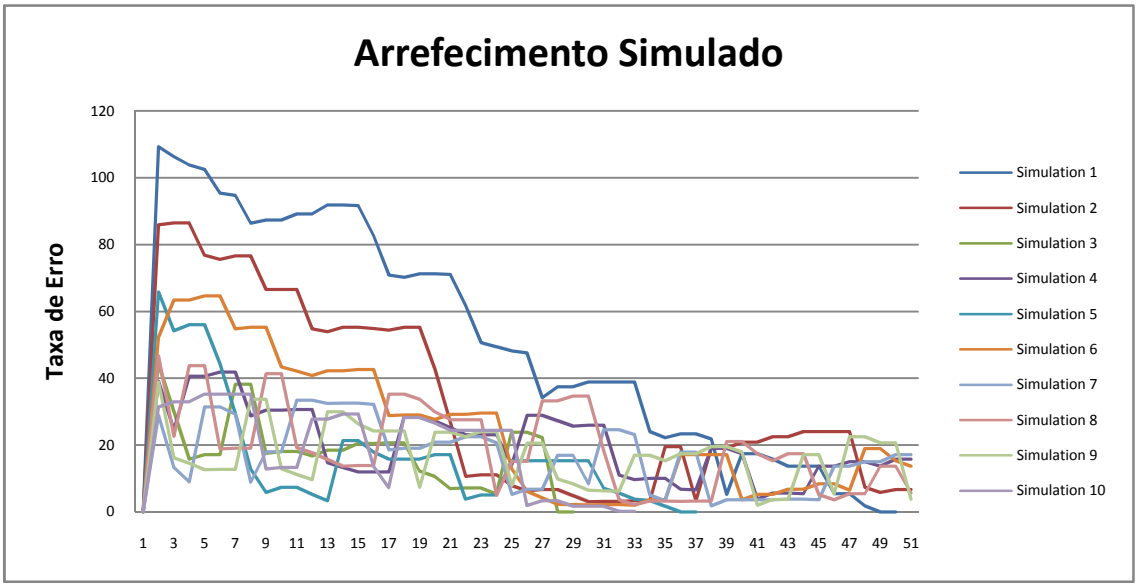

Gráfico 4. Gráfico Simulação Algoritmo Arrefecimento Simulado para modelo Predador-Presa

O algoritmo arrefecimento simulado comporta-se como o Hill-Climbing quando a temperatura é igual a zero. Pela leitura do Gráfico 4, podemos observar que as soluções apresentadas nem sempre são melhores do que as anteriores, evitando assim possíveis mínimos locais, coisa que o algoritmo Hill-Climbing apenas conseguia ultrapassar com *restarts* de solução aleatória. Verifica-se que a componente da probabilidade inerente à aceitação da solução como melhor, nem sempre parece ser a mais lógica, contudo apenas 2 simulações tenderam para mínimos locais, não conseguindo convergir para a solução inicial.

Tabela 6. modelo Predador-Presa com Arrefecimento Simulado

itera-tion	Simula-tion 1	Simula-tion 2	Simula-tion 3	Simula-tion 4	Simula-tion 5	Simula-tion 6	Simula-tion 7	Simula-tion 8	Simula-tion 9	Simulation 10
0	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
1	109,3520	85,8842	44,3308	39,0608	65,8052	52,2982	28,9237	46,7535	38,9080	31,4140
2	106,3710	86,4681	30,1081	24,5229	54,1892	63,4061	13,3562	22,6028	16,1840	32,9556
3	103,8710	86,4681	15,8455	40,6130	56,0367	63,4061	9,0093	43,7777	14,5608	32,9556
4	102,5120	76,8250	17,1692	40,6130	56,0367	64,6742	31,4140	43,7777	12,6228	35,2267
5	95,4703	75,5839	17,1692	41,8279	44,3395	64,6742	31,4140	18,8906	12,7808	35,2267
6	94,8210	76,6332	38,2448	41,8279	29,3907	54,8185	29,1779	19,0566	12,7808	35,2267

7	86,4765	76,6332	38,2448	28,7320	12,8221	55,2965	8,9355	19,0566	33,7376	35,2267
8	87,3851	66,5652	17,4890	30,3919	5,8566	55,2965	18,0457	41,3968	33,7376	12,8422
9	87,3851	66,5652	18,0851	30,3919	7,3581	43,3660	18,0457	41,3968	12,9386	13,2716
10	89,1930	66,5652	18,0851	30,6165	7,3581	42,1200	33,4258	19,2223	11,1820	13,2716
11	89,1930	54,8082	16,7875	30,6165	5,2836	40,8521	33,4258	17,7145	9,7072	27,7628
12	91,8969	53,9476	18,5212	14,7587	3,3831	42,2402	32,4394	15,6925	30,0373	27,7628
13	91,8969	55,2257	18,5212	13,3483	21,3714	42,2402	32,5645	13,7000	30,0373	29,2682
14	91,7264	55,2257	20,4703	11,9349	21,3714	42,6549	32,5645	13,8645	26,3489	29,2682
15	82,7416	54,8276	20,4703	11,9349	17,9103	42,6549	32,2226	13,8645	24,2128	13,6089
16	70,9159	54,4282	20,7087	11,9349	15,8062	28,8260	18,6086	35,2267	24,2128	7,2840
17	70,2912	55,2927	20,7087	28,7084	15,8062	29,0618	19,0752	35,2267	24,2128	28,2632
18	71,2934	55,2927	12,2557	28,7084	15,8062	29,0618	19,0752	33,6851	7,4143	28,2632
19	71,2934	42,6310	10,6072	27,2310	17,1313	27,7301	20,9062	29,8500	23,8842	26,6531
20	71,1179	26,8465	7,0240	25,1268	17,1313	29,2444	20,9062	27,6139	23,8842	24,4170
21	61,8054	10,6588	7,1854	23,0539	3,8350	29,2444	22,4949	27,6139	22,5217	24,4170
22	50,6895	11,0606	7,1854	23,0539	5,0292	29,5871	22,4949	27,6139	24,1338	24,4170
23	49,4625	11,0606	5,3319	23,0539	5,0292	29,5871	20,6584	5,0177	24,1338	24,4170
24	48,2212	7,8169	23,8251	14,1258	15,3434	12,8492	5,3075	15,1634	7,9525	24,4170
25	47,6288	6,3642	23,8251	28,9160	15,3434	6,1466	6,7762	15,1634	20,5671	1,8769
26	34,2183	6,6937	22,2150	28,9160	15,3434	4,2133	6,7762	33,2372	20,5671	3,3831
27	37,4852	6,6937	0,0000	27,3180	15,3434	2,3074	16,9302	33,2372	9,8963	3,3831
28	37,4852	4,8897	0,0000	25,6987	15,3434	2,1637	16,9302	34,6756	8,4445	1,7628
29	38,8670	3,0605		25,9344	15,3434	2,1008	8,4445	34,6756	6,5017	1,7628
30	38,8670	3,2006		25,9344	7,1094	2,1791	24,5772	18,3309	6,3642	1,7628
31	38,8670	3,2006		11,0543	5,6155	2,1791	24,5772	3,6740	6,2266	0,1667
32	38,8670	3,3405		9,6491	3,8165	2,0137	23,1660	2,1637	16,9302	0,1667
33	24,0631	3,3405		10,0309	3,4408	3,6080	5,0292	3,2836	16,9302	
34	22,2412	19,5205		10,0309	1,7628	3,6080	3,3405	3,2836	15,3434	
35	23,4136	19,5205		6,7762	0,0000	17,1313	17,9103	3,1259	17,4163	
36	23,4136	3,3405		6,6391	0,0000	17,1313	17,9103	3,2884	17,4163	
37	21,8329	19,0752		19,0031		17,1313	1,8506	3,2884	19,5205	
38	5,3075	19,0752		19,0031		17,1313	3,6080	21,1073	19,5205	
39	17,4163	20,9062		17,4163		3,6755	3,6080	21,1073	17,9103	
40	17,4163	20,9062		3,8165		5,3075	3,6755	17,4163	2,0137	
41	15,8062	22,4949		5,6155		5,3075	3,6755	15,3434	3,7755	
42	13,7332	22,4949		5,6155		6,7762	3,8165	17,4163	3,7755	
43	13,7332	24,0631		5,4751		6,7762	3,8165	17,4163	17,1313	
44	13,7332	24,0631		13,7332		8,4445	3,6755	5,1684	17,1313	
45	5,4751	24,0631		13,7332		8,4445	13,7332	3,6755	5,4277	
46	5,4277	24,0631		15,0583		6,5017	13,7332	5,4751	22,4949	

47	1,8506	7,3869		15,0583		19,0031	15,0583	5,4751	22,4949	
48	0,0000	5,7938		13,7332		19,0031	15,0583	13,7332	20,6685	
49	0,0000	6,6937		15,8062		15,3434	17,1313	13,7332	20,6584	
50		6,6937		15,8062		13,7332	17,1313	5,6312	3,8165	

Verifica-se com este método de optimização que as soluções encontradas oscilam mais, em termos de convergência, realizando mais passos de simulação que o Hill-Climbing.

Repara-se que em ambos os algoritmos existem valores de convergência idênticos ao longo das simulações, o que pressupõe o teste da mesma solução de parâmetros. A solução passaria pela existência de uma lista de soluções testadas, de forma a ter isso em conta, aquando da criação de nova solução, ganhando tempo de processamento.

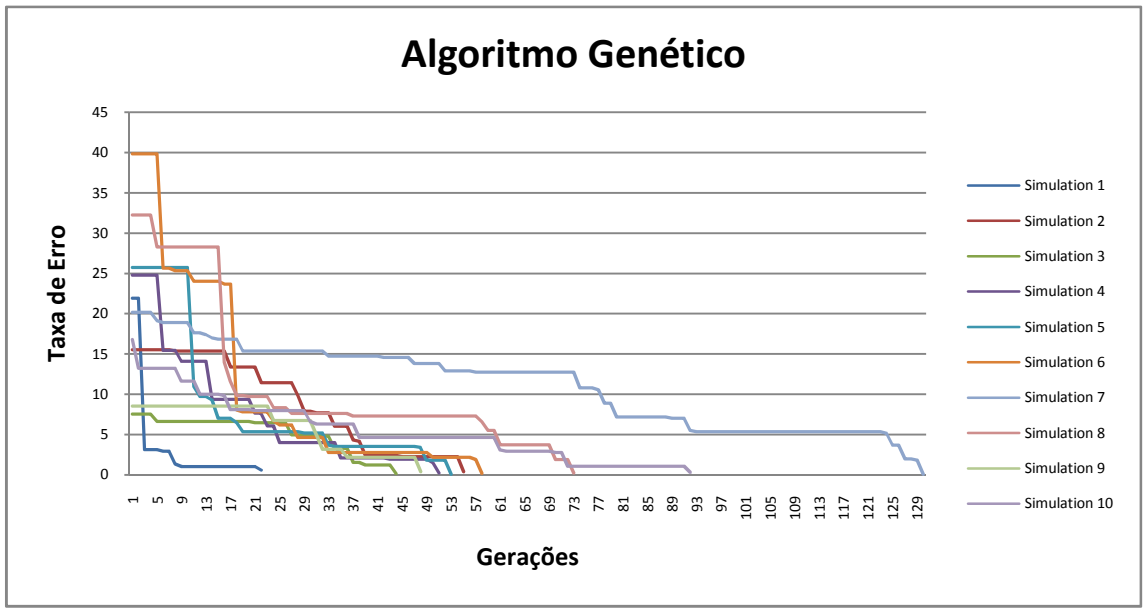

Gráfico 5. Gráfico Simulação Algoritmo Genético (Pequenas Mutações) para modelo Predador-Presa

Este tipo de algoritmo, tem uma filosofia diferente dos algoritmos anteriores, sendo a geração (Pai-Filho) o conceito fundamental em torno do algoritmo. Para este teste, foram consideradas gerações de quatro filhos, sendo em cada geração escolhido como melhor elemento, o que tiver taxa de erro inferior. Pela análise do gráfico 5, podemos constatar em todas as simulações realizadas, que existem sempre alguns valores de erro constantes consecutivos, o que pressupõe que a melhor solução prevalece com o avançar das gerações, não tendo o cruzamento de soluções ou a mutação de uma solução produzido melhores resultados. Neste algoritmo, as mutações inseridas foram

de valores vizinhos da solução escolhida aleatoriamente, sendo apenas adicionado o valor do Passo para cima ou para baixo do valor de referência.

Tabela 7. modelo Predador-Presa com Algoritmo Genético (pequenas mutações)

Genera-tions	Simula-tion 1	Simula-tion 2	Simula-tion 3	Simula-tion 4	Simula-tion 5	Simula-tion 6	Simula-tion 7	Simula-tion 8	Simula-tion 9	Simula-tion 10
0	21.9469	15.5283	7.5494	24.8073	25.7326	39.8630	20.1689	32.2713	8.5442	16.8121
1	21.9469	15.5283	7.5494	24.8073	25.7326	39.8630	20.1689	32.2713	8.5442	13.2541
2	3.1198	15.5283	7.5494	24.8073	25.7326	39.8630	20.1689	32.2713	8.5442	13.2541
3	3.1198	15.5283	7.5494	24.8073	25.7326	39.8630	20.1689	32.2713	8.5442	13.2541
4	3.1198	15.5283	6.6470	24.8073	25.7326	39.8630	19.1287	28.3145	8.5442	13.2541
5	2.9495	15.5283	6.6470	15.4829	25.7326	25.6746	18.8939	28.3145	8.5442	13.2541
6	2.9495	15.5283	6.6470	15.4829	25.7326	25.6746	18.8939	28.3145	8.5442	13.2541
7	1.3390	15.3660	6.6470	15.4829	25.7326	25.3378	18.8939	28.3145	8.5442	13.2541
8	1.0031	15.3660	6.6470	14.1153	25.7326	25.3378	18.8939	28.3145	8.5442	11.6497
9	1.0031	15.3660	6.6470	14.1153	25.7326	25.3378	18.8939	28.3145	8.5442	11.6497
10	1.0031	15.3660	6.6470	14.1153	10.9804	24.0415	17.6096	28.3145	8.5442	11.6497
11	1.0031	15.3660	6.6470	14.1153	9.7306	24.0415	17.6096	28.3145	8.5442	10.0229
12	1.0031	15.3660	6.6470	14.1153	9.7306	24.0415	17.3710	28.3145	8.5442	10.0229
13	1.0031	15.3660	6.6470	9.3874	9.3113	24.0415	17.0089	28.3145	8.5442	10.0229
14	1.0031	15.3660	6.6470	9.3874	7.0363	24.0415	16.8485	28.3145	8.5442	10.0229
15	1.0031	15.3660	6.6470	9.3874	7.0363	23.6988	16.8485	14.0117	8.5442	9.8652
16	1.0031	13.3988	6.6470	9.3874	7.0363	23.6988	16.8485	11.6114	8.5442	8.1351
17	1.0031	13.3988	6.6470	9.3874	6.4660	8.0650	16.8485	9.8742	8.5442	8.1351
18	1.0031	13.3988	6.6470	9.3874	5.3755	7.8136	15.3665	9.8742	8.5442	8.1351
19	1.0031	13.3988	6.6470	9.3874	5.3755	7.8136	15.3665	9.7583	8.5442	8.1351
20	1.0031	13.3988	6.4887	7.6686	5.3755	7.8136	15.3665	9.7583	8.5442	7.9787
21	0.5822	11.4601	6.4887	7.6686	5.3755	7.8136	15.3665	9.7583	8.5442	7.9787
22		11.4601	6.4887	6.0958	5.3755	7.8136	15.3665	9.7583	8.5442	7.9787
23		11.4601	6.4887	6.0958	5.3755	6.7106	15.3665	8.3440	6.7611	7.9787
24		11.4601	6.4887	4.0066	5.3755	6.1998	15.3665	8.3440	6.7611	7.9787
25		11.4601	6.4887	4.0066	5.3755	6.1998	15.3665	8.3440	6.7611	7.9787
26		11.4601	4.9795	4.0066	5.3755	6.1998	15.3665	7.6536	6.7611	7.9787
27		9.8010	4.9795	4.0066	5.3755	4.6369	15.3665	7.6536	6.7611	7.9787
28		7.8933	4.8191	4.0066	5.2157	4.6369	15.3665	7.6536	6.7611	7.9787
29		7.8933	4.8191	4.0066	5.2157	4.6369	15.3665	7.6536	6.7611	6.6470
30		7.7329	4.8191	4.0066	5.2157	4.6369	15.3665	7.6536	4.9870	6.3302
31		7.7329	4.8191	4.0066	5.2157	4.6369	15.3665	7.6536	3.1757	6.3302
32		7.7329	4.8191	4.0066	3.6912	2.7934	14.7202	7.6536	3.1757	6.3302
33		6.0517	3.2884	4.0066	3.5293	2.7934	14.7202	7.6536	3.1757	6.3302
34		6.0517	3.2884	2.1152	3.5293	2.7934	14.7202	7.6536	3.1757	6.3302
35		6.0517	3.2884	2.1152	3.5293	2.7934	14.7202	7.6536	2.2163	6.3302

36		4.3466	1.5732	2.1152	3.5293	2.7934	14.7202	7.3094	2.2163	6.3302
37		4.1817	1.5732	2.1152	3.5293	2.7934	14.7202	7.3094	2.2163	4.6586
38		2.4500	1.2436	2.1152	3.5293	2.7934	14.7202	7.3094	2.2163	4.6586
39		2.4500	1.2436	2.1152	3.5293	2.7934	14.7202	7.3094	2.2163	4.6586
40		2.4500	1.2436	2.1152	3.5293	2.7934	14.7202	7.3094	2.2163	4.6586
41		2.4500	1.2436	2.1152	3.5293	2.7934	14.5608	7.3094	2.2163	4.6586
42		2.4500	1.2436	1.9474	3.5293	2.7934	14.5608	7.3094	2.2163	4.6586
43		2.4500	0.1667	1.9474	3.5293	2.7934	14.5608	7.3094	2.2163	4.6586
44		2.2827		1.9474	3.5293	2.7934	14.5608	7.3094	2.2163	4.6586
45		2.2827		1.9474	3.5293	2.7934	14.5608	7.3094	2.2163	4.6586
46		2.2827		1.9474	3.5293	2.7934	13.8170	7.3094	2.2163	4.6586
47		2.2827		1.9474	3.4207	2.7934	13.8170	7.3094	0.4164	4.6586
48		2.2827		1.9474	1.8190	2.7934	13.8170	7.3094		4.6586
49		2.2827		1.5072	1.8190	2.1664	13.8170	7.3094		4.6586
50		2.2827		0.2504	1.8190	2.1664	13.8170	7.3094		4.6586

Com a mutação de pequenos valores, podemos observar na tabela 6, que a diferença entre as melhores soluções vai decrescendo lentamente, o que pressupõe acompanhar a tendência das pequenas variâncias nas soluções de cada geração. Outro factor a ter em conta, o número reduzido de filhos de cada geração (quatro), levando que passadas algumas gerações possam existir soluções duplicadas, devido ao cruzamento de soluções muito próximas.

Tal como aconteceu nos outros algoritmos, algumas simulações convergiram para a solução óptima depois da geração tida como limite. Todas as simulações realizadas com o algoritmo genético convergiram para soluções muito próximas, ou melhor dizendo, com taxas de erro inferiores a 1.

Um outro teste foi realizado, correr o mesmo algoritmo genético cinco vezes, sendo a mutação genética introduzida em cada geração, feita de forma aleatória. Como explicado no capítulo anterior, a solução a sofrer mutação é escolhida aleatoriamente do grupo de soluções da geração, assim como o parâmetro. A diferença reside em, em vez de ao valor ser adicionado ou subtraído (determinado aleatoriamente) o valor do Passo, este é escolhido aleatoriamente dentro do intervalo de valores definido inicialmente para o parâmetro.

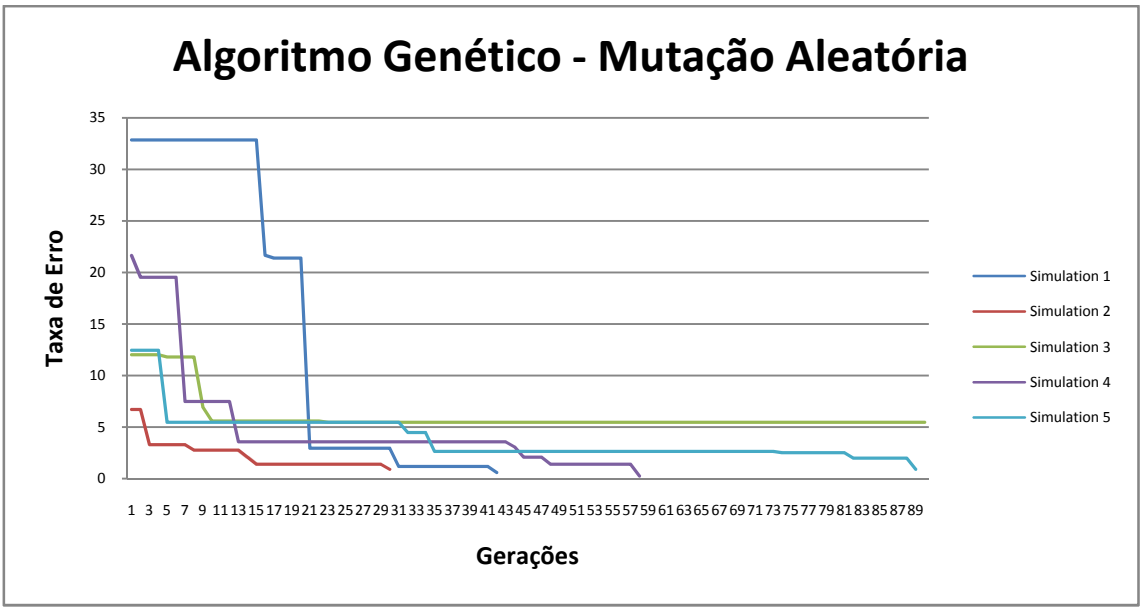

Gráfico 6. Gráfico Simulação Algoritmo Genético (Mutações Aleatórias) para modelo Predador-Presa

Pela análise do gráfico 6, podemos constatar que apenas uma das simulações não convergiu para a solução inicial, ficando "preso" a um mínimo local. Devido ao facto do valor da mutação ser aleatório, isso produz diferenças de erros entre gerações ligeiramente maiores do que com mutações pequenas. Apesar de se terem realizado apenas 5 simulações em vez das 10 definidas para os outros algoritmos, observa-se uma maior dificuldade em encontrar a solução inicial (ou óptima), em comparação com pequenas mutações.

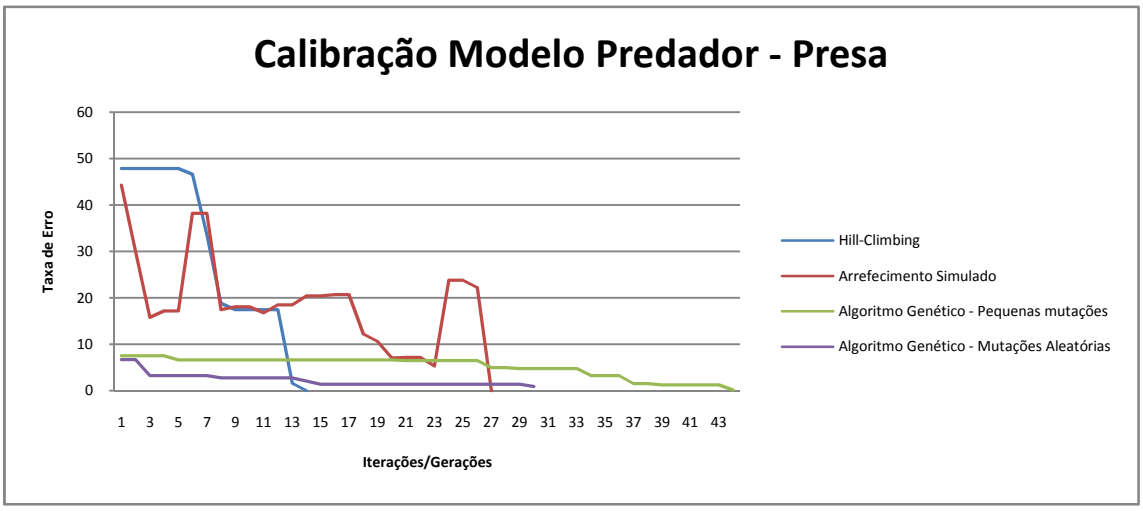

Gráfico 7. Comparação Melhores Resultados de cada Algoritmo: Modelo Predador-Presa

Em termos de conclusão para o modelo predador-presa, podemos constatar que apesar dos algoritmos Hill-Climbing e arrefecimento simulado aparentemente convergirem para a calibração em menos iterações, também conseguem ficar presos em mínimos locais, neste caso o Hill-Climbing.

Tabela 8. Tabela Comparativa das melhores Soluções encontradas Modelo Predador-Presa

	Solução	Erro
Solução Inicial	[0.3][0.01][0.1][0.1]	------
Hill-Climbing	[0.3][0.01][0.1][0.1]	0.00
Arrefecimento Simulado	[0.3][0.01][0.1][0.1]	0.00
Algoritmo Genético -Pequenas mutações	[0.3][0.02][0.1][0.1]	0.17
Algoritmo Genético - mutações aleatórias	[0.3][0.06][0.1][0.2]	0.91

O algoritmo genético permite encontrar com maior fiabilidade a solução óptima. Apenas o Arrefecimento Simulado apresenta comportamento oscilante em termos de taxa de erro, resultante da descida de Temperatura associado a probabilidade de aceitação da solução como melhor.

5.2 Modelo Baía Sangoo

A baía de Sangoo está localizada na província Shandong da República Popular da China (figura 16). Com uma área de 180km^2 e profundidades que oscilam até aos 20 metros, na costa. A baía tem sido explorada para aquacultura há mais de 20 anos (Guo et al., 1999), sendo as espécies mais cultivadas: algas marinhas (*Laminaria japonica*), ostras (*Crassostrea gigas*) e vieiras (*Chlamys farreri*). Vieiras e ostras são cultivadas em redes circulares tipo lanternas, enquanto as algas marinhas, suspensas por cordas. Um dos principais limites no cultivo dos bivalves naquela região, é a alta taxa de mortalidade da espécie vieira. Mortalidades elevadas no Verão, nos últimos anos, levaram a alterações na prática de aquacultura, incluindo mudança no período de plantação.

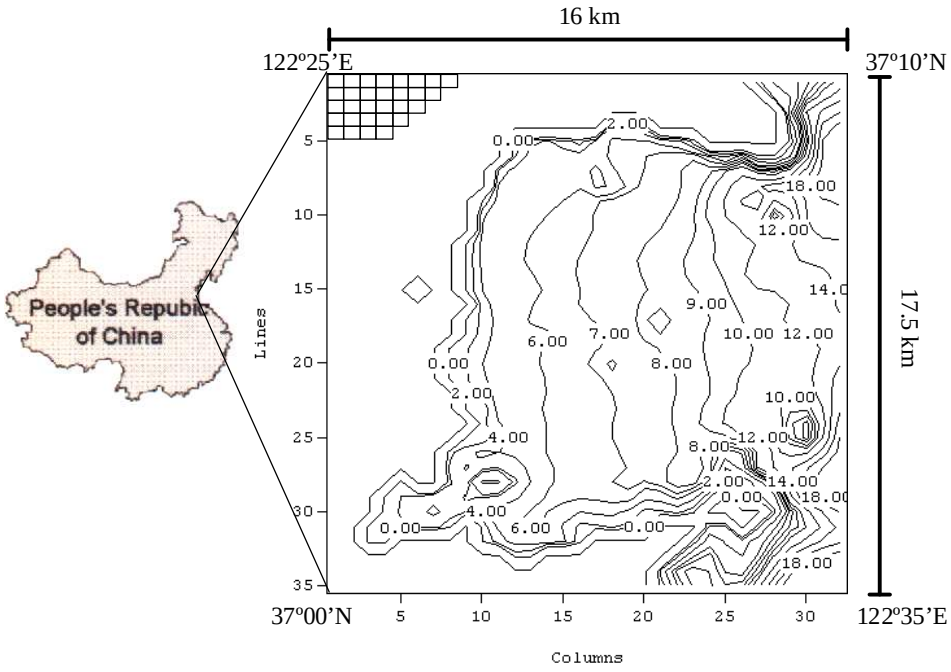

Figura 16. *A localização da Baía de Sangoo, incluindo o modelo do domínio e batimetria (bathymetry) (m).*

A figura 16 mostra a localização da baía de Sangoo e o modelo de batimetria respectivo. O modelo construído é composto por uma grelha de células com uma resolução de 500m. O modelo é bastante detalhado, sendo representado por 9 classes, as quais possuem 146 parâmetros..

Uma das diferenças face ao modelo analisado anteriormente além da complexidade em termos de modelo, reside no facto de alguns destes parâmetros serem valores constantes, pelo que não devem ser considerados para o processo de calibração. Exemplo disso, o valor da gravidade, ou o valor da latitude/longitude da baía. Estes valores devem permanecer inalteráveis nas soluções testadas, garantindo a coerência e integridade das fórmulas matemáticas representadas pelo modelo. Outra diferença assinalável, diz respeito ao espaço de simulação do modelo. O modelo predador-presa era representado apenas por uma dimensão espacial (representado apenas por uma única célula), enquanto este modelo mais realista e próximo dos utilizados em simulações ecológicas, possui 3 dimensões espaciais (*x,y,z*), sendo a coordenada *z* correspondente à profundidade do modelo.

Estas características do modelo levaram numa primeira fase à identificação dos parâmetros cujo valor não deveriam ser alterados pela calibração e numa segunda fase, definição de um sub-modelo com um menor número de classes e parâmetros para realização da experiência.

A experiência foi realizada seguindo os mesmos moldes do modelo predador-presa, tendo sido definido um período de simulação (desde 29-01-1997 16:00 até às 18:00) com um passo de simulação de 60 segundos, o que corresponde a 120 iterações com o simulador.

Tabela 9. Lista de Classes e Parâmetros do Modelo Sangoo para Simulação

Class Model	Parameter Name	Parameter Value	Constant
TTideWithWanContinuousHarmonics	Start Year	1997	Yes
TBiDimensionalSango3	Gravity	9.8	Yes
TBiDimensionalSango3	Difusion coeficient	100	No
TBiDimensionalSango3	Start Year	1997	Yes
TBiDimensionalSango3	Latitude	37	Yes
TBiDimensionalSango3	Northern tidal reference	735	No
TBiDimensionalSango3	Southern tidal reference	351	No
TBiDimensionalSango3	Critical depth for land boundary	0	No
TBiDimensionalSango3	Tidal Phase Lag	0	No
TBiDimensionalSango3	Coeficiente de Manning	0.03	No
TBiDimensionalSango3	Difusion coeficients in W-E direction	0	No
TBiDimensionalSango3	Difusion coeficients in N-S direction	0	No

5.2.1 Resultados

De forma a optimizar o processo de calibração, executaram-se inicialmente algumas simulações com todos os parâmetros em calibração, registo dos valores para todas as variáveis e em toda a extensão (todas as células) do modelo, para o intervalo de tempo definido anteriormente. Devido ao excesso de informação a transmitir pelo simulador, o agente de calibração apresentava algumas vezes erros de falta de memória ou processamento, tornando difícil inquirir sobre os seus resultados. Optou-se por simular apenas para uma célula, e calibrando apenas as classes associadas à hidrodinâmica do modelo (tabela 9). Com estas 2 alterações, o volume de dados reduziu-se drasticamente,

possibilitando analisar os resultados. Verificou-se que algumas variáveis não apresentavam qualquer alteração no valor ao longo da simulação, pelo que poderiam ser descartadas da lista de variáveis a simular.

Foram realizadas baterias de 5 simulações para cada algoritmo de calibração, tendo sido registado os resultados da taxa de erro por variável/iteração. O critério de paragem foi diminuído de 1 para 0,001.

Tabela 10. Exemplo de resultados de Simulação para o Modelo Sangoo

Step	Time	Tidal height	U Flow	V Flow	Dynamic height	Salinity	U Velocity	V Velocity	Water density	Drag coefficient	Box depth	Mean U Flow	Mean V Flow	Mean U Velocity	Mean V Velocity
1	85455360	2696.812475	0	0	1.716	0	0	0	1020.4449	0.003515	15.803813	0	0	0	0
2	85455369	2535.781141	-343.118189	44.333001	1.792194	0	-0.045889	0.005645	1026.75	0.003509	15.880007	-341.682359	0	-0.045889	0.005645
3	85455378	2541.589174	-693.407954	45.77663	1.856353	0	-0.092418	0.005803	1026.75	0.003504	15.944166	-342.400274	22.787846	-0.069154	0.005724
4	85455387	2547.314558	-2475.009437	46.268896	2.201711	0	-0.32326	0.005744	1026.75	0.003479	16.289524	-1036.730244	30.450774	-0.153856	0.005731
5	85455396	2552.957185	-4365.853925	-100.527017	2.48158	0	-0.561232	-0.012254	1026.75	0.00346	16.569393	-1396.300043	-1.839805	-0.2557	0.001234
6	85455405	2558.515668	-6628.595054	-266.49002	2.818281	0	-0.832513	-0.031828	1026.75	0.003437	16.906094	-2412.272475	-21.577247	-0.371063	-0.005378
7	85455414	2563.989922	-8893.112718	-424.561111	3.138389	0.000719	-1.093244	-0.049764	1026.75	0.003415	17.226202	-3114.992905	-87.425835	-0.491426	-0.012776
8	85455423	2569.379437	-7552.793762	-532.958032	2.87288	0.000568	-0.939204	-0.063454	1026.75	0.003433	16.960693	-3761.430873	-135.588017	-0.555394	-0.020015

A tabela 10 apresenta valores para as variáveis que são influenciadas ao longo do tempo pelos parâmetros.

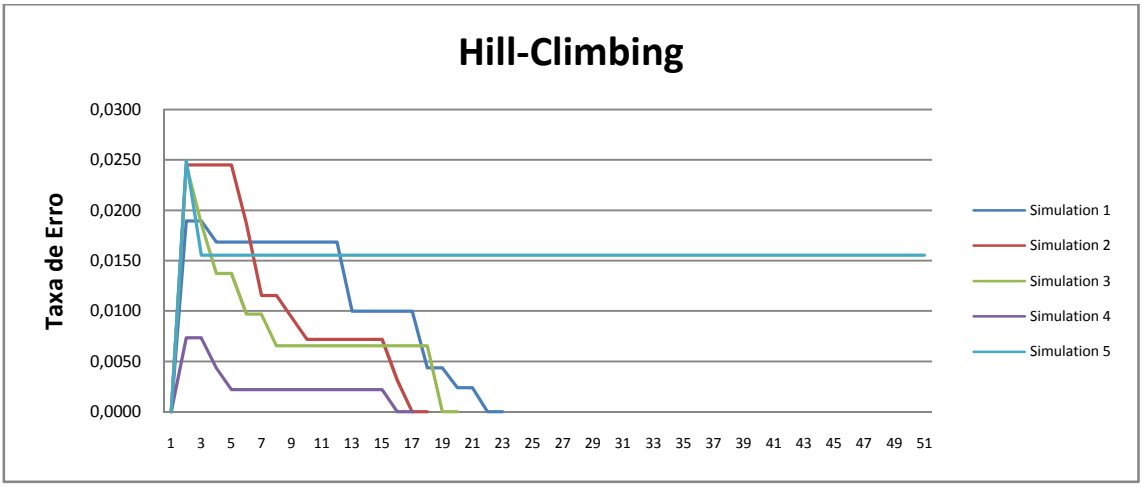

Gráfico 8. Resultados calibração Modelo Sangoo para Algoritmo Hill-Climbing

A primeira observação que se constatou pela análise dos resultados aplicando o algoritmo Hill-Climbing, diz respeito aos valores de erro muito próximos de zero, sendo as alterações introduzidas nas soluções vizinhas com pouco impacto na taxa de erro. O mesmo se observou com os outros algoritmos de calibração.

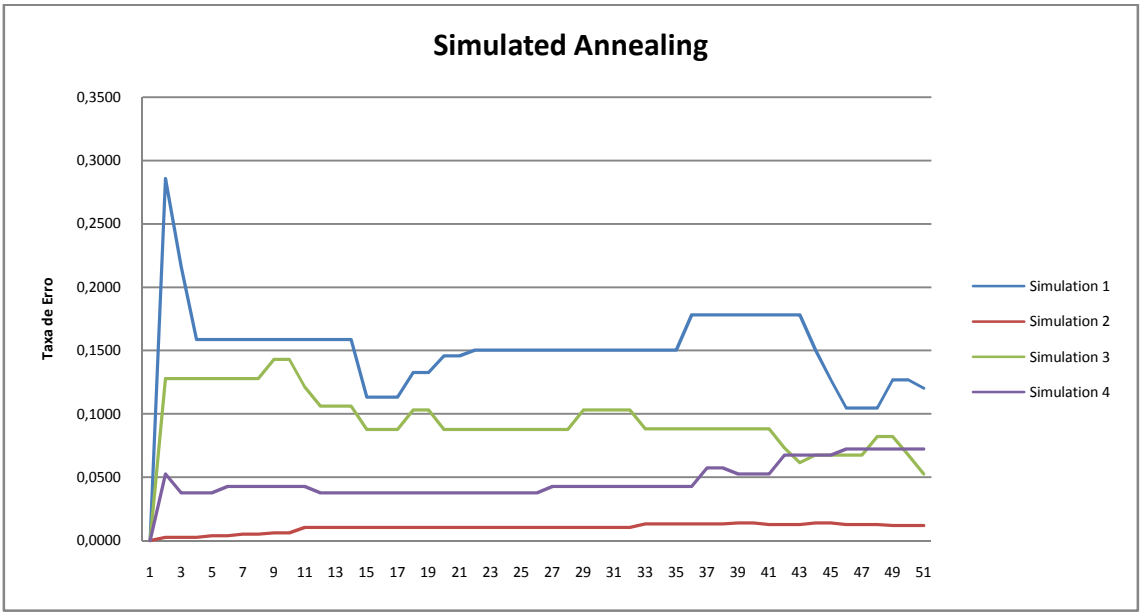

Gráfico 9. Resultados Calibração Modelo Baía Sangoo – Arrefecimento Simulado

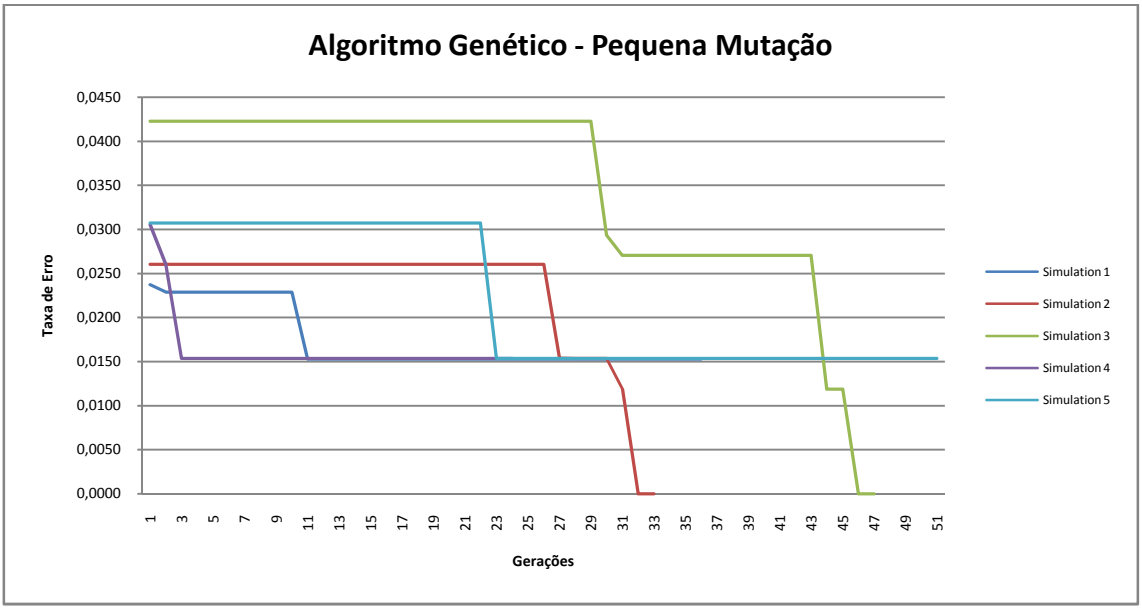

Gráfico 10. Resultados Calibração Modelo Baía Sangoo – Algoritmo genético com pequena mutação

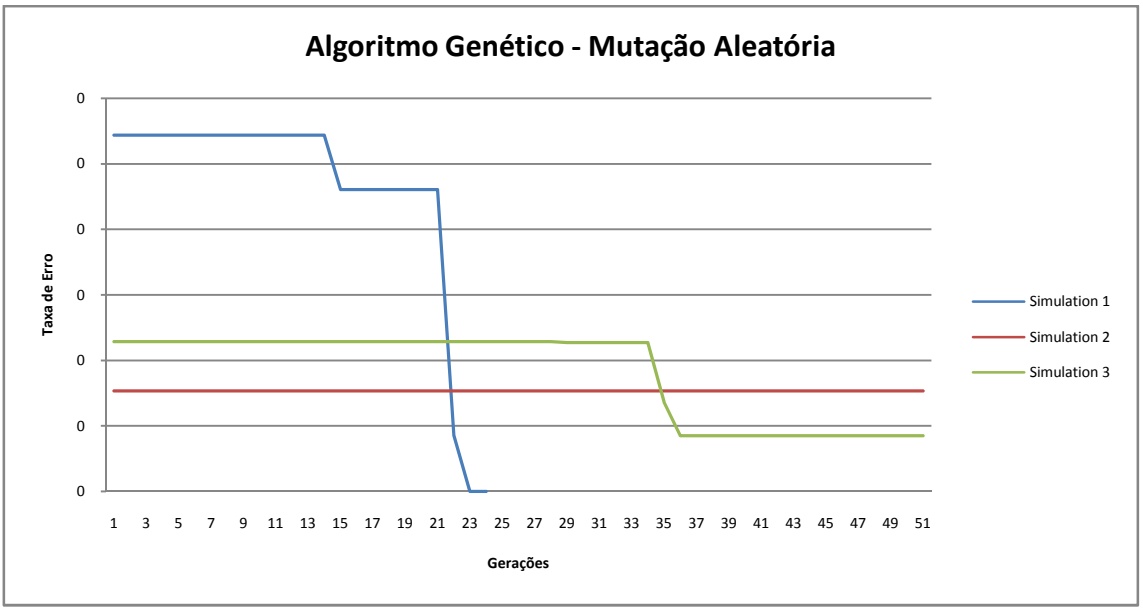

Gráfico 11. Resultados Calibração Modelo Baía Sangoo – Algoritmo genético com mutação aleatória

Tal facto pode ser explicado de 2 formas: a primeira diz respeito ao valor do *step* definido para cada parâmetro, que pode ser pouco significante para os resultados finais. A segunda diz respeito à função objectivo e à definição de pesos para cada variável. No modelo predador-presa, a granularidade das variáveis era quase idêntica, o mesmo não se constata num modelo aquático, onde variáveis representando os valores das marés, têm uma granularidade diferente da variável que representa a taxa de absorção do *fitoplanton*, por exemplo.

A função objectivo deve reflectir tais diferenças, cabendo ao modelador indicar pesos para cada variável, de forma a homogeneizar os valores obtidos do simulador, produzindo diferenças significativas em termos de taxa de erro.

5.3 Conclusões

Neste capítulo foram apresentadas as experiências sobre o modelo ecológico predador-presa, para os algoritmos de optimização subida de colina, arrefecimento simulado e algoritmos genéticos. Os resultados são expressos pelo coeficiente de erro, valor dado pela diferença do resultado da simulação para a solução testada, com os resultados padrão. Apesar de se conseguir que o modelo convirja para a solução óptima, os mínimos locais são um problema a ter em conta, assim como a noção de sensibilidade dos valores dos parâmetros do modelo.

Pela análise dos resultados, pensamos que o processo de calibração deve focalizar a sua atenção na compreensão dos valores dos parâmetros, seus valores mínimos e máximos, assim como o valor mínimo (unidade) a indicar em cada solução de valores. A definição dos intervalos de valores para cada parâmetro, assim como do valor de unidade (*step*) é de extrema importância no processo de calibração, visto aumentar o espaço de soluções possíveis.

Além do número de soluções para simulação, caso os valores atribuídos aos parâmetros sejam desprovidos de algum cuidado, por exemplo em termos de unidade, ou dos limites, o simulador irá produzir resultados incoerentes, e em alguns casos, leva mesmo a erros de sistema, devido ao uso errado de valores de parâmetros nas fórmulas matemáticas.

Capítulo 6

6. Conclusões e Perspectivas de Desenvolvimento

A calibração de modelos é realizada através da comparação dos valores observados com os simulados, sendo uma fase crucial no processo de modelação. Devido ao seu carácter iterativo, após cada simulação, o perito (ou modelador) analisa o resultado, realizando as alterações nos valores dos parâmetros, tentando fazer convergir os resultados. Necessita de possuir um bom conhecimento dos efeitos desses valores no modelo de simulação, assim como conhecer bem as entidades e suas relações, modeladas. Dependendo do tipo do modelo e sua complexidade, é um processo moroso e de tentativa-erro.

Este processo é particularmente trabalhoso e consumidor de tempo na área da simulação de modelos ecológicos, onde as propriedades físicas, químicas e processos biológicos são combinados e os valores de vários parâmetros, que integram as funções dos processos, são apenas estimados e podem variar dentro de uma gama de valores comummente aceites pelos investigadores.

Um dos problemas encontrados no processo de optimização de parâmetros, foi na sensibilidade dos valores dos parâmetros, visto cada parâmetro possuir a sua dimensão e granularidade, sendo difícil saber o passo (*step*) necessário para produzir melhores soluções. Optou-se por definir limites em termos percentuais tendo como referência os valores padrões dos parâmetros, fornecidos pelo simulador, ao carregar o modelo. Ainda necessita da intervenção e conhecimento de um perito, para inicialização correcta do passo assim como estabelecer os limites para cada parâmetro. O estabelecimento de limites para cada parâmetro é fulcral no processo de calibração, pois caso se utilize

valores altos, provoca erro nos resultados da simulação, devido a dificuldade de expressão (*overflow* das variáveis de programação).

Os modelos utilizados pelo simulador EcoDynamo, são determinísticos, não sendo necessária a alteração dinâmica dos valores dos parâmetros ao longo do tempo, tal como acontece nos modelos com características estocásticas. É possível realizar várias simulações com as mesmas características do modelo, obtendo sempre os mesmos resultados. Em termos do processo de calibração, os modelos determinísticos permitem focalizar nos métodos de optimização, e suas estratégias na definição de novas soluções de parâmetros.

O uso de um agente de calibração na optimização de modelos, permite automatizar um processo algo complexo e tedioso de se resolver manualmente, sem ser necessário realizar alterações a nível do código da aplicação de simulação. Apesar do projecto proposto não possuir as características intrínsecas de um agente inteligente, apenas a comunicação com o simulador, via mensagens ECOLANG, este pode com alguma facilidade adaptar mecanismos de inteligência, visto a sua orgânica ser independente do núcleo de simulação.

O desenvolvimento deste projecto abriu diversas perspectivas de trabalho futuro, incluindo:

- Implementação de novos métodos de optimização e seu teste no âmbito do agente de calibração implementado;
- Inclusão de potencialidades de selecção automática do método de calibração no agente desenvolvido;
- Realização de um conjunto mais alargado de experiências utilizando outros modelos ecológicos.

O agente de calibração desenvolvido constitui uma boa base para a implementação destes e de outros desenvolvimentos.

Referências Bibliográficas

[Amirjanov, 2006] A. Amirjanov, 2006. The development of a changing range genetic algorithm. Computer Methods in Applied Mechanics & Engineering. 195:19-22. pp. 2495-2508.

[Baretta et al, 1988] Baretta and Ruardij, 1988 J.W. Baretta and P. Ruardij, Tidal flat estuaries, simulation and analysis of the Ems Estuary, Ecological Studies vol. 71, Springer-Verlag (1988).

[Barteneva et al., 2006] Daria Barteneva, Nuno Lau, Luis Paulo Reis, 2006. Implementa-tion of Emotional Behaviors in Multi-Agent System using Fuzzy Logic and Tempera-mental Decision Mechanism, Proceedings of EUMAS 2006, Fourth European Work-shop on Multi-Agent Systems, Lisbon, Portugal.

[Barteneva et al., 2007] Daria Barteneva, Luis Paulo Reis & Nuno Lau, 2007. Bilayer Agent-Based Model Of Social Behavior: How Temperament Influence On Team Performance, In: I. Zelinka, Z. Oplatková and A. Orsoni (eds), Proceedings of the 21st European Conference on Modelling and Simulation, pp. 181-187, Prague, Czech Republic. ISBN: 0-9553018-2-3 (978-0-9553018-2-7).

[Brito et al., 2001] Brito, António E. S. Carvalho, Teixeira, J. Manuel Feliz; *Simulação por Computador – Fundamentos e Implementação de código em C e C++*; Publindústria, Edições Técnicas, Porto, Junho 2001.

[Cardon et al., 2000] A. Cardon, T. Galinho &J.-P. Vacher, 2000. Genetic algorithms using multi-objectives in a multi-agent system. Robotics and Autonomous Systems. 33:2-3. Pp. 179-190.

[Carpenter, 1989] Carpenter, S. R. 1989. Replication and treatment strength in whole-lake experiments. Ecology 70: 453-463.

[Cruz et al., 2007] Filipe Cruz, António Pereira, Pedro Valente, Pedro Duarte & Luís Paulo Reis, 2007. Intelligent Farmer Agent for Multi-Agent Ecological Simulations Opti-mization. In: J. Neves, M. Santos and J. Machado (eds): EPIA 2007, LNAI 4874, pp.593-604. Springer-Verlag, Berlin Heidelberg. ISBN: 978-3-540-77000-8 (978-3-540-77002-2_50)

[Cruz, 2006] Filipe Cruz, 2006. Desenvolvimento de Agente Aquicultor Inteligente, Relatório de Estágio/Projecto de Fim de Curso em Engenharia Informática e Computação, Faculdade de Engenharia da Universidade do Porto.

[Duarte et al., 2003] Pedro Duarte, R. Meneses, A.J.S. Hawkins, M. Zhu, J. Fang, & J. Grant, 2003. Mathematical modelling to assess the carrying capacity for multi-species culture within coastal water. Ecological Modelling, Vol. 168, pp. 109-143.

[Duarte et al., 2006] Pedro Duarte, B. Azevedo, M. Guerreiro, C. Ribeiro, R. Bandeira, A. Pereira, M. Falcão, D. Serpa, & J. Reia, Biogeochemical Modelling of Ria Formosa (South Portugal). Hydrobiology (in press).

[Duarte et al., 2006] Pedro Duarte, M.J.Guerreiro, J.Reia, L.Cancela da Fonseca, A.Pereira B.Azevedo, M.Falcão, D.Serpa, C.Ribeiro & R. Bandeira, 2006. Ferramentas de gestão de zonas costeiras: Aplicação à Ria Formosa (Sul de Portugal). Proceedings do 2º Seminário sobre Sistemas Lagunares Costeiros, Vila Nova de Santo André.

[Duarte et al., 2007a] Pedro Duarte, Bruno Azevedo, C. Ribeiro, António Pereira, Manuela Falcão, Dalila Serpa, Rui Bandeira & João Reia, 2007. Management oriented mathematical modelling of Ria Formosa (South Portugal). Transitional Water Monographs. 1:1. pp. 13-51.

[Duarte et al., 2007b] Pedro Duarte, Maria J. Guerreiro, João Reia, Luís Cancela da Fonseca, António Pereira, Bruno Azevedo, Manuela Falcão & Dalila Serpa, 2007. Gestão de zonas costeiras: aplicação à Ria Formosa (Sul de Portugal). Revista Ciência Agronômica, v.38, n.1, pp.118-128. ISSN 0045-6888.

Duarte et al., 2008] Pedro Duarte, B. Azevedo, M. Guerreiro, C. Ribeiro, R. Bandeira, A. Pereira, M. Falcão, D. Serpa, & J. Reia, Biogeochemical Modelling of Ria Formosa (South Portugal). Hydrobiologia, v.611, pp.115-132.

[Eykhoff, 1974] P. Eykhoff (1974), System Identification, J. Wiley, London.

[Falcão et al., 2003] M. Falcão, L. Fonseca, D. Serpa, D. Matias, S. Joaquim, P. Duarte, A. Pereira, C. Martins & M. J. Guerreiro, 2003. Synthesis report. EVK3-CT-20022-00084 (available at www.dittyproject.org).

[Ferreira et al., 1998] J. Ferreira, P. Duarte, & B. Ball, 1998. Trophic capacity of Carling-ford Lough for aquaculture - analysis by ecological modelling. Aquatic Ecology, Vol. 31 (4), pp. 361–379.

[Grimm et al., 2005] Grimm, V. and S.F. Railsback. Individual-based modeling and Ecology. Princeton University Press., Princeton, NJ, 2005.

[Guo et al., 1999] X. Guo, ,S. Ford & F. Zhang, 1999. Molluscan aquaculture in China. Journal of Shellfish Research, Vol. 18, pp. 19-32.

[Halton, 1970] Halton, J. H. (1970): Retrospective and prospective survey of the Monte Carlo method. Siam review, 12, 1.

[Hopcroft et al., 2001] Hopcroft, John E.; Rajeev Motwani, Jeffrey D. Ullman (2001). Introduction to Automata Theory, Languages, and Computation (2nd ed. ed.). Reading Mass: Addison-Wesley. ISBN 0201441241.

[Jørgensen & Bendoricchio, 2001] Jørgensen, S. E. and Bendoricchio, G.: Fundamentals of Ecological Modelling. Elsevier Science Ltd, 3rd edition. 2001.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C.D. Gelatt & M.P. Vecchi, 1983. Optimization by Simulated Annealing. Science, Vol. 220:4598, pp. 671-680.

[Mishra et al., 2005] N. Mishra, Prakash, M.K. Tiwari, R. Shankar & F.T.S. Chan, 2005. Hybrid tabu-simulated annealing based approach to solve multi-constraint product mix decision problem. Expert Systems with Applications. Vol 29:2. pp. 446-454.

[Naur 1960] NAUR, Peter (ed.), "Revised Report on the Algorithmic Language ALGOL 60.", *Communications of the ACM*, Vol. 3 No.5, pp. 299-314, May 1960.

[Norvig & Russel, 2003] P. Norvig & S.J. Russel. 2003. Artificial Intelligence: a Modern Approach, 2nd edition, Prentice Hall.

[Pegden et al., 1990] Pegden C.D., Shannon R.E., Sadowsky R.P. (1990) Introduction to Simulation Using SIMAN, McGraw-Hill, New York.

[Pereira et al., 2004] António Pereira, Pedro Duarte & Luís Paulo Reis. 2004. Agent-Based Simulation of Ecological Models. In: H. Coelho and B. Espinasse (eds), Proceedings of the 5th Workshop on Agent-Based Simulation, pp. 135-140, Lisbon, Portugal, May. ISBN: 3-639150-31-1.

[Pereira et al., 2005a] António Pereira, Luis Paulo Reis & Pedro Duarte, 2005. ECOLANG - A Communication Language for Simulations of Complex Ecological Systems. In Y. Merkuriev, R. Zobel and E. Kerckhoffs (eds.) Proceedings of the 19th European Con-ference Modelling and Simulation, ECMS'2005, pp. 493-500, Riga Latvia, 2005, ISBN: 1-84233-112-4

[Pereira et al., 2005b] António Pereira & Pedro Duarte, 2005. EcoDynamo – Ecological Dynamics Model Application. University Fernando Pessoa – Centre for Modelling and Analysis of Environmental Systems, Technical Report.

[Pereira et al., 2007] António Pereira, Pedro Duarte & Luís Paulo Reis, 2007. An Integrated Ecological Modelling and Decision Support Methodology. In: I. Zelinka, Z. Oplatková and A. Orsoni (eds), Proceedings of the 21st European Conference on Modelling and Simulation, pp.497-502, Prague, Czech Republic. ISBN: 0-9553018-2-3 (978-0-9553018-2-7).

[Pereira, 2008] António Pereira, 2008. ECOLANG - Communications Language for Eco-logical Simulations Network. Faculty of Engineering of University of Porto – LIACC – Artificial Intelligence and Computer Science Laboratory, Technical Report TR-LIACC-FEUP-AMCP 01.1.

[Reis & Lau, 2002] Luís Paulo Reis & Nuno Lau, COACH UNILANG – A Standard Lan-guage for Coaching a (Robo) Soccer Team, in Andreas Birk, Silvia Coradeschi and Satoshi Tadokoro, editors, RoboCup-2001: Robot Soccer World Cup V, Springer Verlag Lecture Notes in Artificial Intelligence, Vol. 2377, pp.183-192, Berlin, 2002

[Reis, 2003] Luís Paulo Reis, Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico, PhD Thesis/Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, June, 2003

[Restivo & Reis, 2006] André Restivo & Luis Paulo Reis, 2006. Clustering Agent Optimization Results in Dynamic Scenarios, Proceedings of EUMAS 2006, Fourth European Workshop on Multi-Agent Systems, Lisbon, Portugal.

[Restivo, 2006] André Restivo, 2006. Dynamic Scenario Simulation Optimization, MSc Thesis, Mestrado em Inteligência Artificial e Sistemas Inteligentes, Faculdade de Engenharia da Universidade do Porto.

[Sánchez-Mata et al. 1999]. A. Sánchez-Mata, M. Glémarec & J. Mora, 1999. Physico-chemical structure of the benthic environment of a Galician ría (Ría de Ares-Betanzos, north-west Spain). J. Mar. Biol. Ass. U.K. Vol. 79, pp. 1-21.

[Schriber, 1987] SCHRIBER, T. J. (1987). "The Nature and Role of Simulation inthe Design of Manufacturing Systems." *Simulation in CIM and Artificial Intelligence Techniques*(25): 5-18.

[Siarry et al., 1997] P. Siarry, G. Berthiau, F. Durdin & J. Haussy, 1997. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. ACM Transactions on Mathematical Software. Vol. 23:2, pp.209-228.

[Sutton & Barto, 1998] R.S. Sutton & A.G. Barto, 1998. Reinforcement Learning: An In-troduction. The MIT Press.

[Van der Tol et al., 1998] Van der Tol, M.W.M., Scholten, H., 1998. A model analysis on the effects of decreasing nutrients loads on the biomass of benthic suspension feeders in Oosterschelde ecosystem (SW Netherlands). Aquatic Ecology 31, 395-408.

[Vieira & Bordalo, 2000] M. Vieira & A.A. Bordalo, 2000. The Douro estuary (Portugal): a mesotidal salt wedge. Oceanologica Acta Vol. 23, pp. 585-594.

[Vreugdenhil, 1989] C. B. Vreugdenhil, 1989. Computational hydraulics, An introduction. Springer-Verlag, 183 pp.

[Watson e tal., 1996] Watson, R.T., Zinyowera, M.C. and Moss, R.H., 1996. Climate Change 1995 - Impacts, adaptations and mitigation of climate change, Scientific-Technical Analyses. Cambridge, UK: Cambridge University Press, 879 pp.

[Weiss, 1999] G. Weiss, 1999. ed., Multi-Agent Systems: A Modern Approach to Distri-buted Artificial Intelligence, MIT Press.

[Wooldridge, 2002] M. Wooldridge, 2002. An Introduction to Multi-Agent Systems, John Wiley & Sons, Ltd.

[Youssef et al., 2001] H. Youssef, S.M. Sait & H. Adiche, 2001. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Engineering Applications of Artificial Intelligence. Vol. 14:2, pp.167-181

Anexo 1- Manual do Utilizador

Esta secção introduz as interfaces do agente calibrador, explicando como se utiliza e configura o agente durante o processo de calibração de modelos ecológicos. Apesar do seu simples manuseio, carece de algum conhecimento do domínio, no que diz respeito à sensibilidade dos valores dos parâmetros, assim como à escolha das variáveis para calibração e às boxes a registar durante a simulação.

1.º passo: ligar ao servidor EcoDynamo

O primeiro passo na calibração do modelo ecológico passa pela ligação do agente calibrador ao simulador EcoDynamo. Pressupõem-se que a aplicação do simulador esteja a correr, assim como o modelo para simulação esteja definido.

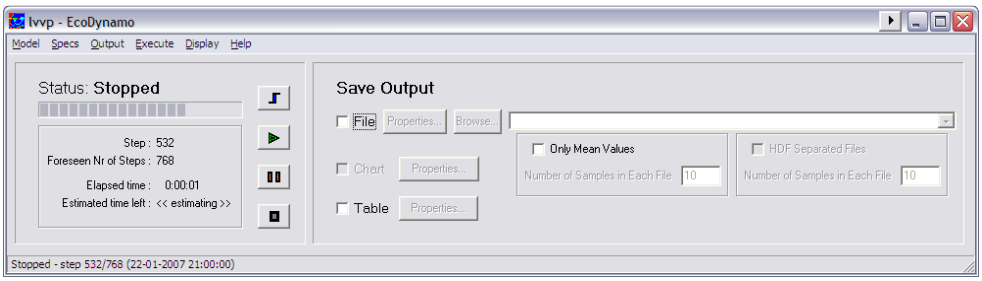

Figura 17. *Interface base do EcoDynamo.*

A figura 17 representa a interface do EcoDynamo para o modelo predador-presa (lvvp). O modelo está devidamente inicializado, podendo o utilizador parametrizar a simulação, sendo esses valores adquiridos pelo agente de calibração.

2.º passo: ligar aplicação – Agente Calibrador

O agente calibrador não realiza a simulação, apenas envia mensagens ECOLANG para o simulador, com instruções e novos valores de parâmetros e trata as mensagens que este envia de volta – Resultados das variáveis para a simulação.

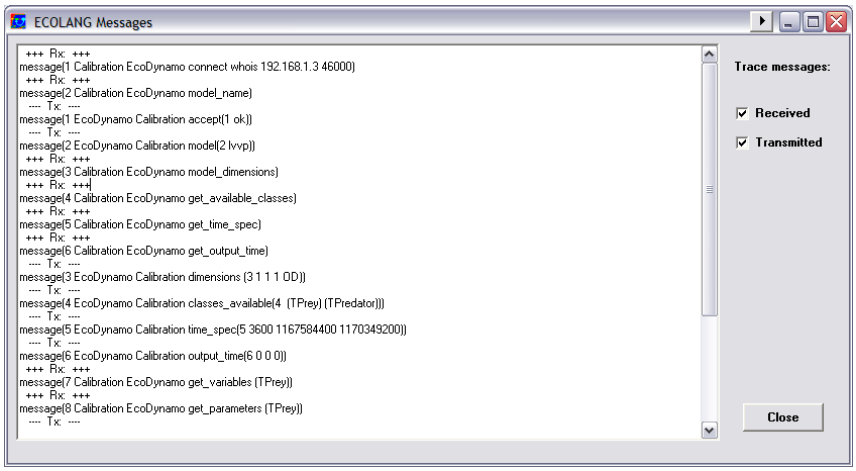

Figura 18. *Mensagens ECOLANG.*

Quando a aplicação do agente calibrador arranca, envia mensagem ao EcoDynamo, registando-se na lista de agentes ligados ao simulador para uma determinada porta de computador. As mensagens seguintes, após o simulador ter enviado mensagem de aceitação, são referentes às características do modelo em simulação (nome do modelo, dimensões do modelo, classes disponíveis, tempo de simulação, parâmetros e variáveis).

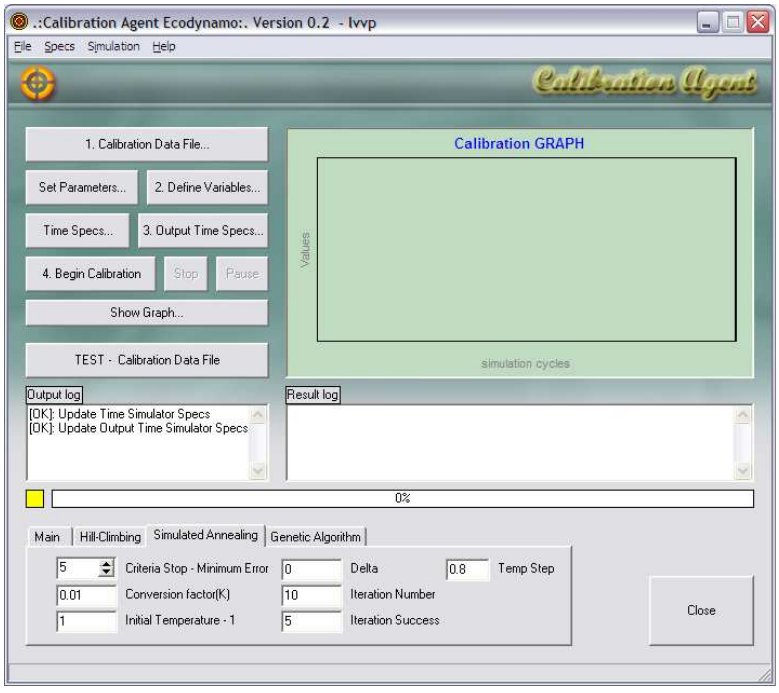

Figura 19. *Interface do agente calibrador.*

Quando aparece a interface do agente calibrador (figura 19), consegue-se identificar o modelo para calibração na barra de topo da janela. A interface está dividida em vários segmentos: definição das características de calibração, comandos de acções para a

calibração, output de resultados do processo de calibração, gráfico de resultados e secção para parametrização dos algoritmos de calibração em uso.

3.º passo: escolha do ficheiro dados Reais

– não esta implementado ainda. Por agora o agente corre a primeira simulação e assume esses valores como os de comparação. A 2.ª simulação é com valores de parâmetros todos aleatórios… em termos de agente de calibração o resultado é o mesmo…pois os algoritmos de optimização necessitam de convergir os resultados. Na realidade não existem ficheiros com dados reais estabelecidos, pelo que se simula um período em qualquer modelo e guarda-se os resultados, em ficheiro com tabulações.

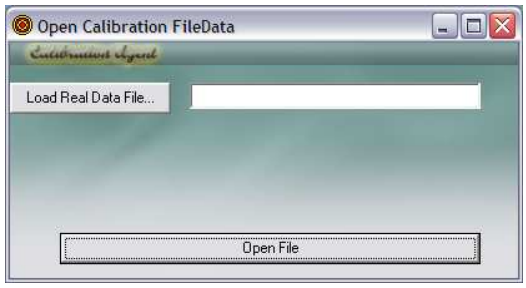

Figura 20 Interface Escolha Ficheiro dados Reais

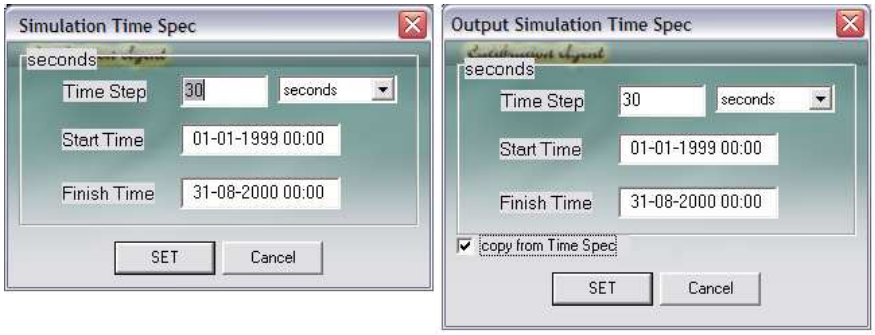

Figura 21. Interface Escolha tempos Simulação

4.º passo: definição do período de simulação e do periodo que se requisita que os dados sejam devolvidos do simulador (normalmente tenho utilizado os mesmos valores) – time spec e output time specs. Assim como o step de simulação. Nota: por exemplo, o modelo Sangoo, necessita forçosamente do step de 30 seg, pois os organismos do modelo assim exigem.

5.º passo: selecção das variáveis do modelo

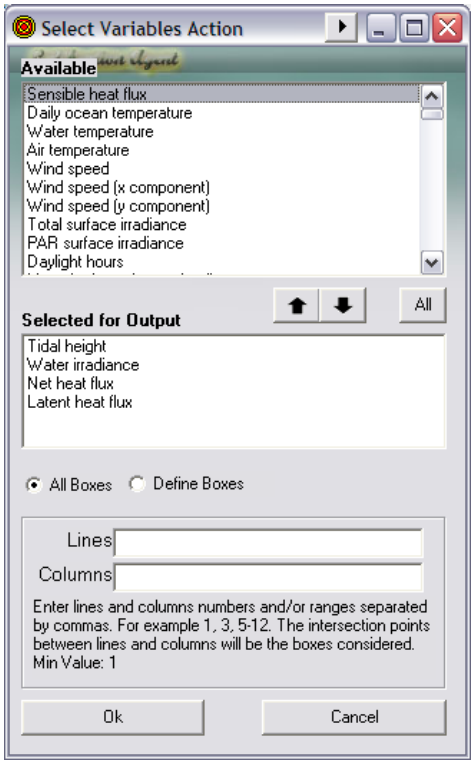

Figura 22 – interface Escolha variáveis simulação

Selecção das variáveis do modelo a ter em conta para calibração, assim como a área a simular. Por defeito o agente calibrador, aplica a todas as boxes. Aqui se calhar também se poderia colocar algum tipo de sinal, para correr simulações para cada variável afim de verificar se o seu valor se altera.

6.º passo: alterar/observar os valores dos parâmetros

por classe de modelo. Por defeito utiliza em termos de arranque os valores obtidos no modelo. Mas possibilita alteração desses valores caso seja necessário.

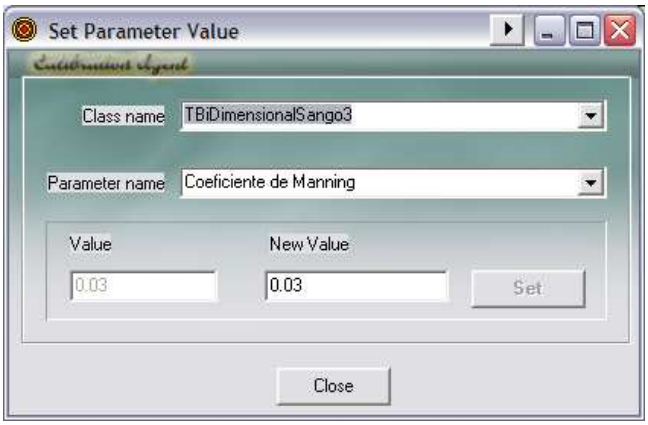

Figura 23 – interface para gestão Parâmetros

7.º passo: escolha do nome do ficheiro de output

Escolha do nome do ficheiro de output a criar como output dos resultados. A representação depende se se coloca o visto na caixa “save to file”, assim como do algoritmo de optimização utilizado.

8.º passo: escolha do algoritmo de calibração

Escolha do algoritmo de calibração e personalização dos campos associados ao algoritmo. Tem um tabulador para cada algoritmo. Por defeito ja aparecem valores listados.

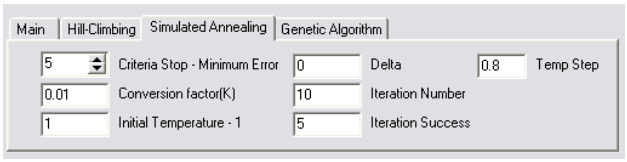

Figura 24 – interface definição características métodos optimização

9.º Passo: Início processo de calibração

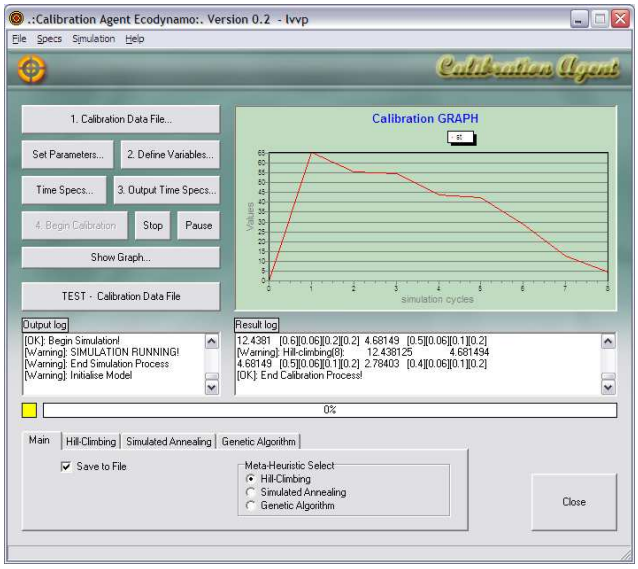

Figura 25 – Interface Global Agente Calibrador

– carregar no botão "begin calibration". Quando a calibração começa, é possível ver algumas mensagens de interacção com o servidor, ou resultados dos algoritmos – caixa de Output log e Result log. A informação da caixa de output, diz respeito ao processo de simulação. A caixa de result log, diz respeito aos resultados dos algoritmos em cada simulação. É possível observar no gráfico o valor da função de avaliação ao longo do tempo, ou melhor…verificar no gráfico as melhor soluções encontradas ao longo do tempo. A área do gráfico possui funcionalidades para realizar zoom e deslocação. Caso seja necessário, existe um botão para mostrar um gráfico em tamanho maior. Quando os critérios de paragem de calibração são atingidos (o valor da função de avaliação é inferior ou igual ao estipulado pelo utilizador, ou se ultrapassou o número de simulações definidas), é apresentado uma mensagem na caixa de Result log, a indicar o fim da calibração e o conjunto de valores dos parâmetros com o respectivo valor de erro. É possível recomeçar a simulação a partir desse ponto, colocando outra linha no gráfico, para ser mais fácil a comparação.

Anexo 2- Lista de Classes

Descrição da estrutura interna das classes utilizadas neste projecto.

Svariables Class

Variáveis

- STRING NAME;
- DOUBLE VALUE;

Construtor

- SVARIABLES(CHAR*);
- SVARIABLES(CHAR*,DOUBLE);

Métodos

- STRING GETNAME();
- DOUBLE GETVALUE();
- VOID SETVALUE(DOUBLE);

SParameter Class

Variáveis

- STRING NAME;
- DOUBLE VALUE;
- STRING CLASSNAME;
- VECTOR <DOUBLE> DOMAINVALUES;
- DOUBLE NEWVALUE;
- DOUBLE STEP; //STEP FOR EACH PARAMETER VALUE...
- DOUBLE MIN_VALUE; //MIN PARAMETER VALUE
- DOUBLE MAX_VALUE; //MAX PARAMETER VALUE

Construtor

- SPARAMETERS();
- SPARAMETERS(CHAR*,DOUBLE);
- SPARAMETERS(STRING,DOUBLE,STRING);

Métodos

- VOID GENERATERANDOMVALUES();
- STRING GETNAME();
- STRING GETCLASSNAME();
- DOUBLE GETVALUE();
- DOUBLE GETNEWVALUE();
- VOID SETNEWVALUE(DOUBLE);

- DOUBLE RANDOMVALUE();//CALCULATE RANDOM VALUE
- DOUBLE GETRANDOMVALUES(INT POSITION);
- INT GETCOUNTRANDOMVALUES(); //RETURN THE SIZE OF RANDOM VALUES ARRAY

ModelClass Class

Variáveis

- STRING CNAME; //NAME OF MODEL CLASS
- INT CNUMMESSAGEP; //ECDP MESSAGE ID FOR CLASS PARAMETER ASK
- INT CNUMMESSAGEV; //ECDP MESSAGE ID FOR CLASS VARIABLE ASK
- VECTOR <SVARIABLES> CVARIABLE; //STRUCT OF CLASS VARIABLES
- VECTOR <SPARAMETERS*> CPARAMETER; //STRUCT OF CLASS PARAMETER

Construtor

- MODELCLASS(STRING, INT, INT);
- MODELCLASS(STRING);

Métodos

- VOID SETVARIABLES(QUEUE*);
- VOID SETPARAMETERS(QUEUE*);
- VECTOR <SVARIABLES> GETVARIABLES();
- VECTOR <SPARAMETERS*> GETPARAMETERS();
- INT GETNUMMESSAGEPARAMETER(); //RETURN ECDP MESSAGE ID
- INT GETNUMMESSAGEVARIABLE(); //RETURN ECDP MESSAGE ID
- VOID SETNUMMESSAGEPARAMETER(INT); //ADD ECDP MESSAGE ID
- VOID SETNUMMESSAGEVARIABLE(INT); //ADD ECDP MESSAGE ID
- STRING GETCLASSNAME(); //RETURN CLASS NAME

TimeVariables Class

Descrição: esta classe permite guardar para cada célula (box) do modelo de simulação o valor das variáveis.

Variáveis

- INT BOX; //DEFINE THE CELL SIMULATION
- INT REGINDEX;
- INT REGTIME;
- VECTOR <SVARIABLES> VARIABLES;

Construtor

- TIMEVARIABLES(INT, INT, INT);
- TIMEVARIABLES(INT, INT, INT, SVARIABLES);

Métodos

- VOID SETVARIABLES(SVARIABLES);
- INT GETBOX();

- int getRegIndex();
- int getRegTime();
- vector <SVariables> getVariables(); //return all SVariable objects
- SVariables getVariables(string); //return SVariable object with name

SimulationModel Class

Variáveis

- double EvaluationResult; //Evaluation Function Result
- vector <SParameters> ParametersSelected; //parameters selected
- vector <TimeVariables> TimeVariablesResult; //simulation Result

Construtor

- SimulationModel();
- SimulationModel(vector <SParameters>, vector <TimeVariables>);
- SimulationModel(vector <SParameters>, vector <TimeVariables>, double);

Métodos

- void setEvaluationResult(double);
- void setParametersSelected(vector <SParameters>);
- void setTimeVariablesResult(TimeVariables);
- double getEvaluationResult();
- vector <SParameters> getParametersSelected();
- SParameters getParametersSelected(string);
- vector <TimeVariables> getTimeVariablesResult();

Generic Functions

As funções genéricas a utilizar pelo agente calibrador são definidas dentro de uma classe com o mesmo nome. Uma das razões pela utilização de métodos de classe em vez de funções, prende-se com a selecção dos algoritmos de calibração, que por sua vez, são classes que herdam todos os métodos desta.

typedef enum {PP, HC, SA, GA} TYPE;

Definição da lista de acrónimos disponíveis para os algoritmos de optimização. Apenas uma variável de estado para identificar a classe correspondente, à escolha do utilizador na interface gráfica. PP – Pre-processing, HC – HillClimbing, SA – Simulated Annealing, GA – Genetic Algorithm.

GenericFunc Class

Descrição: classe base dos algoritmos de optimização. Toda a aplicação tem acesso a estes métodos. Define na prática as variáveis onde todos os resultados de simulação

serão armazenados, como é o caso da variável SimulationResult, que armazena o último resultado simulado, o ClassModel que armazena os parâmetros e variáveis iniciais.

Variáveis

- EcoDynProtocol *ecdp;
- TYPE type;
- bool SIMULATION_RUN, SIMULATION_CHANGE_PARAMETER; //status flags
- string DataFileName;
- int simulStep, regStep, iteration, contadorParm;
- time_t simulStart, simulFinish, regStart, regFinish;
- int numberOfLines, numberOfColumns, numberOfLayers; //get from model dimensions model
- vector <ModelClass> ClassModel;
- vector <TimeVariables> SimulationResult; //save for current simulation variables data
- vector <string> SimulationVariable;

Construtor

- GenericFunc(EcoDynProtocol *pEcdp);

Métodos

- vector < vector <SParameters> > NeighboursResults; //save the solutions generated
- vector <SimulationModel> SimulationFinal; //save best solutions
- SimulationModel RealData; //load initial data from file
- SimulationModel BestData; //best simulation result
- Mensagens:
  - void MessageError(string msg);
  - void MessageWarning(string msg);
  - void MessageOk(string msg);
  - void MessageError1(string msg,bool prefix);
  - void MessageWarning1(string msg);
  - void MessageOk1(string msg);
- void FillTimeSpecDialog(int step,time_t start, time_t end);
- void FillOutputTimeDialog(int step,time_t start, time_t end);
- void UpdateParameterAction(string className, string parameter_name, double parameter_value);
- void UpdateVariablesAction(TListBox* ListBox1, BOXES* pBoxes);
- void AddSimulationStep(int box_aux, int regIndex_aux, int regTime_aux, SVariables aux);
- void RegisterVariables(char* varName, int regIndex,long regTime, Queue* pQueue);
- Comandos simulador:
  - void sendRunMessage();
  - void sendPausedMessage();
  - void sendStoppedMessage();
  - void EndSimulation();
- int getIndexModelClass(string className);
- int getIndexClassParameter(string parameterName, int ClassItem);

- void ClassesAvailable(Queue* pQueue);
- bool UpdateModelClass(int ID_message, Queue* message);
- Imprimir/salvar:
  - string alterPathFileData(string originalPath, int iteration);
  - void SaveInitialFileData(string FileName, vector<SParameters> SimulationParameter, vector <TimeVariables> SimulationResult);
  - void SaveFileData(string FileName, vector <TimeVariables> Result);
  - void ReadFileData(string namefile);
  - void PrintResultsIntoFile();
  - void PrintResultsIntoGraph(double value);
  - void printFinalSimulation();
  - void printInitInterSimulation();
  - void printInterSimulation(double error, double errorNew);
- vector <SParameters> ConvertParameters();
- void LoadBaseData(string);

Calibration Class

Variáveis:

- virtual void Run()=0;
- double best_cost;
- SimulationModel *best_solution;
- vector <SParameters> working_solution;
- bool direction;
- GenericFunc *parent;

Construtor:

- Calibration(EcoDynProtocol *pEcdp, GenericFunc *Parent);

Métodos:

- bool compareRandom(vector <int> randomIndex, int random_position); //test if some random_position already picked
- bool TestSolution(vector < vector <SParameters> > allSolutions,vector <SParameters> newSolution);
- void UpdateRunSolution(vector <SParameters>); //update into simulator new parameter solution and run
- virtual double Eval(bool typeResult); //calculate quality solution
- void NewSolution(unsigned int numRandom); //define a new parameter solution
- void AddSolution(); //copy solution to data structure - good solution
- SimulationModel BestSolution(); //best solution simulated
- double BestCost(); //update best cost simulated
- void WorkingSolution(); //update Current Solution Test

HillClimbing Class

Variáveis

- unsigned int number_cycles;
- double condition_stop;

Construtor

- HillClimbing(EcoDynProtocol *pEcdp, GenericFunc *parent);

Métodos

- void InitialData(double conditionStop, int maxIteration);
- void Run();
- bool Algorithm(double error, double errorNeighbour);

SimulatedAnnealing Class

Variáveis

- double condition_stop, k, temperature, delta, temp_step;
- int condition_number, iteration_number, iteration_success, success, iteration, iteration1;
- double population_error, new_population_error;
- string fileLog;
- int step;
- SimulationModel population; //best population until now
- SimulationModel new_population; //new population

Construtor

- SimulatedAnnealing(EcoDynProtocol *pEcdp, GenericFunc *Parent);

Métodos

- void Algorithm();
- void InitLog();
- void Log(SimulationModel solution, string desc);
- void Log(SimulationModel solution, double temperature, string desc);

GeneticAlgorithm Class

Variáveis

- unsigned int number_cycles, indice;
- double condition_stop;
- unsigned int num_childs;
- double error, errorNew;
- bool alterSolution, alterSolution1, type_mutation;
- int step;
- vector <SimulationModel> population; //population of n individuals
- vector <SimulationModel> population1; //population of n individuals
- SParameters v; //individuals

Construtor

- GeneticAlgorithm(EcoDynProtocol *pEcdp, GenericFunc *Parent);

Métodos

- void InitialData(double conditionStop, int maxIteration, int numChilds, int typeMutation);
- void Run();
- void Log(SimulationModel solution, string desc);
- void Algorithm();
- void FillPopulation();
- void CrossOver();
- SParameters generateNeighbourSwitch(SParameters);
- vector <SimulationModel> SelectBreed(vector <SimulationModel> p, vector <SimulationModel> p1);
- void UpdateSolution();
- void InitLog();
- int MutateSolution(vector <SimulationModel> p);