

J-Artificial Mind Project: Objectives and Methods

Short description of the project's topics and goals

Authors:

Gianluigi Ferraris and Marco Lamieri

Contents

1	Introduction	2
2	Requirements	2
3	Technology	3
4	Structure	4
5	Methodology	5
6	Work phases	6

1 Introduction

The main goal of the project is to implement two packages available for general usage in a wide variety of agent based simulation models. The packages will be, respectively, devoted to handle:

1. Genetic Algorithms;
2. Classifier Systems.

They will be used, into the simulation models, to easy obtain "minded" agents, i.e. agents that are fully autonomous, able to decide their own behaviours and to change it to fit their actions to the different environmental conditions. Another main usage will be searching bounded optimal solutions in very wide solution spaces closed to quite undefined problems.

2 Requirements

The main audience of the project is intended to be the researchers. This kind of users are generally interested in understanding what the software does, in which way the results are achieved and in a detailed analysis of how the algorithms are implemented in order to modify them to their specific needs.

To reach the goals of the project some requirements have to be respected during the implementation phase. In more detail:

Integrated and general tool for booth GA and CS the programs will handle both GA and CS in an high general way;

Clear source code special care will be given to write easily read and understood source code;

Re-usability source code must be flexible and modular to permit the re-usability of its single pieces in different applications;

General purpose the program have to be highly bounded to a large set of parameters in order to be as much general as possible, in a way that it can be used for many purposes and not only for specific applications. In this way modification of program's behaviour will be possible in a safer and easier way, also by non programmer people;

Customizable all the features must be easily customizable for the highest possible number of facts;

Non bounding assumption no bounding assumptions will be allowed during the code writing;

Complete parametrization all behaviours or specifications will be linked to a parameter value, allowing modification and inhibition of the parameter by the user;

Default values provided for all the parameters will be provided default values, to simplify the usage for people that has only a few knowledge of the GA and CS methods.

Parameter's domain control special controls will be done on the parameter's acceptability. The users will receive warning about incoherence of parameter values and other possible mistakes. The controls will always be limited only to warn the users in order to permit them testing strange cases also with apparently inconsistent parameters;

Suitable to develop cluster of GAs and CSs the programs are structurally able to nest several instances of themselves to realise specialised structures from clustered GAs or CSs and multiple GAs to mixed rings of GAs and CSs. In the mixed rings case the sharing of tasks between GAs and CSs may be realised via exchanging information or rules.

3 Technology

Usually the best strategy to develop a tool to use in agent base simulations is through an object oriented language. In this project we decide upon the Java language because of its qualities: with the Java language it is possible to create classes that represent element of the implemented algorithm in an abstract way. To represent a real problem the classes must be instantiated in objects. By this method is possible to create flexible programs that can be easily used in different kind of environment and solve many different problems.

The Java language is also full compliant whit many of the major simulation tools as *Swarm*¹ and *JAS*². Our software can be used also with the *jES*³ project, that is a large agent based simulation framework able to reproduce in a detailed way the context of a single enterprise or of a system of enterprises.

In more technical details:

- the code will be written in pure Java language;
- the selected programming approach will be strictly object oriented;

¹The *Swarm* project is an agent based simulation tool born at the Santa Fe Institute, <http://www.swarm.org>

²The *JAS* (Java Agent-based Simulator) project is a java agent based simulation tool realised by Michele Sonnessa at Turin University <http://jaslibrary.sourceforge.net/>

³The *jES* (Java Enterprise Simulator) project is developed by prof. Pietro Terna at the "Dipartimento di scienze economiche e finanziarie G. Prato"

- the objects will be as small and specialized as possible, in order to allow high re-usability and easy maintenance;
- boundary method will be set up to face the higher number of user's request: for all the variables of each object a "set" and a "get" method will be written;
- a precise naming convention will be adopted; the names of variables and modules will be descriptive, even if they will become very long.

4 Structure

The two packages will be realized in respect of a strictly hierarchical architecture made by stratification of three layers:

1. **Front-end** it contains the components able to deal with the user's programs. The user can call methods from the front-end's objects as black boxes to compute GA and CS, without a complete comprehension of the underlying algorithm. The methods implemented by these objects will be the most specialised. Typically they will consists on:
 - get/set methods for variables and parameters values;
 - set methods to describe the problem;
 - get methods to obtain suggestion;
 - set methods to reward the system.
2. **Core** it implements the objects involved in driving the AI engine. They will be able to interact with both front-end objects and services ones.
3. **Service** this layer contains objects that supply common computation services as, for example, random number generator and debug facilities. This layer is highly re-usable and strongly generalised.

The use of a hierarchical structure means that:

- each object in a certain layer can call methods only from objects of the same level or from those of levels below;
- each object in a certain layer can answer only to call from objects of the same level or from those of levels above.

5 Methodology

A simple methodology, based on some fundamental statements, will be followed to face the objectives of the project and to facilitate the work:

1. Basic version
 - (a) At first there will be implemented a basic version of each package. This version will realise the "classical" approach to the AI method.
 - (b) The basic version will be fully tested to ensure that it works as desired;
2. Full version
 - (a) Starting from the basic version there will be added some features in order to realise a full version.
 - (b) The full version will be tested.
3. Advanced version
 - (a) At this point we move from the classical AI approach to an advanced one in order to handle special features as fitness apportioning routines, multiple genomes based individuals, enlarged alphabet definition, special chromosome types and so on.
 - (b) The advanced version will be tested.

Each version will be tested under two different points of view:

Technical test the technical test will prove the goodness of each version based on four levels:

1. unit test: involving simple objects or single methods;
2. module test: regarding the whole functionality of single objects;
3. system test: devoted to prove the interaction among objects;
4. stress test: to benchmark the performance of each version compared to the previous one and to a baseline considered acceptable.

Functional test the functional test consist in running a complete model that involve all the main functionality of the software. This model will also represent sample application.

For each of the described steps we would like to produce a paper that describes the achieved result. We would also like to support the software with a complete documentation realised by use of specific tools as *javadoc*.

After the software development we would like to prepare a complete application that tries to solve an easy economic problem on which it is possible to run some experiments. Analysing the results of those experiments it is possible to compare the project's methodology (GA and CS) with more classical approaches (i.e. operative research technique).

At least all the papers will be merged creating a complete documentation of the project. This "summa" will be completed by a critic evaluation of the achieved results and by a project report.

6 Work phases

To minimize the time to delivery and to keep a complete control on the status of the project we decide to split the job in the following logical phases:

1. functional Analysis;
2. user's interface definition;
3. empowerment of the classical AI methods specification;
4. description of the test models;
5. functional test planning;
6. description of the economic model;
7. documentation of the functional analysis and validation;
8. technical analysis;
9. documentation of the technical analysis and validation;
10. implementation;
11. technical test;
12. documentation, test and acceptance;
13. user's manual preparation;
14. test models realisation and runs;
15. collection of results of the runs and evaluation;
16. result's acceptance;
17. documentation of the test models and user's manual improvement;
18. final economical model realisation;

19. runs of the model and results evaluation;

20. final report and validation of the whole project's output set.

Tasks from 8 to 13 will be reiterated four times to develop GA basic version, GA advanced version, CS basic version and CS advanced one. The failure of task 12 will impose the regression to task 10 each time.