Intelligent Farmer Agent for Multi-Agent Ecological Simulations Optimization

Filipe Cruz[1], António Pereira[1], Pedro Valente[1], Pedro Duarte[2], and Luís Paulo Reis[1]

[1]LIACC – Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal, +351 22 508 14 00
[2]CEMAS – Universidade Fernando Pessoa
Praça 9 de Abril, 349, 4249-004 Porto, Portugal, +351 22 507 13 00
{filipe.cruz, amcp, pedro.valente, lpreis}@fe.up.pt, pduarte@ufp.pt

Abstract. This paper presents the development of a bivalve farmer agent interacting with a realistic ecological simulation system. The purpose of the farmer agent is to determine the best combinations of bivalve seeding areas in a large region, maximizing the production without exceeding the total allowed seeding area. A system based on simulated annealing, tabu search, genetic algorithms and reinforcement learning, was developed to minimize the number of iterations required to unravel a semi-optimum solution by using customizable tactics. The farmer agent is part of a multi-agent system where several agents, representing human interaction with the coastal ecosystems, communicate with a realistic simulator developed especially for aquatic ecological simulations. The experiments performed revealed promising results in the field of optimization techniques and multi-agent systems applied to ecological simulations. The results obtained open many other possible uses of the simulation architecture with applications in industrial and ecological management problems, towards sustainable development.

Keywords: Ecological simulations, agents, optimization, simulated annealing, tabu search, genetic algorithms, reinforcement learning.

1 Introduction

Coastal ecosystems are used for multiple purposes (fishing, tourism, aquaculture, harbor activities, sports, etc.) and are the final destination of many pollutants generated by agriculture and other human activities. Considering that over 60% of the world's population lives within 60 km from the sea, the correct management of these ecosystems is very important towards world sustainable development [1] [2].

The diversity of usages and the opposite interests of stakeholders and some institutional authorities, coupled with the slowness of the decision process due to the huge number of possible decisions generated by the different management policies, make the implementation of efficient automatic management algorithms very difficult to achieve [3].

In this context, the use of intelligent agents [4] [5] [6] seems to be very promising. Each institutional authority and stakeholder may be modeled as an agent, interacting with simulation tools - able to predict the outcome of different decisions - getting results and configuring new conditions for further simulation experiments.

The human factors can be represented by agents with specific goals and visions for the coastal area that can sometimes contradictory or of unsure effect upon the ecosystem. Examples of such agents are tourism managers, bivalve farmers, and representatives of civil authorities. Each has different goals that they want to see maximized or minimized. Including them in a multi-agent system surrounding the ecological simulator allows them to be concurrently represented and enabling the prediction, to some extent, of the outcome through time of their interactions with the ecosystem [7].

Realistic simulations of ecosystems require describing several physical, chemical and biological processes in mathematical terms. Physical processes include flow and circulation patterns, mixing and dispersion of mass and heat, settling and resuspension of planktonic organisms and suspended matter, insulation and light penetration. Chemical processes include biogeochemical cycles of important elements, such as nitrogen and phosphorus. Biological processes include growth and death rate of any organisms that may alter the concentration of different elements. The accurate simulation of these processes is very important for setting up a reliable and realistic model of the whole ecosystem. Detailed representations of these processes can be found in the literature since the 1980's [8].

This paper refers, in particular, to the development of a bivalve farmer agent which interacts with a realistic ecological simulator (EcoDynamo [2]) to find out the best combination of bivalve seeding areas within an allowed area of exploitation and a maximum seeding occupied area.

The developed bivalve farmer agent has implementations and adaptations of known multi-objective optimization algorithms, well documented in the literature such as Simulated Annealing [9], Tabu Search [10], Genetic Algorithms [11] and Reinforcement Learning [12]. It attempts to take advantage of each algorithm positive aspects, minimizing their limitations. The bivalve farmer agent has the objective of finding an optimum combination of locations providing better shellfish growth and production within a bay.

The paper is organized as follows: section 2 describes the problem in analysis; section 3 presents the architecture and implementation; section 4 contains a description and analysis of obtained results; section 5 mentions the conclusions and indicates future work.

2 Problem Statement

As a proof of concept, a bivalve farmer agent was developed to discover by itself the best combinations of locations to seed and harvest bivalve species. The bivalve farmer agent interacts with the simulator application in order to run series of test simulations seeking to find the optimum, or very near optimum, combination of lagoon bivalve farming regions (cells) where bivalve production would be maximized.

The tests were carried out using one validated model for Sungo Bay, People's Republic of China [1]. Sungo Bay is modeled as a 2D vertically integrated, coupled hy-

drodynamic-biogeochemical model, based on a finite difference bathymetric staggered grid with 1120 cells (32 columns x 35 lines) and a spatial resolution of 500m [1]. The idea is to find the best solution - five regions within a rectangular area of 88 cells (square regions) that maximize bivalve production. Hereafter, "solution" will be used to refer to any combination of five regions within the mentioned area. It is important to refer that due to the realistic characteristics of the ecological simulation, the existence of bivalves in one location will affect the growth of bivalves in the neighborhood. Simulated bivalves feed on suspended particles such as detritus and phytoplankton cells, with the potential to cause significant local depletion of these food items. Therefore, placing many regions together could negatively affect the potential yield of those regions. The extent of the influence depends on many factors such as tidal flux, quantity and quality of suspended food particles, and water quality, substantially increasing the complexity of the problem. Figure 1 shows two different possible combinations of regions (cells).

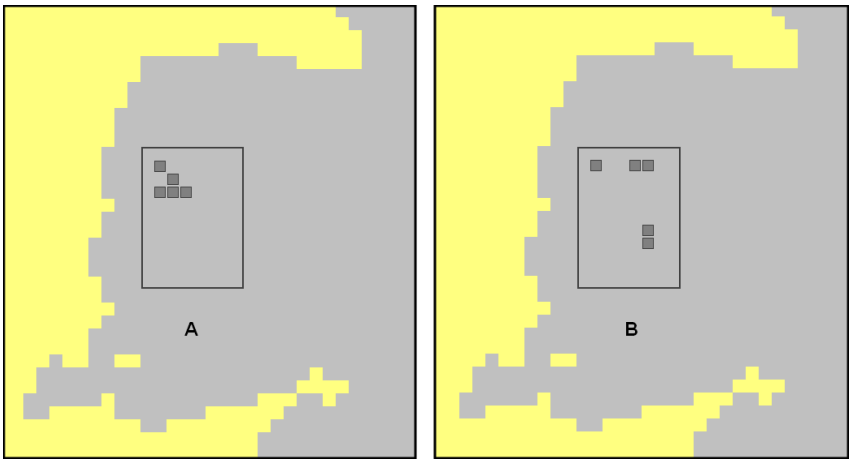

Fig. 1. Two different configurations (A and B) of farming regions: five farming cells are selected from the wide available area in each simulation.

Taking into account the heavy time and processor power required to perform full length realistic simulations (complete bivalve culture cycle is approximately 1.5 years – about 1 576 800 simulation time steps of 30 seconds) it was a requirement of the bivalve farmer agent to intelligently choose its test combinations of cells with simulations of only 1000 time steps.

3 Implementation

3.1 Architecture

The implementation is based on a multi-agent architecture following the principles of Weiss [5] and Wooldridge [6], where agents communicate with the simulator applica-

tion via TCP/IP packets. The simulation tool (EcoDynamo [2]) was developed for the simulation of aquatic ecosystems, and is able to communicate with several agents representing the human interests over the simulated ecosystem, namely the stakeholders, the national authorities and the municipality. This communication is supported by ECOLANG messages [2] - ECOLANG is a high-level language that describes ecological systems in terms of regional characteristics and translates living agent's actions and perceptions. The format of the messages enables easy readability, simplicity and expandability, and is independent from any computational platform or operating system. Figure 2 shows the architecture used in the experiment with the simulator, the farmer agent and one visualizer application.

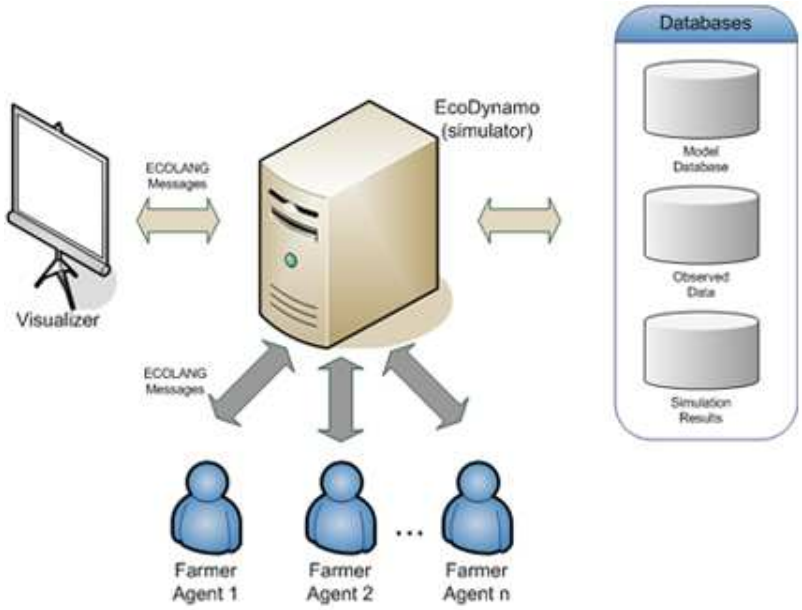

Fig. 2 Experimental system architecture.

The architecture of the multi-agent system was structured to allow several agents to interact with the simulator at the same time with different purposes in mind. There is much space for diverse applications of machine learning in ecological modeling which can be better exploited with the proper architecture [7] [8] [13].

3.2 Implemented Algorithms

To implement the sense of intelligence in the choice of region combinations of the bivalve farmer agent, a system of customizable tactics was developed. The system allows different multi-objective optimum solution finder techniques to be applied at the same time. The base of the program is a simple hill-climbing optimization algorithm based on Simulated Annealing with Monte Carlo probability [9] [14], iteratively seeking a new possible solution and accepting it as the best found so far if its quality

is considered higher than the previous best. The program, however, allows for several other configurable optimizations to be activated influencing the selection logic. These optimizations are based and adapted from documented implementations of Tabu Search [10] [14] [15], Genetic Algorithms [16] [17] and Reinforcement Learning [12]. A simple example: one configuration may allow the iterations to start functioning with a random search algorithm, trigger a genetic algorithm based crossbreeding at 45% of the search process, and switch to a low dispersion local neighbor solution selection near the end.

It is important to also notice that depending on the algorithms that are used, the selection of the initial combination to test can either greatly assist or greatly hinder the overall performance of the algorithm. Thus, a decision was made to create an architecture that would always start from a random seed but could then easily be configurable to alternate, in real time, between the different implemented algorithms. The selection of what and when to change is described through the so called tactics files, which are nothing more than simple text files with listed algorithms and their correspondent parameters. The fine tune of the parameters in these files to achieve better results can be compared to a meta-heuristic problem.

The algorithms implemented in the farmer agent were called FarmerSA, FarmerTabu, FarmerGA and FarmerRL. A small description of each one follows.

FarmerSA. The implemented Simulated Annealing (SA) algorithm follows the usual guidelines of Kirkpatrick [9], allowing the user to define the typical parameters of initial temperature, final temperature and descent rate.

As documented widely in literature, SA is based in the natural process of slower cooling giving higher resistance to certain materials. A threshold formula slowly increases the probability of accepting a given test solution of superior quality as the temperature lowers by each passing iteration. The overall concept behind the algorithm is allowing the system to have enough time to explore most part of its universe of solutions whilst the entropy is high enough to prevent the algorithm to follow only one path of the best solutions. By slowly restricting its entropy, the algorithm will eventually constrain itself to the best local solutions and hopefully this slow process will prevent it from getting stuck in local maximums instead of finding the global maximum. In critic terms, for some cases it can work very well, but for others it requires a great number of iterations to assure a high probability of finding a good solution and without certainty of being the optimum. It depends on the complexity/linearity of the solution search area itself, on how the neighbor solutions are defined, on the speed of the temperature drop and on a hefty amount of luck with the Monte Carlo probabilities. In synthesis - too many factors that make the exclusive use of this algorithm not recommendable for complex problems.

A special parameter exists in this implementation that allows the algorithm to ignore the Monte Carlo probability. This parameter is used to guarantee using a new solution when its quality is higher than the one considered as the best.

FarmerTabu. The adaptation of Tabu Search [10] was called FarmerTabu and its implementation is based on maintaining a hash list of all the previously tested solutions, so that when it is toggled on, it simply prevents the simulation from choosing a previously tested solution as the next solution to test, also keeping a

counter of how many times the algorithm has revoked the choice of a new and previously untested option from its choice of possible next (referred to as neighbor) solution [14]. The user can define a threshold value for the counter of refused solutions to activate a special mutation factor that serves to stir the process into other solution search paths once the current path has apparently already been overly explored without much success.

FarmerGA. The FarmerGA implementation maintains a list of best solutions found so far and crossbreeds them to form new combinations of regions. The user can define the number of solutions to take into account as parents and the number of new breeds. Unlike common Genetic Algorithms [16], this FarmerGA doesn't account for any mutation factor, making imperative that the list of solutions used as parents contains combinations of regions as spread out through the solution universe as possible, in order to achieve results that guarantee a solution not restricted to a local maximum.

The way the FarmerGA crossbreeds the parents list is based on the quality hierarchy and in one parameter configurable by the user. In simple terms, it attempts to breed the best solution with all the others in the list of top candidates, the second best with half of all the other members of the list, the third best with a third of all the others and so on. The child genes are selected as the best individual locations amongst both parents. For each pair of two good solutions (A and B) as seen on figure 3, the best locations are genes of each parent chosen according to their individual performance (measured in tons of bivalve harvested in previous simulation). The child derived (C) is either registered in a list of children to be tested or disregarded if it is present on the FarmerTabu list.

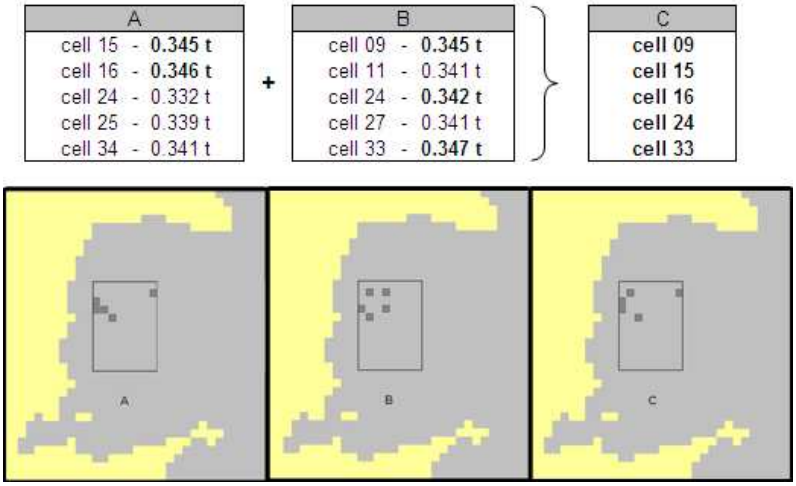

Fig. 3 FarmerGA breeding result: A and B - parent configurations; C - child configuration

FarmerRL. The FarmerRL optimization is somewhat far from what is usually referred to as Reinforcement Learning [12]. This implementation maintains a neighbors list for each possible region, containing information on its neighbor locations quality for farming. As the test iterations occur, calculated weights are summed to the quality of the areas selected in the tested solution, increasing or

decreasing its value. The value of the weight depends on the geometric distance between the farming result of an individual zone and the average amongst all the zones of the tested solution.

The farming quality value of each region is taken into account when the next selection of other neighboring solutions for testing is performed. As more iterations occur, the quality values of good bivalve farming regions gets higher and the quality values of bad bivalve farming regions gets lower, thus restraining the scope of search to the better regions. The strength of the weight is configurable by the user and can vary in real-time. The definition of neighbor solutions also influences greatly on how this optimization will perform. There are 8 implemented types of algorithms for choosing the next neighbor solution of FarmerRL:

1. Choosing the best neighbors of each position, using Monte Carlo probability factor.
2. Choosing the best neighbors of each position, only using Monte Carlo probability factor for the positions below average quality.
3. Choosing the best neighbors of each position, without using Monte Carlo probability factor.
4. Choosing the best neighbors of each position, without using Monte Carlo probability factor, but changing only one of the positions.
5. Randomly shift all positions to their neighbors.
6. Randomly shift positions to their neighbors of only one of the positions.
7. Random search new position in positions below average quality.
8. All new positions completely randomly selected.

3.3 Tactics System

A tactics system was developed to allow the users to configure the real-time parameter reconfiguration of the implemented algorithms.

The base algorithm FarmerSA is always being applied and can be configured/parameterized through a text file (*config.cfg*). The parameters on this file will define the number of iterations that the process will take to finish (extracted from initial value, final values and descent rate of the temperature), the number of simulation steps and the tactics file to use during the process.

The remaining algorithms can be configured through the specific tactics file used. This tactics file contains text lines which are referred to as toggles lines. These lines contain the parameterization of the algorithms described previously. If these algorithms are not toggled in this file to start at a given time frame of the process, they will not be used during the process.

FarmerTabu and FarmerRL can be toggled on, re-parameterized or toggled off at any point in time of the process. FarmerGA functions in a different manner: as each toggle line for FarmerGA will trigger the process of running the genetic crossing itself, it is expected to only be toggled a few steps into the process so that it has time to build a large and wide spread enough list of good results for its cross breeding.

Toggles for the same algorithm can be defined more then once per tactics file, typically representing reconfigurations of the parameters at different time frames of the process. As seen in table 1 each toggle line (*tog*) contains several parameters, the first

one refers to the algorithm it is configuring (*0 for FarmerTabu, 1 for FarmerGA, 2 for FarmerRL*), the second one to the time frame in percentage during which it shall be activated (e.g: *0.0 will be triggered right from the first iteration, 0.5 will be triggered after half the iterations have occurred*). The other parameters depend on the specific algorithm.

Table 1. Example of a tactics file: 'toggles_rand.tgl'

Keyword	Toggle Number	Param1	Param2	Param3	Param4
init					
tog	**0 (TA)**	**0.0**	**1**	**0.8**	**2**
tog	**1 (GA)**	**0.5**	**1**	**15**	**8**
tog	**1 (GA)**	**0.8**	**1**	**15**	**8**
tog	**2 (RL)**	**0.0**	**1**	**0.05**	
tog	**2 (RL)**	**0.5**	**8**	**0.05**	
tog	**2 (RL)**	**0.7**	**7**	**0.05**	
eof					

Table 1 exemplifies a tactics file with a total of 6 toggles defined:

- The first refers to FarmerTabu (*tog 0*), defining it to actuate from the beginning of the process (*0.0*), turned on (*1*) using a *0.8* threshold for repeated selections in tabu list before applying mutation of type *2* (distribute one random number of solution members for neighboring regions with good qualitative indicators).
- The second type of toggles refers to the FarmerGA (*tog 1*), it is used twice during this process, one at half of the iterations and again after 80% of the iterations have been processed. It is turned on (*1*) and will create a list of maximum 15 children members out of the top 8 members of the best found results so far.
- The third type of toggles refers to the FarmerRL (*tog 2*), which starts with neighbor selection algorithm type *1* (best neighbor with Monte Carlo probability factor), using quality weight update of *0.05*, then changes to neighbor selection type *8* (completely random) at half of the process with weight update of *0.05*, and finally changes to neighbor selection type *7* (half random) at 70% of the process and using weight update of *0.05*.

Different tactics may easily be defined and different syntaxes for the tactics configuration file are also very easy to define, including new parameters for the available algorithms.

4 Results and Discussion

Several different tactics were tested on the Sungo Bay model to assist on determining the optimum combination of 5 bivalve farming regions out of the allowed bivalve farming area. Table 2 shows the results of the best solution encountered by each tactic.

Table 2. Results of tested tactics

Tactic	Result (Ton)
hillclimber 01	**8019.142**
hillclimber 02	**8027.930**
neigh 01	**8051.824**
neigh 02	**8053.852**
3GA 01	**8049.961**
3GA 02	**8050.868**
neighrand 01	**8056.120**
neighrand 02	**8055.063**
neighrand 03	**8053.148**
totrand 01	**8051.824**
totrand 02	**8051.824**

The different tactics were intuitively determined whilst undergoing the development of the application. Some were uniquely written to show the performance following traditional simplistic methods for terms of comparison - *hillclimber* performs a simple choice of the best close neighbor (only one position from the possible five is changed from each solution at a time) and its refined version named *neigh* which allowed for up to 3 positions to shift from each solution at a time.

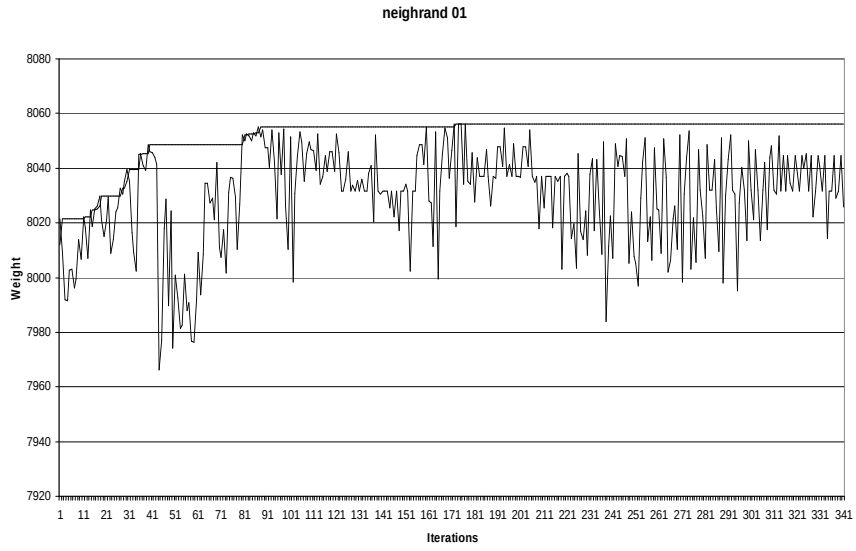

Fig. 4 Results of the *neighrand 01* development

Other tactics were aimed at abusing a certain capability of the application beyond reason to test its impact on the outcome result. One such is the *3GA* tactic which repeats the genetic algorithm variance three times without *a priori* having a proper

scope of the whole possible universe. In similar fashion, *totrand* tactic had a more random search orientation (admits bursts of best neighbor searches) in an attempt to measure the real value of scoping the entire universe prior to applying the remaining available methods. At last, *neighrand* is the more balanced of all the tactics devised, despite not being fully fine tuned to always unravel the optimum or very near optimum solution.

Figure 4 shows the development throughout the iterations of tactic *neighrand 01*. This tactic was scripted by the file showed in table 1 (*toggles_rand.tgl*) - had FarmerTabu on during all the process, had 2 instances of FarmerGA occurring once at half way through the simulation and again at 80%, and its FarmerRL changed three times through the process to alter its neighbor selection parameters. The weight refers the amount of bivalves harvested in each iteration of the simulation. Each iteration represents the same simulation period of a new combination/solution of bivalve zones. So the first line on the graph shows the weight result for the current iteration, whilst the second line shows the weight result for the best solution found so far. Testing the different tactics under the same conditions in short time period simulation provided several different combinations of locations which could be tested later in longer time period simulations for the validation of results.

The best results from the tests carried out are presented in tables 3 and 4 and can be seen in figure 5. Analyzing the test results leads to conclude that the best farming regions of the area are located on the lower bottom of the global search area of 88 zones. There is a strong possibility that other, more optimum combinations could be found by re-analyzing the problem concentrating on that area alone.

Table 3. Best solution

Location	Weight (Ton)
81	1613.490035
82	1612.912944
83	1612.423538
72	1611.792726
80	1613.570428
Total weight	8056.11995

Table 4. Second best solution

Location	Weight (Ton)
81	1613.646772
82	1613.165278
83	1612.508254
63	1610.178968
80	1613.654490
Total weight	8055.084283

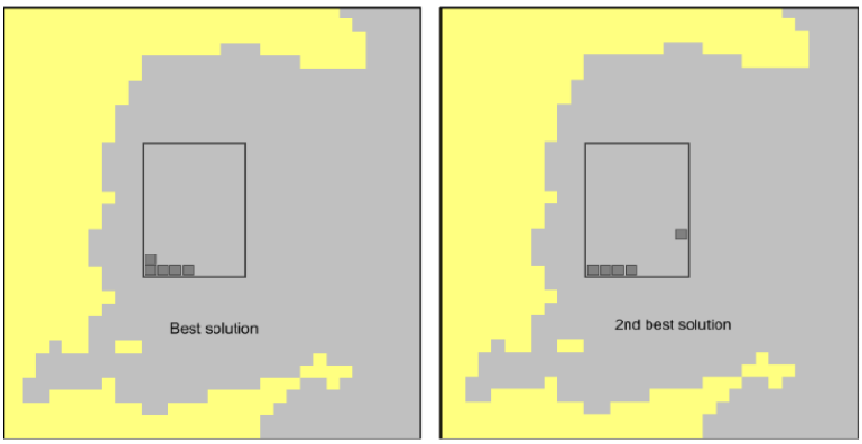

Fig. 5 Visualization of the two best solutions obtained by *neighrand 01* development

A reminder must be added that the program is based on a random initial search solution. The fact that all of the tactics converged to similar solutions (containing a majority of elements in early 80s locations) reassures the quality of the location as a most profitable zone to explore bivalve farming.

5 Conclusions and Future Work

Generally, the management of coastal ecosystems may be done in many different ways and there is hardly one optimal solution, but most likely a "family" of "good" management options. Giving the large complexity of these systems and the numerous synergies between environmental conditions and management options, finding "good" choices cannot be reduced to simple optimization algorithms, assuming linear or some defined form of non-linear relationship between a set of parameters, variables and goal seeking functions. Mathematical models may be very useful in finding "good" management solutions. However, finding these may require many trial and error simulations and this is why using agents that may look automatically for the mentioned solutions may be advantageous. This requires the *a priori* definition of "good" solutions and constraints. For example, one may wish to increase aquaculture production but keeping water quality within certain limits for other uses.

This paper presented a different approach for the problem of bivalve aquaculture optimization in coastal ecosystems. The approach is based on the development of a bivalve farmer agent interacting with a realistic ecological simulation system. The farmer agent optimizes bivalve production by using known optimization techniques for solving the problem of selecting the best farming regions combinations, maximizing the production of the zone. The approach enabled also to achieve better results than using hand-tuned techniques.

From the comparison between distinct optimization methodologies, it was also concluded that better results are achieved by using a combination of different optimization methodologies by the use of our tactics configuration file. However more experiences with different scenarios and longer growing times must be carried out in order to fully validate this conclusion.

There is still plenty of research to be accomplished within the combined fields of ecological simulation and multi-agent systems. There are many scenarios where intelligent programs acting as agents could emulate and enhance the realistic behavior of many different factors within simulation systems. Decision support systems based on realistic ecological simulator have much to gain with the inclusion of multi-agent systems interaction in their architecture.

The methodology experienced in this work will be extended to test more combinations with benthic species and regions: one region/several benthic species, several regions/one benthic species, several regions/several benthic species, restricted farming areas/unrestricted farming areas, etc.

Also the optimization methodologies must be improved in order to allow the simulation period to grow in order to verify the behavior of the best tactics in one complete farming cycle (about 1.5 years).

Acknowledgements. This work is supported by the ABSES project – "Agent-Based Simulation of ecological Systems", (FCT/POSC/EIA/57671/2004). António Pereira is supported by the FCT research scholarship SFRH/BD/16337/2004. Filipe Cruz was supported by a POCI2010 program grant.

References

1. Duarte, P., Meneses, R., Hawkins, A.J.S., Zhu, M., Fang, J., Grant., J.: Mathematical modelling to assess the carrying capacity for multi-species culture within coastal waters. Ecological Modelling 168 (2003) 109-143
2. Pereira, A., Duarte, P., Reis, L.P.: ECOLANG – A Communication Language for Simulations of Complex Ecological Systems. In: Merkuryev, Y., Zobel, R., Kerckhoffs, E. (eds): Proceedings of the 19th European Conference on Modelling and Simulation, Riga (2005) 493-500
3. Pereira, A., Duarte, P., Reis, L.P.: An Integrated Ecological Modelling and Decision Support Methodology. In: Zelinka, I., Oplatková, Z., Orsoni, A. (eds.): 21st European Conference on Modelling and Simulation, pp. 497-502, ECMS, Prague (2007)
4. Russel, S., Norvig, P.: Artificial Intelligence: A modern approach. Prentice-Hall, 2nd ed. (2003)
5. Weiss, G.: Multiagent Systems. MIT Press (2000)
6. Wooldridge, M.: An Introduction to Multi-Agent Systems. John Wiley & Sons, Ltd (2002)
7. Pereira, A., Duarte, P., Reis, L.P.: Agent-Based Simulation of Ecological Models. In: Coelho, H., Espinasse, B. (eds.): Proceedings of the 5th Workshop on Agent-Based Simulation, Lisbon (2004) 135-140
8. Jørgensen, S.E., Bendoricchio, G.: Fundamentals of Ecological Modelling. Elsevier (2001)
9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimizing by Simulated Annealing. Science, Number 4598, vol. 220 (1983)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers (1997)
11. Michalewicz,Z.:Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag (1999)
12. Sutton, R.S., Barto, A.G.:Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
13. Dzeroski, S.: Applications of symbolic machine learning to ecological modelling. Ecological Modelling 146 (2001)
14. Mishra, N., Prakash, M.K., Tiwari, R., Shankar, F., Chan, T.S.: Hybrid tabu-simulated annealing based approach to solve multi-constraint product mix decision problem. Expert Systems with Applications 29 (2005)
15. Youssef, H., Sait, S.M., Adiche, H.: Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Engineering Applications of Artificial Intelligence 14 (2001)
16. Amirjanov, A.: The development of a changing range genetic algorithm. Computer Methods in Applied Mechanics and Engineering 195 (2006)
17. Sait, S.M., El-Maleh, A.H., Al-Abaji, R.H.: Evolutionary algorithms for VLSI multi-objective netlist partitioning. Engineering Applications of Artificial Intelligence 19 (2006)