Ant Colony Systems and the Calibration of Multi-Agent Simulations: a New Approach

Benoît Calvez, Guillaume Hutzler

Ant Colony Systems and the Calibration of Multi-Agent Simulations: a New Approach

Benoît CALVEZ and Guillaume HUTZLER

IBISC, FRE 2873
Université d'Évry-Val d'Essonne/CNRS
{benoit.calvez,guillaume.hutzler}@ibisc.univ-evry.fr

Abstract. In this paper, we propose a new approach for the exploration of the parameter space of agent-based models: Adaptative Dichotomic Optimization. Agent-based models are generally characterized by a great number of parameters, a lot of which cannot be evaluated with the current knowledge about the real system. The aim of the work is to provide tools for the calibration of these models, which consists in finding the optimal set of parameters for a given criterion. The criterion can be for example that the model achieves a specific function optimally or that the results of the simulation are as close as possible to experimental data. Our approach is based on the partition of the parameter space (the interval of variation of each variable is divided into a finite number of intervals) and on a parallel exploration of the various parameters by the agents of the model. The navigation in the parameter space is done by grouping or dividing adaptively some of the intervals, according to an algorithm which is adapted from Ant Colony Systems.

1 Introduction

Agent-based simulation (ABS) is interested in the modelling and the simulation of complex systems. Its aim is to reproduce the dynamics of real systems by modelling the entities as agents, whose behaviour and interactions are defined, and which "live" in a simulated environment. A first validation of such models is obtained by comparing the resulting dynamics, when the model is simulated, with that of the real system (measured thanks to experimental data).

One of the crucial aspects of the design process lies in the tuning of the model. Indeed, this kind of model is generally characterized by lots of parameters which together determine the global dynamics of the system. The search space is thus gigantic. Moreover, the behaviour of these complex systems is often chaotic: on the one hand small changes made to a single parameter sometimes lead to a radical modification of the dynamics of the whole system; on the other hand some emergent phenomena are only produced in very specific conditions and won't occur if these conditions are not met. The solution space can thus be very small. As a consequence, the development and the parameter setting of an agent-based model may become long and tedious if we have no accurate, automatic and systematic strategy to explore the parameter space.

The development of such a strategy is precisely the object of this paper, which follows several works on the subject (see next section). All of these, however, appeared to be too limited, either because they achieved only a limited exploration of the parameter space, or because they required a great number of simulation runs in order to converge. The main difference between classical optimization and the calibration of agent-based models is but that the evaluation of a single set of parameters generally requires at least a complete simulation run, which may take quite a long time. If the model is stochastic, it may even require several runs for the same parameter set, in order to have a good confidence in the evaluation. It is thus most important that the number of evaluations be reduced to its minimum if the method is to be usable.

The key idea of our approach is to take advantage of the natural concurrency in agent-based models, so as to achieve a parallel exploration of the parameter space. In this approach, which we call Adaptive Dichotomic Optimization (*ADO*), every individual agent is instantiated with its own specific parameter values, chosen in a finite number of intervals, for each variable. The different agents of the model thus have completely different settings. The basic intuition is that the "performance" of the resulting model will be statistically better if it includes a larger proportion of agents with "good" parameters. The intervals in which the parameters have been chosen for the agents of a given model will thus be rewarded depending on the performance of the model. The exploration of the parameter space will then be refined adaptively by merging together adjacent intervals with low rewards and by dividing in two intervals with high rewards.

The next section describes previous works on the subject and discusses their respective strengths and weaknesses. We then present our approach in more details in section 3. We present our results in section 4. We discuss our results in section 5 before concluding in section 6.

2 Previous works

Different methods have already been proposed to explore automatically the parameter space of discrete models. In the `NetLogo` platform for instance, the "BehaviorSpace" [1] tool allows to explore automatically and systematically the parameter space. This space is a Cartesian product of values that each parameter can take, some of them being chosen as fixed, others being limited to a subset of all possible values. However when the model has lots of parameters, some of which can take a good many values (real-valued parameters for example), the parameter space becomes huge and the systematic exploration becomes impossible. The number of values tested for each parameter will thus be very low and the corresponding exploration limited to a very small part of the whole parameter space.

Other methods have been proposed, which differentially explore the whole parameter space, focusing on the most interesting areas. That's the case of the method developed by Brueckner and Parunak [2]. They use a "parameter sweep infrastructure", which is similar to the "BehaviorSpace" tool of `NetLogo` . How-

ever, to avoid a systematic exploration, they use searcher agents and introduce the notion of fitness. The aim of a searcher agent is to travel in the parameter space to look for the highest fitness. Starting from a given location in the parameter space, searcher agents have two choices: move or simulate. Each agent chooses according to the confidence of the fitness estimate (proportional to the number of simulations at this point) and the value of the fitness. If it chooses to move, it heads for the neighboring region with highest fitness. A disadvantage of this method is that searcher agents may head for local fitness maxima.

Another method is to add knowledge to the agent-based model, as is the case with white box calibration [3, 4]. The principle is to use the knowledge of the agent-based model to improve the tuning process. The aim is to reduce the parameter space by breaking down the model into smaller submodels, which can be done using different methods (General Model Decomposition, Functional Decomposition, ...). Each of the submodels is then calibrated, before merging them back to form the model. The division and fusion operations are the difficult steps of the method. The division operation, on the one hand, requires the addition of knowledge about the model, which may not be available. The fusion operation, on the other hand, has to merge calibrated submodels into a calibrated higher model, which is not automatic.

Sallans et al. [5] present a work for a model with lots of parameters. Some of these parameters are chosen based on initial trial simulations. The other parameters are chosen using the Metropolis algorithm, which is an adaptation of the Markov chain Monte Carlo sampling to do a directed random walk through parameter space. This method performs well on a continuous parameter space, but will hardly be usable when the parameter space is chaotic (a small change in the value of a parameter may lead to a dramatically different dynamical behaviour) or if the model is stochastic.

Another approach is to consider the problem of the development and the validation of agent-based models as an optimization problem. The validation can thus be reformulated as the identification of a parameter set that optimizes some function. The optimization function would be for example the distance between the artificial model that we simulate and the real system. To solve this optimization problem, several authors [6–9] propose to use genetic algorithms with different variants. The use of these methods needs the computation of a fitness function in order to assess the quality of each parameter setting. The trouble, as we underlined in the introduction, is that the computation of the fitness function for a single model potentially requires several simulation runs, each of which may last for several hours. Without a powerful computational cluster, the method is therefore difficult to apply.

To reduce the need for simulation runs, De Wolf et al. [10] propose to couple ABS to macroscopic analysis techniques such as the "equation-free" paradigm developped by Kevrekidis et al. [11]. In this approach, macroscopic measures are made during the simulations that are given to the analysis algorithm. The results of the latter are then used to re-initialize the simulations with different settings, and the all process is iterated until a satisfactory dynamics is obtained in the

ABS. However, the approach is based on the observation of global trends that are supposed to evolve gradually, which may not always be the case with complex systems. More importantly, the approach requires a pretty good understanding of the relationships between the macroscopic behaviour of the system and the microscopic behaviour of its constituents, which is precisely what is seek in a lot of simulations. As the authors state it, "Some of the steps in the road-map are far from trivial, because some knowledge on how to bridge the micro-macro gap is required."

Finally, Klein et al. [12] propose to explore the parameter space of an agent-based model by building a model of the agent model itself, which could establish "a relationship between some parameters governing local behaviour and some indicators of the global phenomenon." The model has to be chosen appropriately, then trained by running simulations so as to produce accurate results. To this end, the parameter space is divided into sub-spaces, called 'meshes', inside which the behaviour of the system is supposed to be homogeneous. The main drawback of this approach is that it introduces yet another model in the process, which is itself only an approximation of the agent-based model, and that it requires a preliminary phase of restriction of the domain of study, which questions about the applicability of the approach for really complex systems.

3 *ADO* Global vision

The basic idea is to take advantage of the fact that agent-based simulations rely on multiple agents. Instead of considering that all of them should be parameterized in the same way, we propose to enable the parameterization of the different agents with different settings. Instead of evaluating parameter settings one at a time, we can thus evaluate several settings in parallel in a single model. The hypothesis that we made is that if such a model behaves "correctly", then the agents that compose the model are considered to have adequate parameters. In a way, the idea is similar to the principle of Ant Colony Systems [13].

The other idea is explore the parameter space differentially, depending on the potential interest of the different regions of the space. Taking inspiration from dichotomic search and from octrees, we consider that a parameter space of dimension n (n independent parameters) is initially divided into hypercubes of dimension n. Practically, the definition set of each of the parameters is initially divided into a fixed number of identical intervals. Then, for each individual parameter, depending on rewards received by the different intervals, we may differentially merge or divide the intervals. If an interval did receive high rewards, this signals an interesting area of the parameter space. The corresponding interval may thus be divided into two sub-intervals so as to have a more precise evaluation of the interesting parameters. On the contrary, if two adjacent intervals received few rewards, this signals an area of the parameter space of low interest. The corresponding intervals may thus be merged so as to stop wasting time on exploring this area.

When an agent is instantiated, the value of each parameter is chosen randomly among the intervals that divide the definition set of the parameter. After a model has been evaluated (by running a simulation and computing the fitness of the model), the intervals in which the parameters of the agents have been chosen are rewarded. For each parameter, and for each interval, the reward is proportional to the global fitness of the model and to the number of agents that have taken their parameter's value in the interval.

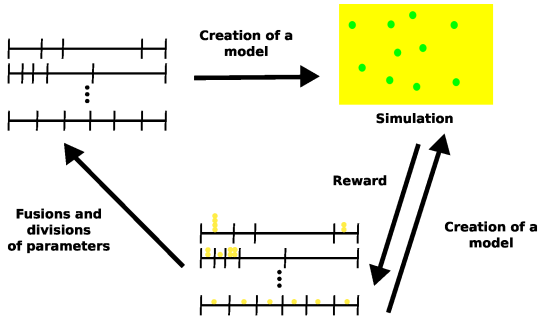

Fig. 1. *ADO* Method outline

The global process of the Adaptive Dichotomic Optimization method proceeds as follows (see figure 1): a model is created, its fitness is computed, and the corresponding intervals are rewarded. This is iterated several times (it can also automatically be distributed on several computers), until the different intervals have received enough rewards so that it becomes significant. For each parameter independently of the others, the best interval (the one having the largest reward) is selected and divided in two. The whole process is itself iterated until it stabilizes.

3.1 Parameters

In this method, we have made the working hypothesis that the application of the *ADO* algorithm on each parameter could be performed independently of the other parameters. In this case, our parameter space is broken down into n parameters divided into m_i intervals (the size of the search space is thus about $n \times \mathbb{R}$). In reality, the parameter space is a space of n dimensions (the size of the search space is about $\mathbb{R}^n$.). Without this hypothesis, in the selection phase, the number of intervals would raise by $2^n - 1$, which would cause serious difficulties. With this hypothesis, the number of intervals only raise by n.

This is not to say that the parameters are mutually independant (the tuning of one parameter will of course influence the tuning of the other parameters) but that we may ignore this in the application of the algorithm. This hypothesis can be intuitively justified by the fact that we run lots of simulations with lots of

agents, which implies that the parameter space is globally covered. If we consider a single parameter, it has been evaluated with the other parameters taking a great number of different configurations. Thus, if an interval has received high rewards, we can be confident that this interval is interesting, disregarding the specific values of the other parameters.

One could think that specific difficulties may arise if the tuning of the model requires that two (or more) parameters take simultaneously a specific value. This has been tested experimentally and the algorithm proved to be valid even in that case. The hypothesis has also been tested with different models, and it appeared to remain correct for all the configurations. Figure 2 shows the distribution of the parameter values for the 100 first simulations in the agent-based model (see the subsection 4.1 page 10). Despite few agents (25 agents) in this model we can see that the distribution globally covers all the space.

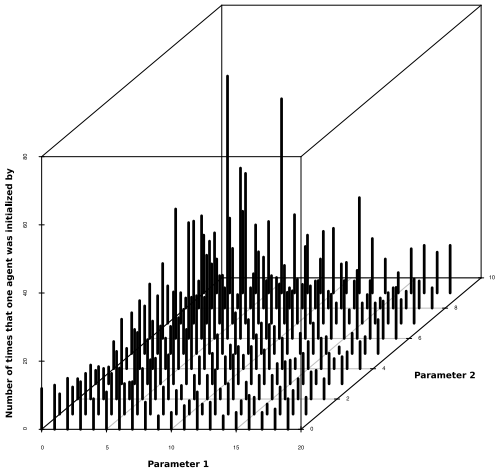

Fig. 2. Distribution of the parameter values for the 100 first simulations

3.2 Method

We have developed several variants of the *ADO* method, which can be differentiated in the way the intervals are rewarded, divided or merged. In this paper, we only present one variant for which we took inspiration from the Ant Colony Optimization approach. We will first present Ant Colony Optimization before exposing our method in detail, and the corresponding results.

Ants Colony. Ant colony algorithms[14–16] can be seen as a part of swarm intelligence[17], that is, the design of intelligent multi-agent systems by taking

inspiration form natural collectively intelligent behaviours in social animals, especially insects.

The principle is inspired by the creation of trails of pheromones, like in the foraging model that we used as an example (see subsection 4.1 page 10). We drew our inspiration from a specific Ant colony algorithm: Ant systems [18]. In the example of the traveling salesman problem (TSP), the algortihm is applied on a completely connected undirected graph where the nodes represent the cities and the edges represent the distances between the cities. A node is chosen as starting node for the ants. Then the ants explore the graph. The probability for ant k to go from the city i to a city j is given by:

$$p_{ij}^k = \begin{cases} \frac{(\tau_{ij})^\alpha \times (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{ij})^\alpha \times (\eta_{ij})^\beta} \text{ if } j \in J_i^k \\ 0 \text{ if } j \notin J_i^k \end{cases}$$

where τ_{ij} is the quantity of pheromone, η_{ij} is the visibility ($= \frac{1}{d_{ij}}$ where d_{ij} is the distance between the cities i and j) (it represents a heuristic information), and J_i^k is a memory of already visited cities.

After the completion of a tour, each ants k lays a quantity $\Delta\tau_{ij}^k$ of pheromones on edge (i,j):

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L^k} \text{ if } (i,j) \in T^k \\ 0 \text{ if } (i,j) \notin T^k \end{cases}$$

where T^k is the tour done by ant k at iteration t, L^k is its length, Q is a parameter.

The quantity of the pheromone is then updated so as to simulate its evaporation, using the formula :

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}$$

where $\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k$ (m is the number of ants)

We can generalize the method with the ant colony optimization metaheuristic [15]. The general schema of the ant colony optimization metaheuristic is:

---

while termination conditions not met **do**
  ScheduleActivities
    AntBasedSolutionConstruction()
    PheromoneUpdate()
    DaemonActions()
  end ScheduleActivities
end while

Moreover our parameter space is a continuous space. An interesting approach in continuous spaces is the algorithm proposed by Socha [19].

The method in detail. Taking inspiration from ACS, we made a parallel between pheromones in the ACS model and the rewards given to the intervals in the *ADO* approach. More precisely, it is the mean reward which is chosen as pheromone.

Moreover we add a heuristic information and a confidence in this heuristic information. For each interval, we compute the heuristic information in three steps. In the first step, we create a model where all the agents are initialized with the same parameter set, which is chosen in the following way (see figure 3):

- for the specific interval, whose heuristic value we wish to estimate, we choose the median value of the interval
- for all the other parameters, we choose the median value of the parameter

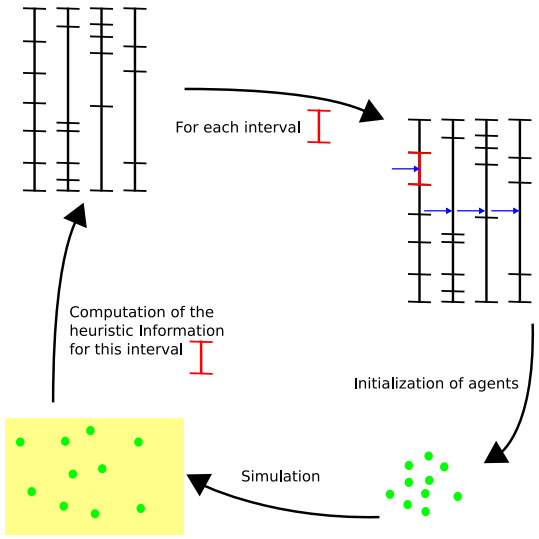

Fig. 3. Computation of heuristic information

In the second step, we simulate this model several times (because of the stochasticity). In the last step, we calculate the heuristic information: the value of the heuristic information is the average value of the fitness for the various simulation runs. The confidence in the heuristic information of an interval is 1 after its calculation. During a fusion stage, the confidence is the average of the two values of confidence for the two intervals. During a division step, the confidence is divided in two. We compute again the heuristic information when the confidence is lower than a given value.

The phase in which the values of the parameter are chosen is modified to distinguish two steps. First, we choose an interval j of the parameter i with a probability proportional to

$$[\tau_{ij}]^{\alpha} \times [\eta_{ij}]^{\beta}$$

where τ is the pheromone quantity in this interval (that is to say the average reward), η is the heuristic information, and α and β are parameters that control the relative importance of the pheromone versus the heuristic information. Second, we choose a value in this interval with a uniform probability.

The division and fusion stages are also modified. An interval j of the parameter i is divided if $\tau_{ij} > \overline{\tau_i} + 2 \times \sigma_i$, where σ_i is the standard deviation. An interval j of the parameter i is merged if $\tau_{ij} < \overline{\tau_i} - \sigma_i$.

To summarize, we can adopt the ACS schema 3.2:

- `AntBasedSolutionConstruction()`: this step corresponds to the creation phase of the simulation with the choice of the parameters values (see figure 4);
- `PheromoneUpdate()`: this step corresponds to the computation of the simulation and the reward of the intervals;
- `DaemonActions()`: this step corresponds to the update of the heuristic information.

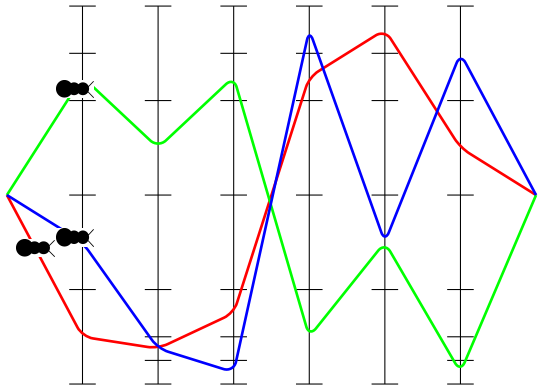

Fig. 4. AntBasedSolutionConstruction step (inspired by ACS): for each parameter (vertical lines), an agent (ant) chooses stochastically an interval based on its heuristic value and on the quantity of pheromones on it; the agent adopts the center of the chosen interval as its value for the given parameter

3.3 Solutions

After the algorithm has converged, it is necessary to compute a candidate solution and to evaluate the fitness of this solution. Indeed, what the algorithm computes is an adaptive discretization of the parameter space. But since the discretization is achieved independently on each axe, this does not lead to a single global solution. We therefore have to recompose such a global solution and to evaluate its fitness. There are several possible ways to compute this final solution.

A first possibility is, at each selection phase of the algorithm, to identify for each parameter, the interval that received the most rewards. The global solution will be composed by all of these intervals. The corresponding fitness will be obtained by running a new simulation (in fact several runs because of the stochasticity) with all the agents initialized with these parameters.

Another possibility to compute the global solution is to consider that the solution interval for each parameter is the one that is smallest. Indeed, it is the one that did receive the most rewards in the previous iterations of the algorithm, and thus that got the most divided. This interval is therefore probably interesting for the global solution. In fact, there are at least two such intervals because during the selection phase, the interval that received the most rewards is divided into two intervals of the same width. In the case of adjacent intervals of minimal width, we therefore take the union of the two intervals. Then, if several intervals of the same width remain, one of them is chosen randomly. The final global fitness is computed as before.

After various tests, the latter possibility has been adopted to compute the solution because it is less variable than the former. Indeed, it is not only dependent on the rewards at the last iteration of the algorithm but also on the rewards of the preceding iterations. In the remaining of this paper, all results are computed with the second version.

4 Results

4.1 Model

We have tested our method on different models. In this paper, we will focus on a single model to illustrate and to test the method. To this end, we decided to use the well-known ant foraging model, provided by the modeling environment platform `NetLogo` [1, 20]. Figure 5 shows an illustration of this model. In this very simple model, the agents are ants, and their aim is to bring food back to their nest: the nest is located in the centre of the environment, and the items of food are dispatched into three areas on the periphery. Initially the ant agents leave the nest and make a random search for food. When an ant agent finds food, it brings it back to the nest secreting a chemical on its way. When other ant agents feel the chemical, they follow the chemical way up to the food source, which reinforces the presence of the chemical and finally produces trails between the nest and the food sources. We can observe ant lines emerging, which are qualitatively similar to the ones that can be observed in natural conditions.

There are several parameters in this model. Among these, two global parameters control the way the chemicals are diffused in the environment:

- `diffusion-rate`: this parameter characterizes the diffusion of the chemical in the environment.
- `evaporation-rate`: this parameter characterizes the evaporation of the chemical.

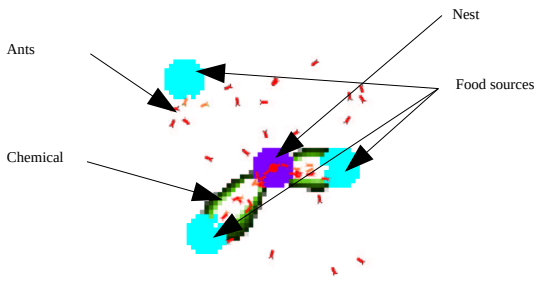

Fig. 5. Example of an ABS: Ant foraging

We also designed several variants that included a larger number of parameters in order to test our approach. This includes, for instance, the addition of a new parameter that characterizes the speed of an ant agent. In this paper, the main modifications are the addition of local parameters for the agents (these parameters were present in the original model, but are not considered to be parameters: they were fixed and could not be modified in the original model):

- `speed`: this parameter characterizes the speed of an agent. It varies between 0 and 20 patches per simulation step.
- `patch_ahead`: this parameter characterizes the number of patches looked ahead to "sniff" the chemical. It varies between 0 and 10 patches.
- `angle_vision`: this parameter characterizes the angle of vision. It varies between 0°and 360°.
- `drop_size`: this parameter characterizes the initial quantity of chemical that the agents drop in the environment when they come back to the nest with items of food. It varies between between 0 and 200.

4.2 Results

Table 1. Optimized parameters values

	Genetic algorithm method	ant colony method
`speed`	7.35	7.69
`patch_ahead`	9.76	9.84
`angle_vision`	193.76	195.18
`drop_size`	109.89	144.99

In our example, the model have 25 ants, we simulate 500 steps. The global parameters remain constant. We try to optimize the four local parameters. The

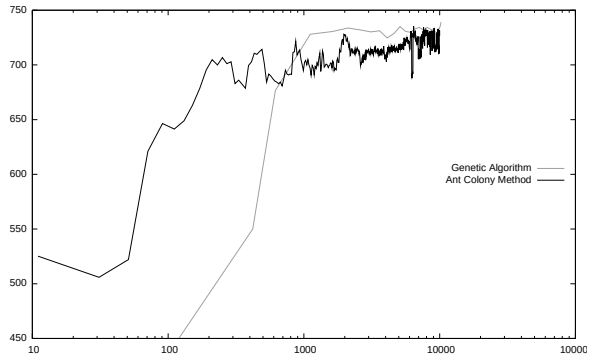

Fig. 6. Evolution of the fitness according to the number of simulation runs

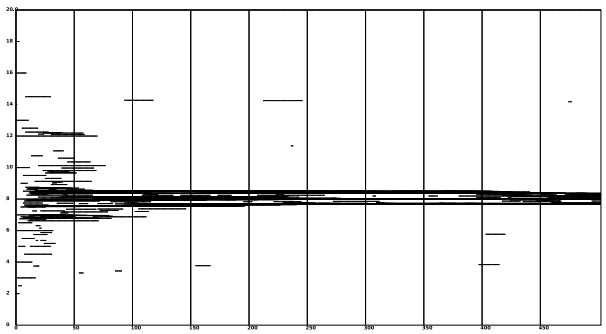

Fig. 7. Evolution of divisions of parameter "`speed`"; the y axis corresponds to the range of variation of the speed parameter (from 0 to 20); the x axis corresponds to the succesive iterations of the *ADO* algorithm; for each iteration (vertically), one can see how the range of variation of the parameter is divided into smaller intervals: in particular, one can see that these intervals get smaller and smaller around the value 7.7, which indicates that the algorithm converged

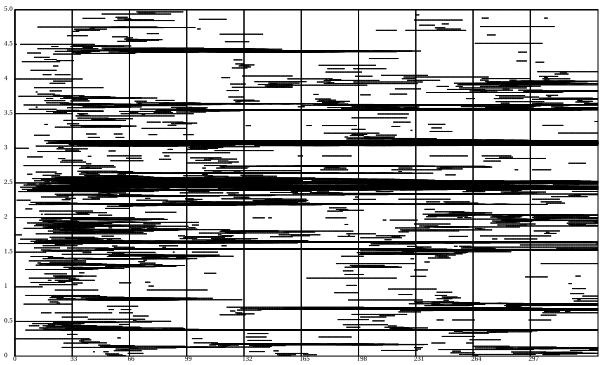

Fig. 8. Evolution of divisions of an artificial parameter; the y axis corresponds to the range of variation of the parameter (from 0 to 5); the x axis corresponds to the succesive iterations of the *ADO* algorithm; for each iteration (vertically), one can see how the range of the parameter is divided into smaller intervals: in this case, the whole range remains explored until the end of the algorithm, which indicates that this parameter has no influence on the fitness of the simulations

fitness is the quantity of food brought back to the nest at the end of the simulation. To compare our method with previous works, we also ran a genetic algorithm method (described in [6]). We stop the algorithm after 10000 simulations. Figure 6 shows the evolution of the fitness according to the number of simulation runs.

The interesting result is that with few simulation runs, the Ant colony version of the *ADO* method outperforms the GA method. In this example, with only 300 simulation runs, we have already a good approximation of the solution.

Moreover we have more information with the *ADO* method than with the genetic algorithm method. Figure 7 presents the evolution of the division of the first parameter in our *ADO* method. At the beginning of the algorithm, the parameter is divided into ten equal parts, then as the method runs, the divisions of the parameter evolve: we have a convergence of the divisions to accurate zones of parameters. This illustrates one of the interests of this method: we start with the entire parameter space before focusing on more interesting parts but without forgetting though the other parts. Instead of providing only an optimal value, the method thus provides a kind of a cartography of the parameter space. Figure 8 shows the evolution of the division of an test parameter (which is not used in the model and whose value doesn't modify the fitness) in our *ADO* method. In this case we don't have a convergence of the divisions but we have divisions spread out over parameter range regularly, which is what is expected: the parameter doesn't have any influence on the model, and whatever its value, the result is the same for the simulation of the model.

We can then study the solution parameters obtained by the two methods. Table 1 shows the optimized parameters. Whatever the method, the solutions have the same characteristics: the agents move at high speed (about 7.5), with a

vision angle about 190°(forward), and with a farthest vision. The initial quantity of chemical dropped off is very high but without maximizing the quantity. To summarize, the best agent is the one that moves quick, and that looks ahead farthest, in front of it, which appears to be pretty logical.

5 Discussion

We presented the general framework of *Adaptive Dichotomic Optimization* and we focused on specific variant inspired by *Ant Systems*. We applied our approach on a simple benchmark, namely the ant foraging problem. The first results appear to be rather promising as they show a quick convergence of the *ADO* method towards a solution that is almost as good as the one that was found with genetic algorithm. Both methods are easily distributable on several computers so as to speed up the process.

The approach also share with genetic algorithms the interest of covering the whole parameter space, but in a more exhaustive way. As compared to local optimization techniques such as a random search or gradient descent, this avoids the risk of being stucked in local optima. As compared to a simple classical dichotomy, this presents the advantage of breaking down the complexity by exploring the parameter space in parallel, using the multiple agents to test simultaneously several settings.

As compared to genetic algorithms, we can also see two major advantages of the *ADO* approach. The first one is that convergence is very fast with few simulation runs, whereas a great number of runs is necessary to bootstrap the genetic algorithm. We can thus achieve a first exploration of the parameter space with only 100 or 200 simulation runs, which is potentially interesting for the modeller during the prototyping phase. The second advantage is that with the *ADO* method, the parameter space can, at any moment of the algorithm, be interpreted as a kind of a cartography of the parameter space, sparse intervals corresponding to low interest regions, dense intervals corresponding to high interest regions. Specific visualisation techniques remain to be developed so as to take advantage of these characteristics.

Finally, we have to underline that the *ADO* approach works best for agents' parameters, not so well for global parameters. However, this should not be too problematic since agents' parameters are generally the ones that are to be estimated, global parameters being estimated from experimental data.

6 Conclusion and Perspective

In this paper, we presented a new algorithm for the automatic calibration of agent-based models. We also presented preliminary results on a simple benchmark with four continuous independent parameters to tune. These results suggest that the method is interesting especially when the execution of simulation runs is very costly, because of its very fast convergence. The argument is that agent-based models present characteristics that make their calibration difficult,

mainly because of the long time required to execute simulation runs, as compared to the fitness functions usually used in genetic algorithms. Another characteristics of agent-based models is that they are concurrent models, property that we exploited so as to propose an efficient algorithm, inspired by the principles of dichotomic search and of ant colony systems. In addition, we underlined that the method enables an exhaustive cartography of the parameter space, which is a real advantage against genetic algorithms.

Further investigations still need to be undertaken to better understand the role of the number of agents in the optimization process. We will also study the signification of the patterns of division for the parameters and develop visualization tools in order to analyze these patterns. If a parameter shows a very densely divided area with very small intervals, this parameter is probably an important parameter in the dynamics of the simulation. On the contrary if the division of a parameter is relatively homogeneous, this parameter may not be essential for the model. Finally, we investigate the possibility to mix the different approaches, namely Adaptive Dichotomic Optimization and Genetic Algorithms, so as to combine the strengths and advantages of both methods.

In addition, we need to enlarge the set of models on which the method is evaluated, with larger parameter spaces. We are now on the process of applying the method to "real" models, that is models with true modelling questions, experimental data, hypothesises and theories. This will be the real test, that is, "can our method give automatic answers to some of the modelling questions that are raised by the modeller?". This is the real challenge.

References

1. Wilensky, U.: Netlogo (1999) Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.
2. Brueckner, S.A., Parunak, H.V.D.: Resource-aware exploration of the emergent dynamics of simulated systems. In: AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM Press (2003) 781–788
3. Fehler, M., Klügl, F., Puppe, F.: Techniques for analysis and calibration of multi-agent simulations. In Gleizes, M.P., Omicini, A., Zambonelli, F., eds.: ESAW. Volume 3451 of Lecture Notes in Computer Science., Springer (2004) 305–321
4. Fehler, M., Klügl, F., Puppe, F.: Approaches for resolving the dilemma between model structure refinement and parameter calibration in agent-based simulations. In Nakashima, H., Wellman, M.P., Weiss, G., Stone, P., eds.: AAMAS, ACM (2006) 120–122
5. Sallans, B., Pfister, A., Karatzoglou, A., Dorffner, G.: Simulation and validation of an integrated markets model. J. Artificial Societies and Social Simulation **6**(4) (2003)
6. Calvez, B., Hutzler, G.: Automatic tuning of agent-based models using genetic algorithms. In Antunes, L., Sichman, J.S., eds.: Proceedings of the 6th International Workshop on Multi-Agent Based Simulation (MABS'05). Volume 3891 of lnai., Luis Antunes, Jaime Simao Sichman (2005) 39–50

7. Calvez, B., Hutzler, G.: Exploration de l'espace de paramètres d'un modèle à base d'agents. In Guéré, E., ed.: 7èmes rencontres nationales des jeunes chercheurs en intelligence artificielle (RJCIA 2005), Presse Universitaires de Grenoble (PUG) (2005) 99–112
8. Rogers, A., von Tessin, P.: Multi-objective calibration for agent-based models. In: 5th Worksho on Agent-Based Simulation. (2004)
9. Narzisi, G., Mysore, V., Bud Mishra, B.: Multi-objective evolutionary optimization of agent-based models: An application to emergency response planning. In Kovalerchuk, B., ed.: The IASTED International Conference on Computational Intelligence (CI 2006). (2006)
10. Wolf, T.D., Samaey, G., Holvoet, T., Roose, D.: Decentralised autonomic computing: Analysing self-organising emergent behaviour using advanced numerical methods. In: ICAC. (2005) 52–63
11. Kevrekidis, I.G., Gear, C.W., Hummer, G.: Equation-Free: The Computer-Aided Analysis of Complex Multiscale Systems. AIChE Journal **50**(7) (2004) 1346–1355
12. Klein, F., Bourjot, C., Chevrier, V.: Dynamical design of experiment with mas to approximate the behavior of complex systems. In: MA4CS. (2005)
13. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Technical Report No. 91-016, Politecnico di Milano, Italy, 1991 (1999)
14. Blum, C.: Ant colony optimization: Introduction and recent trends. Physics of Life Reviews **2**(4) (December 2005) 353–373
15. Dorigo, M., Di Caro, G.: New Ideas in Optimization. In: The Ant Colony Optimization Meta-Heuristic. McGraw-Hill (1999) 11–32
16. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial Life **5(2)** (1999) 137–172
17. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence From Natural to Artificial Systems. Oxford University Press (1999)
18. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics **26**(1) (1996) 29–41
19. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. In: EURO XXI in Iceland. (2006)
20. Wilensky, U.: Netlogo ants model (1998) Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.