Parameter Space Exploration of Agent-Based Models

Benoît Calvez and Guillaume Hutzler

Universite d'Evry-Val d'Essonne/CNRS
LaMI, UMR 8042
523, Place des Terrasses 91000 Evry, France
{bcalvez,hutzler}@lami.univ-evry.fr

Abstract. When developping multi-agent systems (MAS) or models in the context of agent-based simulation (ABS), the tuning of the model constitutes a crucial step of the design process. Indeed, agent-based models are generally characterized by lots of parameters, which together determine the global dynamics of the system. Moreover, small changes made to a single parameter sometimes lead to a radical modification of the dynamics of the whole system. The development and the parameter setting of an agent-based model can thus become long and tedious if we have no accurate, automatic and systematic strategy to explore this parameter space.

That's the development of such a strategy that we work on suggesting the use of genetic algorithms. The idea is to capture in the fitness function the goal of the design process (efficiency for MAS that realize a given function, realism for agent-based models, etc.) and to make the model automatically evolve in that direction. However the use of genetic algorithms (GA) in the context of ABS raises specific difficulties that we develop in this article, explaining possible solutions and illustrating them on a simple and well-known model: the food-foraging by a colony of ants.

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 behavior and interactions are defined. 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). Similarly, Multi-Agent Systems (MAS) are designed so as to accomplish a given function in a collective and decentralized way. The validation of the system is thus given by the fact that the function is realized and that it is efficient. In both cases, 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 behavior 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 approach that we suggest is to consider the problem of the development and the validation of ABS or MAS 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 for ABS would be the distance between the artificial model that we simulate and the real system. The optimization function for MAS would be the efficiency in the realization of the function. Given the large dimensionality of the problem, optimization techniques such as Genetic Algorithms (GA) can then be used to explore the parameter space and find the best parameter set with respect to the optimization function. However the use of genetic algorithms in this context is not so simple, as we will explain.

In section two we present the problematics related to the parameter tuning of an agent-based simulation. Then in section three we present the general framework of genetic algorithms and show the difficulties that arise from the application of these techniques to agent-based simulation.

2   Parameter tuning

2.1   Parameters of agent-based models

In the context of agent-based simulation, a model and the simulator with which it is executed include lots of parameters. These parameters can be of different natures. Some parameters are peculiar to the simulator: the discretization step for the modeling of time and space for instance can be a fixed feature of the simulator. As a consequence, these parameters can generally not be modified by the user. For this reason, we do not include this type of parameters in our parameter space. We only include the parameters that are specific to the model. Some of them can be extracted from the knowledge of the field (either experimental or theoretical) and can thus be associated to fixed values. Other parameters have to be kept variable, which can be for different reasons: on the one hand, the knowledge of the field is generally not exhaustive (which is the reason why we build a model and simulate it); on the other hand, this knowledge may not be directly compatible with the model. In this case, a common approach can be to try some values and simulate the model to see how it behaves globally. What we propose is to have a general approach to automate this long and tedious process.

2.2   Objective

Depending on the motivation of the modeling work, the criteria used to explore the parameter space will also be different. This motivation may be to model

and simulate a real system, but it can be to study the discrete models that may produce a given emergent phenomenon. Finally, the motivation may be to propose models that perform best in the realization of a specific function.

In the first case, we want to check if the simulated model correctly grasps the behavior of the real system. The validation of the model will thus be to have a behavior identical to (as close as possible) experimental knowledge. The search problem can be seen as the search of the parameter set that minimizes the distance between real and simulated data.

Having a similar behavior can also mean that specific emergent phenomena known to occur in a real system can be observed in the simulation. Emerging ant lines for example, will only occur if the chemical trails leaved by the ants behind them (see next paragraph) have specific properties, as we will see in next section. The emergence of this phenomenon will thus be associated to specific parameter values, and the search problem will consist in searching the different ranges of parameters where an emergent phenomenon is observable. In some cases, choosing slightly different values may lead to completely different results during the simulation, which complicates a manual exploration of the parameter space and justifies the development of automatic techniques.

2.3   Example

We will present the parameter setting of an agent-based model with the example of ant foraging (search for food), in which ants leave chemical trails behind them when coming back to the nest with food (we use the multi-agent programmable modeling environment `NetLogo` [1] and its ”Ants” model).

In this model, two parameters condition the formation of chemical trails. The first one is the diffusion rate of the chemical, which corresponds to the fact that a given proportion of the chemical will be diffused to the neighboring patches (regions of the environment) at the next time step. This is used to simulate the diffusion of the chemical in the atmosphere. The second parameter is the evaporation rate of the chemical, which corresponds to the fact that a given proportion of the chemical will disappear from the patch at the next time step. This is used to simulate the evaporation of the chemical in the atmosphere.

For example, we can be interested more precisely in the dynamics of ant lines. Table 1 shows three models with small modifications for the two parameters. We can obtain different dynamics: the difference lies in the way that food sources are exploited. In model 1, food sources are exploited in turn while in model 3, they are all exploited at the same time. As a result, we observe one, two or three ants lines.

2.4   Previous work

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. However when we have lots of parameters (real-valued parameters
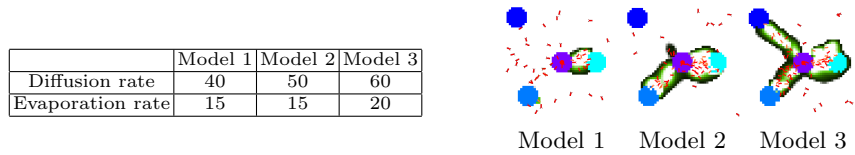
	Model 1	Model 2	Model 3
Diffusion rate	40	50	60
Evaporation rate	15	15	20

Model 1   Model 2   Model 3

Fig. 1. Models with slightly different parameters.

for example), the parameter space becomes huge and the systematic exploration becomes impossible.

Other methods have been proposed, which differentially explores 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` . However, to avoid a systematic exploration, they use searcher agents and introduce the fitness notion. 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.

3   Use of genetic algorithms

As the tuning of the parameters of a model is a strongly combinatorial problem, we propose to use genetic algorithms, which generally provide good results on problems of this kind.

3.1   Choice of the fitness function

If we consider the exploration of the parameter space as an optimization problem, we need to define very carefully the function that will have to be maximized by the algorithm. This fitness function is of fundamental importance since the models that will be selected are the one that perform best with respect to this function. In the context of agent-based simulation, the choice of the fitness function is problematic for several reasons: as a first thing, it is not the result of a computation but the dynamics of a process that has to be assessed; secondly, emergent phenomena may be difficult to characterize quantitatively since they are often related to a subjective interpretation by a human observer.

Quantitative vs. qualitative. Validating an agent-based model by assessing the distance between the simulation and the real system can be done either quantitatively or qualitatively.
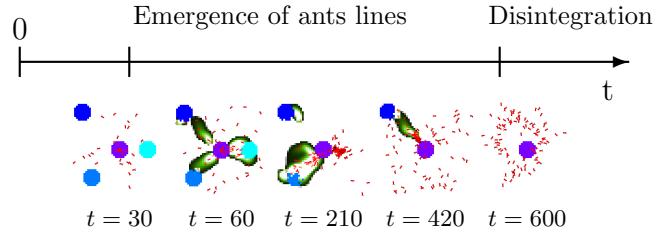
Fig. 2. Ant foraging at different time-steps

In the quantitative case, data are measured in the simulation and compared to data measured in similar conditions in the real system. The distance between the simulation and the real system is then the Euclidean distance between the two data vectors. If we try to select models that are optimized for the realization of some function, the fitness function can also be directly measured by the performance of the system for that function.

In the qualitative case, what is important is that a given emergent phenomenon be present in the simulation: for example, the ants line. The difficulty is then to translate this observation into a quantitative measure (the fitness function). In some cases, the characterization of such emergent phenomena may not be so simple since it may be the result of a subjective interpretation by an observer, which cannot be captured easily by a quantitative measure.

A dynamic process. In classical optimization problem, the fitness function corresponds to the result of a computation. Therefore, the question of the time at which the measure should be made doesn't make sense: the measure is done when the computation has ended. On the contrary, agent-based simulations are dynamic processes that evolve along time and generally never end.

We can clearly see in the example given in the previous section that the evaluation of the fitness function generally has to be done at a given time-step of the simulation. The choice of this time-step is not neutral and may greatly influence the performance of the genetic algorithm and the resulting model.

Figure 2 shows the foraging simulation at five different time-steps. We can see that the ant lines are not present during all the simulation. This example shows the difficulties to choose the time-steps for the evaluation.

3.2   Computation of the fitness function

Time. Since no mathematical model can anticipate the dynamics of an agent-based model without executing it, the computation of the fitness function requires one or even several simulations. This means that the time required to compute the fitness function will be significant. We must therefore find methods to reduce either the number of chromosomes, the time to converge towards an optimum or the time to compute the fitness function. We mainly studied the last possibility through distributed computation and fitness approximation.

Distributed computation. Since the different models are independent from each other, the evaluation of their fitness is also independent. Therefore each evaluation of the fitness (that is to say each agent-based simulation) can be done on a different computer. We can thus have several computers to simulate the models and use the master-slave parallel genetic algorithms [3], which improves the performance as compared to standard GA.

Fitness approximation. Fitness approximation comes to approximate the result of the simulation by a mathematical model, such as a neural network or a polynomial for instance. We tried this approach by training a neural network with test data. After the learning phase, we used it to compute the fitness, with the generation-based control approach, in which the whole population of η generations is evaluated with the real fitness function in every λ generations [4]. The results however were not so good and this approach has been temporarily abandoned. We suspect in that case that the approximation was not good enough to obtain satisfying results but this has to be explored in more details.

Stochasticity. Two agent-based simulations can generally bring slightly different results even if the underlying model is exactly the same due to the stochasticity of the model and of the simulator. One simulation is not enough to evaluate the fitness function: it can only be considered as an estimate for the fitness.

In such noisy environments, a first solution is to increase the size of the population [5]. To multiply the number of the simulated models reduces the effect of the stochasticity. A second solution is to simulate each model several times to improve the evaluation of the fitness function. Both solutions greatly increase the number of simulations, thus the time, of the genetic algorithm.

Another solution is to use the same technique as with fitness approximation. A solution to the stochasticity problem is then to estimate the fitness of each model with one simulation, and each n generations of the GA (n to choose according to the stochasticity of the model and the desired quality of the estimation), to estimate the fitness of each model with x simulations.

We use the elitism genetic algorithm [6] that is to say we keep the best chromosomes during the algorithm, which allows to continuously improve the solution. Our implemented genetic algorithm replace only 25 % of the population at each generation. Every 3 generations, we estimate the fitness of the models with more simulations. The interest to choose these values is to keep always the best chromosomes.

4   Discussion & Conclusion

We applied the method to some simple examples: the ant foraging with different fitness functions (both quantitative and qualitative). We do not show the results because of the lack of space. As we could already see with the study of the stochasticity, we obtained very different results depending on the choice of the fitness function. The models are strongly optimized for a specific fitness function

and may not perform so well with another one. The optimization creates a loss of the flexibility of the dynamics of the agent-based model. A possible solution would be to use several different initial conditions to evaluate the fitness function. This would however increase again the time necessary to run the algorithm.

The optimization by the genetic algorithm also depends on the constraints imposed to the agents in the model. If a model has lots of constraints (fewer resources for example), it is necessary that it optimizes its global functioning. On the contrary, if the resources are abundant, the pressure on the model to adapt and optimize its functioning will be weaker. As a result, the use of our approach will be mostly beneficial when constraints on the model are high.

The next step is to apply the method to a more complex example. We began a work for the simulation of the glycolysis and the phosphotranferase systems in Escherichia coli. In this work, we are interested in testing the hypothesis of hyperstructures [7]. The hyperstructures are dynamic molecular complexes, enzyme complexes in the case of this work. These complexes allow to improve the behavior of a cell : more flexibility, quicker adaptation. In our study, we have 25 kinds of molecules (or agents). There are altogether about 2200 agents in the simulation. We want to study the potential interest of hyperstructures for the cell. To do this we make the rates of enzymes association and dissociation variable. In this context, the simulation of a model lasts about 10 minutes, which imposes to use the methods described in this article like the distributed computation. To explore this complex example, we will need to develop additional strategies to reduce the parameter space (e.g. by introducing coupling between parameters), to accelerate the evaluation of the fitness function (e.g. by developing approximation methods), and to accelerate the convergence of the algorithm (e.g. by using interactive evolutionary computation). Finally, another important perspective is to explore the effect of varying dynamically the simulation conditions so as to produce more versatile models.

References

1. Tisue, S., Wilensky, U.: Netlogo: Design and Implementation of a Multi-Agent Modeling Environment. Proceedings of Agent 2004 (2004)
2. Brueckner, S., Parunak, H.V.D.: Resource-Aware Exploration of the Emergent Dynamics of Simulated Systems. AAMAS 2003 (2003) 781–788
3. Cant-Paz, E., Goldberg, D.E.: Efficient parallel genetic algorithms: theory and practice. Computer Methods in Applied Mechanics and Engineering **186** (2000)
4. Jin, Y., Olhofer, M., Sendhoff, B.: A Framework for Evolutionary Optimization with Approximate Fitness Functions. IEEE Transactions on Evolutionary Computation **6** (2002) 481–494
5. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. Complex System **6** (1992)
6. Beker, T., Hadany, L.: Noise and elitism in evolutionary computation. In: Soft Computing Systems - Design, Management and Applications. (2002) 193–203
7. Amar, P., Bernot, G., Norris, V.: Modelling and Simulation of Large Assemblies of Proteins. Proceedings of the Dieppe spring school on Modelling and simulation of biological processes in the context of genomics (2002) 36–42