Automated Multi-Agent Simulation Generation and Validation

Philippe Caillou

Automated Multi-Agent Simulation Generation and Validation (Early Innovation)

Philippe Caillou
Laboratoire de Recherche en Informatique (LRI)
Universite Paris Sud
Orsay, France
caillou@lri.fr

ABSTRACT

Multi-agent based simulation (MABS) is increasingly used for so-
cial science studies. However, few methodologies and tools ex-
ist. A strong issue is the choice of the number of simulation runs
and the validation of the results by statistical methods. In this arti-
cle, we propose a model of tool which automatically generates and
runs new simulations until the results are statistically valid using
a chi-square test. The choice of the test configuration allows both
a general overview of the variable links and a more specific inde-
pendence analysis. We present a generic tool for any RePast-based
simulation and apply it on an Academic Labor Market economic
simulation.

Keywords

Multi-Agent Based Simulation, Simulation Validation, Simulation
Tool, Chi-square test, statistical test

1. INTRODUCTION

Multi-Agent Based Simulations (MABS) are increasingly being
considered as flexible and versatile modeling frameworks, enabling
positive and normative investigations of phenomena out of reach
when one uses analytical studies[2, 14]. However, few method-
ologies exist on MABS usage. The main problem of MABS is
validation: since simulations are by definition too complex to be
validated analytically (otherwise they are only useful to inspire an-
alytical analysis), other methods have to be considered. The result
of a simulation is a set of observations (for example a set of evac-
uation times for a simulation of a stadium fire evacuation). As for
empirical observations, statistical tools can be used to validate re-
sults obtained by the simulation[1]. Their usage is growing, even if
the expert validation is still mainly used [7]. One important con-
dition to be able to apply most statistical tests is to have a large

---

[1]We consider here the validation of the results considering the
model is sound. For example, statistical tests can validate the fact
that the most important variable for the evacuation time is the num-
ber of exits in the simulation. However the model in itself - the
agent behaviors, the stadium model - needs to be validated sepa-
rately.

enough number of observations. Compared to empirical observa-
tions or biological experimentations, MABS has a big advantage:
it is easy and almost free to generate new simulation results. Our
goal here is to use this advantage to generate automatically new
simulations until observed results are statistically valid.

An ideal tool would work with any simulation framework and
would select the best statistical test considering the experimenter
goal. As a first step, we will however begin with a single test and a
single framework. The chosen framework is RePast[9] as one of the
most used framework for MABS. To be generic, we aim to keep the
interaction with the framework minimal: the tool reads and write
parameter files, starts the simulation program and analyzes result
logs.

One of the first results of any MABS is usually the link between
the observed variables and the parameters, which ones are the most
important parameters and which ones have an influence on the fi-
nal result. As our goal is to propose a generic tool, we consider
here a test with a small number of hypotheses on the tested vari-
ables: the Pearson's Chi-square test. It tests if two observed vari-
ables/parameters are independent or not with a known percentage
of error (by comparing the observed distribution of data with an ex-
pected distribution obtained from the distribution of the two tested
variables considered independently).

Our objective here is to propose a model for a generic tool which
automatically run new simulations from any RePast simulation un-
til the independence results (obtained by chi-square tests) on se-
lected variables are statistically valid. To test and illustrate our
method, we apply a corresponding tool on a simulation of the French
Academic Labor Market (presented - without statistical validation
- in [4, 5]). The main parameters describe the hiring system proper-
ties and the candidates and universities utility functions, while the
observed variables measure the quality of the hiring and the rate of
jobs fulfilled by local candidates.

In the following, we present the state of the art in simulation
methodology and tools in section 2. Then we describe the method,
the chi-square test, propose some heuristics to increase the analy-
sis efficiency and discuss the possible applications of our tool in
section 3. The simulation of the French academic labor market is
introduced in section 4, some results are presented in section 5 and
finally we conclude.

2. RELATED WORK

A variety of social and economic problems have been investi-
gated using multi-agent systems (MAS) [2]. MAS have demon-
strated their ability to represent (cognitive) agents and constrained
interaction rules, and provide insightful pictures of the dynamics of
the system [11]. Several frameworks are available, such as RePast[9],
NetLogo[16] and ModulEco[10] (see a review in [12]). The use of

automated simulation generation and analysis is not yet integrated in these frameworks. NetLogo has the *BehaviorSpace* tool (and its corresponding API), but it is mainly the equivalent of RePast parameter files: it allows the user to choose parameter ranges to launch multiple experiments.

The closest approach to our work is the "robot scientist"[13] developed to achieve biological experiments autonomously. Our approach represents the equivalent for Multi-Agent Simulation, and improves it with efficiency heuristics and the statistical tests both for analysis and as a termination criterion. Another similar approach is the SimExplorer project [1]. Its goal is to manage simulations parameters and results with a generic framework. It is an ongoing project, with many limitations on parameter values, no possibility to "program" simulation runs with stopping criteria. Their work is very complementary to our objective, since they have no specific analysis tool in their system. An extension of our work would be to integrate it into SimExplorer to improve its interface with our goal-oriented statistical analysis.

Calibration and validation have always been a serious issue for MABS. Few general methodologies have been proposed, due to the huge variety of simulation types. Some classifications of empirically observed methods have been done ([17][8]). A survey of methods used from 1998 to 2008 [7] shows that usage of statistical tools for validation stays marginal (less than 10%) even if it has increased since 1998. The chi-square test is one of the most used for simulation analysis since it has very few requirements. A good introduction to chi-square test can be found in [18] and a more precise discussion on the test in [6].

3. MODEL

In this section we will first make a general presentation of our model (3.1), then we will precise the notations (3.2) we use. The Chi-squared test is described (3.3), then we detail the method algorithms (3.4), we propose some heuristics to improve our tool efficiency (3.5), and finally we discuss the objectives achievable with our tool (3.6).

3.1 Presentation

The global objective is simple and is summarized Fig.1: on a simulation model, the user chooses some parameters and observed variables, and the tool has to run simulations and make statistical tests until the tests are valid for each couple of parameters/variables. It finally indicates to the user which couple of variables are not independent and the error margin.

More precisely, we can distinguish between four main steps:

- **Parameters and variables extraction**: Starting with an existing Parameter file and result log, the tool extracts the set of parameters and variables used by the simulation and their past values.

- **Configuration and objective choice**: The user chooses which observed variables and parameters he wants to test, and gives eventually more configuration details (specific number of classes, specific parameter ranges, ...).

- **Parameter file generation and simulation run**: The tool generates a new parameter file and starts the simulation.

- **Results update**: The observed data set is updated from the result log. For each couple of variables/parameters a Chi-square test is made. If it is valid for every couple, the results are presented, otherwise the previous step is executed again.
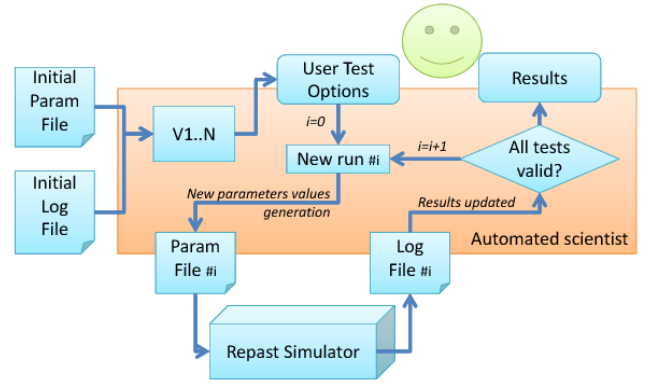

Figure 1: Model overview: Starting with a configuration and a result file, the tool extracts the model variables. The user chooses the configuration. The tool generates, runs and analyzes new simulations until results are statistically valid, and finally present the results to the user.

3.2 Notation

The following notations will be used in this article:

- $V = \{v^1..v^{p+r}\}$ for the set of variables, including[2]:

- $v^1..v^p$ for the simulation parameters.

- $v^{p+1}..v^{p+r}$ for the observed variables.

- $U \subseteq V$ for the set of selected variables for the independence tests (an independence test is made for every couple $v^i \times v^j$ with $v^i \in U$ and $v^j \in U$.

- $x^1..x^{p+r}$ for observed values for variables $v^1..v^{p+r}$ after a simulation run.

- c^i for the number of classes used for variable v^i.

- l^i_t and l^i_{t+1} for the lower and upper bounds for class t of variable v^i.

- $n.$ for the total number of observations.

- n^i_t for the number of observations for class t of variable v^i.

- $n^{ij}_{tt'}$ for the number of observations simultaneously for class t of variable v^i and class t' of variable v^j.

- $e^{ij}_{tt'}$ for the number of expected observations for class t of variable v^i and class t' of variable v^j.

- Pr^{ij} is the probability for variable v^i and v^j for not being independent according to Chi-square test.

- $nbrun$ for the number of simulation runs between each result update and statistical test.

---

[2]Since the test is identical and we study both parameter/observed variable and observed variable/observed variable independence, we use the same notation for parameters and observed variables

Table 1: Assistant professor candidate sex and hiring success, France, 2007.

$n_{tt'}^{12}$	Men	Women	Total $n_{t'}^2$
Hired	1081 (11.7%)	725 (7.9%)	1806 (19.6%)
Not Hired	4234 (46.0%)	3173 (34.4%)	7407 (80.4%)
Total n_t^1	5315 (57.7%)	3898 (42.3%)	9213 (100%)

Table 2: Expected distribution

$e_{tt'}^{12}$	Men	Women	Total $e_{t'}^2$
Hired	1041,9	764,1	1806
Not Hired	4273,1	3133,9	7407
Total n_t^1	5315	3898	9213

3.3 Chi-Square test

The Pearson's Chi-square test of independence applies to two binned variables (variables put into classes). The tested hypothesis is H_0: Variable v^1 is independent from v^2. Simply put, it tests if the lines and the columns of a contingency table are independent. To test the hypothesis, the observed distribution (the contingency table) is compared to the expected distribution if the variables were independent. If H_0 is true, then $P(v^1 \bigcup v^2) = P(v^1) \times P(v^2)$.

For example, to test if the sex of the candidate (v^1) has an impact on the hiring probability as assistant professor (v^2), we consider the data presented table 1. The two variables v^1 and v^2 have two classes: $c^1 = c^2 = 2$.

From this observed distribution, we compute in table 2 the expected distribution (if H_0 is true) : $e_{tt'}^{12} = \frac{n_t^1 * n_{t'}^2}{n_{..}}$

The value of the test can be computed from the sum of the squared differences divided by the expected distribution value:

$$X^2 = \sum_{t,t'} \frac{(n_{tt'}^{12} - e_{tt'}^{12})^2}{e_{tt'}^{12}}$$

Here, $X^2 = 4,317$. This test value has to be compared to the Chi-square law value $\chi_{df,\alpha}^2$ to know if the hypothesis can be rejected or not with $\alpha\%$ probability (see for example [18] for a mathematical definition of the $\chi_{df,\alpha}^2$ function). The number of degree of freedom is the product of the number of classes minus 1 : $df = (c^1 - 1)(c^2 - 1)$. Here df=1. With an error margin of 5%, the value of the Chi-square function is $\chi_{1,0.95}^2 = 3.84$. We can thus reject the hypothesis "The sex of the candidate has no impact on the hiring" with only a 5% error margin. We could not have done it with an error margin of only 1%, since $\chi_{1,0.99}^2 = 6,63$.

One advantage of this test is the absence of hypothesis on the form of the variable distribution. To apply it on continuous variables, the only requirement is to sample them by defining automatic or user-defined classes.

To be valid, the test has 2 requirements: the independence of the observations and a sufficiently large data set. A usual condition [15] is that the expected population has to be greater than one ($e_{tt'}^{12} > 1$) for every cell of the contingency table and greater than 5 ($e_{tt'}^{12} > 5$) for 80% of the cells. We use this condition, applied on every analyzed variable couple, to determine when to stop the simulations.

3.4 Model

The main algorithm of our tool model (Alg. 1) follows the basic steps presented in section 3.1 :

```
ExtractVariables();
ChooseObjective();
ChooseConfig();
ResValid=false;
while ResValid=false do
    GenerateRunSimulation() → ResFile;
    UpdateResults(ResFile);
    UpdateClasses();
end
ShowResults();
```
Algorithm 1: Global simulation analysis algorithm
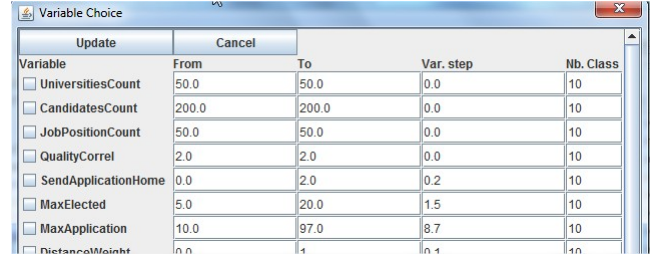

Figure 2: Variable choice window: for each variable, the user checks if he wants to select it for the independence test, and chooses eventually the number of classes and their lower and upper bound.

$ExtractVariables()$ is mainly a parsing function that extract parameters names $v^1...v^p$ and values from a parameter file and variables names $v^{p+1}...v^{p+r}$ and values from a result log file.

$ChooseObjective()$ let the user choose the variables that will be tested (U) and, eventually, to set manually the number and bounds of the initial classes for each variable ($c^1...c^{p+r}$) - see Fig. 2. By default, the first simulation run is used to determine $c_{default}$ uniform classes for each variable. The choice of classes uniform in size rather than uniform in observation number is made for two reasons: first, the number of observations is low at the beginning of the process and the distribution is very likely to change. Second, it is very common to have very concentrated variables (for example, 75% of the observations have an unique value for one of the observed variable of our application example). To use very small classes for v around a concentrated value x would influence the results of the test, because the variables which influence v around x would have more influence on the independence test than variables which influence v only for other values than x (because classes will switch very quickly around x, whereas bigger changes are needed to switch classes elsewhere). More discussions about user-defined classes can be found in section 3.6.

$ChooseConfig()$ let the user choose the parameters values for the simulations (see Fig. 3). RePast parameter files accept sets and loops (For..step..until). For the test to be valid, experiments have to be independent, which is not true for loops. For this reason, we replace loops by random choices between possible values. For each parameter, the user can choose between a set (containing one or several values) and a random value with or without steps (a random value without steps is a continuous uniform random variable). By default, the values are the ones used in the current parameter file. More discussion about user-defined parameter values can be found in section 3.6.

$GenerateRunSimulation()$ creates a new parameter file using the chosen values for $nbrun$ new simulation runs, creates the
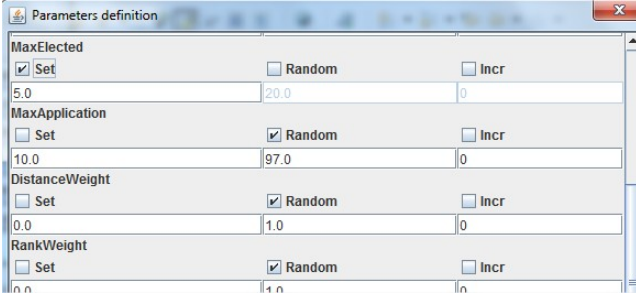
Figure 3: Parameter configuration window: for each parameter, the user chooses between a set of values (set), a continuous random variable (random) and a set of values with fixed steps (incr).

adapted script file (to save the results in a new file) and starts the simulations. To increase the tool efficiency, the random values for one parameter can be biased (see section 3.5).

> **while** *new res $x^1..x^{p+r}$ in ResFile* **do**
> $inc(n.)$;
>   **for** *each v^i, v^j with $v^i \in U$ and $v^j \in U$ and $i <= j$* **do**
>   **for** *each class c_t^i* **do**
>   **if** $l_t^i \leq x^i < l_{t+1}^i$ **then** $inc(n_t^i)$
>   **end**
>   **for** *each $c_t^i \times c_{t'}^j$* **do**
>   **if** $l_t^i \leq x^i < l_{t+1}^i$ and $l_{t'}^j \leq x^j < l_{t'+1}^j$ **then**
> $inc(n_{tt'}^{ij})$
>   **end**
>   **end**
> **end**
> $ResValid = true$;
> **for** *each v^i, v^j with $v^i \in U$ and $v^j \in U$ and $i <= j$* **do**
> $Nb5 = 0$;
>   **for** *each $c_t^i \times c_{t'}^j$* **do**
> $e_{tt'}^{ij} = \frac{n_t^i * n_{t'}^j}{n.}$;
>   **if** $e_{tt'}^{ij} < 1$ **then** $ResValid = false$;
>   **if** $e_{tt'}^{ij} < 5$ **then** $inc(Nb5)$;
>   **end**
> $X^{ij} = \sum_{t,t'} \frac{(n_{tt'}^{ij} - e_{tt'}^{ij})^2}{e_{tt'}^{ij}}$;
> $Pr^{ij} = pr$ with $\chi^2_{(c^i-1)(c^j-1),pr} = X^{ij}$;
>   **if** $Nb5 > 0.2c^i c^j$ **then** $ResValid = false$
> **end**

Algorithm 2: $UpdateRes(ResFile)$ algorithm

$UpdateRes(ResFile)$ function is described in Alg. 2. It updates the observation contingency tables (the $n_{tt'}^{ij}$) in the while loop and computes the Chi-square value and check the test validity criteria in the For loop. The program stopping criteria ($ResValid$) is the Chi-square validity criteria: each expected distribution value ($e_{tt'}^{ij}$) has to be greater than 1 and 80% have to be greater than 5 for every selected variable couple. The Pr^{ij} value is the minimum error margin to reject the hypothesis H_0 (v^i and v^j are independent). This value gives more information than a binary answer independent/not independent since the user can compare the values between variables.

$UpdateClasses()$ is used to merge or create new classes in case the user didn't define specific classes. Classes are added when an observed value is out of the lowest/highest bounds, if the user did not choose infinite boundaries or force fixed classes. Both options allow the user to keep a fixed number of classes, without losing any observation (infinite boundaries) or by focusing on a specific range (fixed classes). A class merging heuristic can also be used to increase the class definition efficiency (see section 3.5). The reason why new uniform classes are not redefined at each loop is that it would suppose to consider all the past results again (since the only information kept is the contingency tables with the populations in each class).

$ShowResults()$ finally presents the Pr^{ij} (which can be interpreted as probability of not being independent) for each couple of selected variable to the user. The user can have the detail of observed/expected observations for every couple by selecting it (see Fig. 4 and 5).

3.5 Heuristics

The total time required for one simulation run is usually low, but not negligible, and for some models it can even be relatively high. Moreover, several thousand simulations may be required to obtain statistically valid results. For these reasons, we propose two heuristics to decrease the required number of simulations without losing the statistical properties:

The **class merging** heuristics: During the *UpdateClasses()* step, one class is merged with its smallest neighbor when its population is lower than $minclass\%$ the average class size. This heuristic is here to prevent a class existing only because of an exceptional configuration/error to block the test and the program (if a class population is very small, expected values will be lower than 1 and the process will never stop). The main advantage of this heuristic is its efficiency (see results in section 5.4). Its downside is that it can not be applied to not orderd/string variables (because it is impossible to know which classes to merge) or when classes are fixed (when the user wants to analyze fixed classes).

The **biased values** heuristic: During the *GenerateRunSimulation()* step, the idea is to identify the most problematic variable and class (the one with the lowest number of observations), the parameter which will increase its population with the highest probability (the one with a high proportion of observations in the selected class) and to bias its values accordingly. Formally:

- Choose class t and variable i with $n_t^i = \min_{j,t'} n_{t'}^j$.

- For each continuous parameter v^j, give a score to each of its classes: $s_{t'}^j = \left(\frac{n_{tt'}^{ij}}{\sum_t n_{tt'}^{ij}}\right)^2$.

- For the next simulations set generation, choose the biased parameter v^j with the highest total score $s^j = \sum_{t'} s_{t'}^j$.

- For each random value of this parameter, choose the class t' of v_j with probability $\frac{s_{t'}^j}{\sum_t s_t^j}$. The final value is selected uniformly in class t'.

The main advantage of this heuristic is that it can be applied to any variable. It can however be applied to only one parameter for each simulation generation: To apply it to several parameter at the same time may introduce artificial relations and would invalidate the statistical validity of the test.

3.6 Discussion

The main objective of our method is to test the independence of variables. However, several more precise objectives can be achieved:

- The first intuitive objective is to present a global overview of the variable interactions in the simulations. Simulations can use hundreds of parameters and variables and a first overview of the important parameters for each variable is already very useful and not trivial to obtain with current simulations framework. This is the default objective, with all default parameter values and all variables selected (see section 5.1 for an illustration).

- To change the parameters values may have a huge impact on the results. Specific submodels can thus be analyzed, for example with a fixed parameter. More interestingly, to limit the parameter range for all the parameters may give significantly different results: some variables may be interdependent only for very specific and unrealistic parameter values. With these constraints, it is possible to study the variables relations around a specific point (for example with +/- 10% variation only - see section 5.2).

- Finally, it is possible to manually define some variable classes number/range to alter the results. To choose more classes means a more precise test. But it may be interesting to know what are the variables which have the *highest influence* on a specific variable v^1 by decreasing its number of classes. The effect will be that only the variables which have an important impact on v^1 will influence the observed classes, and thus be considered as not independent by the test (see section 5.3).

4. APPLICATION EXAMPLE: THE FRENCH ACADEMIC LABOR MARKET

Our application is a simulation of the French assistant professor academic labor market. As it is just an illustration for our simulation tool, a precise understanding of the model is not necessary (interested reader may refer to [4, 5]) and we will just give a brief overview here.

Let $\{u_1, \ldots, u_U\}$ and $\{c_1, \ldots c_C\}$ respectively denote the set of universities and candidates, listed according to their quality. University u_t is characterized from four parameters. The first two parameters (in $[0, 1]$) govern his preference ordering: i) elitism e_t stands for its bias toward the best candidate; ii) locality $_t$ stands for its bias toward local candidates. Lastly, a random perturbation modeled as $(1 - e_t)V$ with V uniformly drawn in $[0, 1]$, accounts for the "subjective" preferences of university u_t. Overall, the quality $r(i, t)$ of candidate c_i for university u_t is:

$$r(i, t) = (r_t \times \frac{i}{C} + (1 - e_t)V)(1 - {}_t.L(i, t))$$

where $L(i, t)$ is 1 iff c_i is local to u_t and 0 otherwise.

University u_t uses its risk-adversity parameter r_t to decide which candidates will be interviewed among those applying to u_t. Its strategic ordering is defined as:

$$s(i, t) = r_t \times r(i, t) + (1 - r_t) \times \frac{|i - (t)|}{C}$$

The candidates parameters, quality function $r'(i, t)$ and strategic ordering $s'(i, t)$ are symmetrical.

Interaction Rules Every candidate c_i applies for positions after its preference ordering, and to his home university with probability h_i. Every university u_t produces a shortlist of 5 candidates. Every candidate c_i thereafter ranks the universities having shortlisted him. Eventually, the candidate and university ranking are aggregated by a variant of Stable Marriage algorithm [3], an optimal matching is derived, and the recruitment decisions are made accordingly.

Figure 4: Result window with default parameters: each cell value is the probability for the couple of variables not to be independent. Stars indicate interdependent variables with an error margin lower than 5%.

Agents parameters are defined as random uniform variables, the boundaries are the parameters of the simulation. For example:

- Application number $NbApp_i \sim U[MinApp, MaxApplication]$,

- Risk factor $r_t \sim U[MinUnivRisk, MaxUnivRisk]$

The two main observed variables are the number of positions fulfilled ($NbHire$) and the number of positions fulfilled by local candidates ($NbLocalHire$). Two other variables evaluate the quality of the hiring process: the rank of the last hired candidate ($LastRelACand$) and the rank of the first university with no hired candidate ($FirstRelNAJob$).

5. RESULTS

To illustrate our tool model, we used it on the simulation application with the three possible objectives described in section 3.6. First, we get a global overview of the simulation, then we try a more precise test around the equilibrium and we test the influence of class number on the results. Finally, we test the efficiency of the proposed heuristics[3].

5.1 Global overview

With all default parameters, all parameters and four observed variables selected for test, we obtain a global view of the simulation (Fig. 4). On the top of the window is indicated the number of simulations runs completed until all tests were valid (4182)[4]. Each line and column corresponds to a variable, with the first four lines for the observed variables (results of the simulation), and the next lines for the parameters. Each cell contains the test results. The meaning of a positive test value (value higher than 0.95 on the figure, signaled with a star) is that the two variables are not independent with 5% of error margin. Interestingly, the test values for parameter/parameter couples (all values except the first four lines) are not all low. The parameter values are all random variables (using Java standard Random function - similar results were obtained with other random number generators). Nevertheless, for several parameters, the hypothesis that they are independent has to be rejected for a high enough error margin (15% for the couple MaxLocalBonus/SendApplicationHome). The lesson here is that it is important to take a very low error margin to be sure that the independence is rejected (5%, 1% or less).

For the user, the figure gives a good insight on the simulation behavior: the parameter $MaxApplication$ has clearly the most

---

[3] For each test, we use $nbrun = 102$ runs for each step and $c_{default} = 10$ classes by default for each variable.

[4] The whole analysis, with the 4182 simulations and 250 agents took approximately 6 minutes with our test configuration.

Figure 5 (Detail window):

Nb. XP						
Dif sum	1402.958591343494					
DL	99					
Proba indep.	0.0					
Proba not indep.	1.0					
NBHire	MaxApplication	10.0>	18.7>	27.4>	36.1>	44.8>
4182	4182	420	422	413	395	455
0.15>	93	0.0/9.52683	0.0/9.5722	0.0/9.36805	0.0/8.95976	7.0/10.3
0.19>	61	0.0/6.24878	0.0/6.27854	0.0/6.14463	1.0/5.87683	4.0/6.76
0.21>	175	0.0/17.92683	0.0/18.0122	2.0/17.62805	10.0/16.85976	23.0/19.
0.239>	239	0.0/24.48293	0.0/24.59951	5.0/24.07488	14.0/23.02561	18.0/26.
0.268>	328	0.0/33.6	3.0/33.76	10.0/33.04	16.0/31.6	45.0/36.
0.297>	353	0.0/36.16098	4.0/36.33317	12.0/35.55829	38.0/34.00854	64.0/39.
0.326>	303	0.0/31.03902	8.0/31.18683	40.0/30.52171	43.0/29.19146	49.0/33.
0.355>	288	3.0/29.50244	27.0/29.64293	51.0/27.74634		35.0/31.
0.384>	302	8.0/30.93659	50.0/31.0839	62.0/30.42098	33.0/29.09512	32.0/33.
0.413>	311	26.0/31.85854	53.0/32.01024	51.0/31.32756	36.0/29.9622	32.0/34.
0.442>	454	84.0/46.50732	74.0/46.72878	51.0/45.7322	42.0/43.73902	47.0/50.
0.471>	1193	299.0/122.20976	203.0/122.79171	141.0/120.17293	111.0/114.93537	99.0/13

Figure 5: Detailed view for the independence test of parameter $MaxApplication$ **and observed variable** $NBHire$**: each cell contains the observed and expected observation count.**

influence on the observed variables. Interestingly, this was one of the result of the initial analysis ([4]), but without statistical proof, only with "clear evidence" on a plot. Here, it is possible to say that this is the only parameter for which the independence hypothesis can be rejected for all observed variable with the lowest margin of error. Other analysis can be maid with statistical proof : for example, the fact that the candidates are risk-taker or risk-adverse ($MinCandRisk$ and $MaxCandRisk$ parameters) have very low influence on the result.

To have more details about the observed values and the test, the user can select any cell to obtain the contingency table with both the degree of freedom (DL), X^{ij} value (Dif Sum), observed and expected values (in each cell). For example, Fig. 5 details the observations for parameter $MaxApplication$ (maximum number of applications sent by a candidate) and $NbHire$ (hiring rate). It is possible to see that the parameter has a negative impact on the observed variable.

5.2 Parameters influence

The previous analysis was global, every parameter explored its whole definition space. To know which variables have an influence around a specific point, it is possible to limit the parameter range. We have tried here to test all parameters and variables, but to limit the parameter range to a 10% variation around a specific equilibrium (the empirically observed equilibrium, see [4]). Results are presented Fig. 6. In this situation, some variables (like the first one, $SendApplicationHome$) have disappeared because their value is fixed.

In this situation, the variables influences are clearly different. The impact of $MaxApplication$, for example, is very different because its range dropped from [10,97] to [18,22]. Excessive values which conduct to a saturation of the hiring process are eliminated. Interestingly, even without this saturation effect, this parameter has still influence on some qualitative ($FirstRelNAJob$) and quantitative ($NBHire$) observed variables. Moreover, with the elimination of excessive values, some influences which were previously hidden because they were too small may appear: The influence of $MinCandRisk$ is low compared to the global influence of other variables (Fig. 4), so low that the independence hypothesis can not be rejected. But around the equilibrium this low influence becomes the strongest (Fig. 6).

5.3 Variables classes influence

Finally, it is possible to study more precisely some variables by manually choosing the classes used for the analysis. Lower-

Figure 6 (Population window):

	MaxApplication	DistanceWeight	RankWeight	MinCandRisk	MaxCandRisk	MinUnivRisk
Nb. XP						
3468						
LastRelACand	0.91423	0.7009	*1.0	*1.0	*1.0	0.70544
FirstRelNAJobs	*0.99992	0.79006	0.82513	*1.0	0.84114	0.79328
NBLocalHire	0.8412	*1.0	0.84399	*0.98821	0.67382	0.23346
NBHire	*0.9991	*0.99963	*1.0	*1.0	0.20851	0.90913

Figure 6: Result window for a limited parameter range around a specific point.

Figure 7 (top Detail window):

Nb. XP							
Dif sum	142.49730729647922						
DL	45						
Proba indep.	4.64672744726613E-12						
Proba not indep.	0.9999999999953533						
NBLocalHire	RankWeight	0.0>	0.19>	0.38>	0.57>	0.76>	0.95>
4182	4182	219	482	426	441	446	228
0.0>	1839	67.0/98.22951	186.0/216.19463	172.0/191.07659	166.0/197.80463	176.0/200.04732	112.0/1(
0.046>	1040	73.0/55.55122	136.0/122.26341	122.0/108.05854	125.0/111.86341	130.0/113.13171	53.0/57.
0.092>	451	38.0/24.09	69.0/53.02	51.0/46.86	51.0/48.51	54.0/49.06	27.0/25.
0.138>	305	21.0/16.29146	47.0/35.8561	34.0/31.69024	28.0/32.8061	26.0/33.17805	18.0/16.
0.184>	287	10.0/15.33	21.0/33.74	38.0/29.82	43.0/30.87	36.0/31.22	9.0/15.9
0.276>	178	10.0/9.5078	23.0/20.92585	9.0/18.49463	28.0/19.14585	24.0/19.36293	9.0/9.89

Figure 7 (bottom Detail window):

Nb. XP				
Dif sum	2.62228669934629			
DL	2			
Proba indep.	0.26951173400514516			
Proba not indep.	0.7304882659948548			
NBLocalHire	RankWeight	0.0>	0.63333>	1.26667>
1224	1224	359	366	475
0.0>	1023	297.0/306.0475	315.0/312.015	411.0/404.9375
0.15333>	177	62.0/52.9525	51.0/53.985	64.0/70.0625

Figure 7: Detailed view for independence test of parameter $RankWeight$ **and observed variable** $NBLocalHire$**. In experiment 1 (top window), the number of classes is 10 for** $RankWeight$ **and 6 for** $NBLocalHire$**. In experiment 2 (bottom window), it is respectively 3 and 2. Each cell contains the observed and expected observation count.**

ing the number of classes decreases the precision of the test, but this "blurredness" can help to identify the variables which have the strongest influence. For example, Fig. 7 details the observations for parameter $RankWeight$ (the importance of the quality of the university for the candidate) and the observed variable $NBLocalHire$ (number of hired local candidates) in two experiments: in experiment 1 (corresponding to the experiment of section 5.1, top figure), the default number of classes is 10. For experiment 2 (bottom figure), it is set to 3. Moreover, for each experiment, some classes where automatically merged because their observed population was too small. In the experiment 1, even if the variables are statistically not independent (with a very low error margin, less than 0.1%), the influence doesn't appear clearly when looking directly at the observed and expected population values (Fig. 5 is an opposite example of apparent interactions between the variables). The decrease of the observation precision in experiment 2 confirms this observation: with few classes, the independence hypothesis can not be rejected anymore.

5.4 Heuristic efficiency

To evaluate the heuristics efficiency, we have applied our tool on the same configuration with each heuristic enabled/disabled (10 times for each situation). The selected configuration used 10 classes for each variable, some of these classes were rather rare and thus difficult to obtain. The average number of required simulations and the variance are given Table 3. The heuristic efficiency was statistically tested (with a Chi-Squared test, of course) and every hypothesis "The heuristic HC/HB has no effect on the number of required simulations" can be rejected with less than 1% of error probability.

Table 3: Average number of simulation runs (and variance) with/without the Class-merging heuristic (HC) and the biased-values heuristic (HB)

	HC	$\overline{HC}$
HB	4760(1517)	16442(1943)
$\overline{HB}$	6800(1819)	24112(579)

The HC (class merging) heuristic appears to be the most efficient and decreases significantly the number of required simulations. When this heuristic can not be used (for example with string variables), the HB heuristic may still be useful. Even if its efficiency is lower, it does not require any specific configuration to be applied.

6. CONCLUSION

In this paper, we have presented a tool model to help the scientist using Multi-Agent Based Simulation to explore its simulation and obtain statistically valid results. We applied a corresponding tool on an Academic market simulation, and we have shown that it successfully generates and runs new simulations until Chi-square independence tests on selected variables are valid. It presents a global overview of the simulation results with the most important variables and the main interactions. It can also be used to obtain more precise results on the simulation behavior for specific parameter ranges, or focus on a specific couple of parameters/variables. We proposed heuristics to decrease the number of required simulations and tested their efficiency.

The first step to continue this work will be to generalize it to other statistical tests and simulation frameworks. A complementary goal would be to integrate this tool in the project of generic simulation explorer SimExplorer.

7. REFERENCES

[1] Frédéric Amblard, David R. C. Hill, Stéphan Bernard, Jérome Truffot, and Guillaume Deffuant. http://www.simexplorer.org/.

[2] R. Axelrod, 'Advancing the art of simulation in the social sciences', *Advances in Complex Systems*, **7**(1), 77–92, (2004).

[3] Mourad Baiou and Michel Balinski, 'Student admissions and faculty recruitment', *Theor. Comput. Sci.*, **322**(2), 245–265, (2004).

[4] Philippe Caillou and Michèle Sebag, 'Modelling a Centralized Academic Labour Market: Efficiency and Fairness', in *ECCS08*, Jerusalem Israel, (2008). Complex Systems Society.

[5] Philippe Caillou and Michèle Sebag, 'Pride and Prejudice on a Centralized Academic Labor Market', in *Artificial Economics 09*, LNEMS, pp. 29–40, Valladolid Espagne, (2009). Springer-Verlag.

[6] P. E. Greenwood and M. S. Nikulin, *A Guide to Chi-Squared Testing*, Wiley, New York, NY, 1996.

[7] Brian Heath, Raymond Hill, and Frank Ciarallo, 'A survey of agent-based modeling practices (january 1998 to july 2008)', *Journal of Artificial Societies and Social Simulation*, **12**(4), 9, (2009).

[8] Scott Moss, 'Alternative approaches to the empirical validation of agent-based models', *Journal of Artificial Societies and Social Simulation*, **11**, (2007).

[9] M.J. North, N.T. Collier, and J.R. Vos, 'Experiences creating three implementations of the repast agent modeling toolkit', *ACM Transactions on Modeling and Computer Simulation*, **16**(1), 1–25, (2006).

[10] D. Phan, 'From agent-based computational economics towards cognitive economics', in *Cognitive Economics*, Handbook of Computational Economics, 371–398, Springer Verlag, (2004).

[11] Denis Phan and Frederic Amblard, *Multi-agent Modelling and Simulation in the Social and Human Sciences*, Bardwell Press, http://www.bardwell-press.co.uk/, septembre 2007.

[12] Steven F. Railsback, Steven L. Lytinen, and Stephen K. Jackson, 'Agent-based simulation platforms: Review and development recommendations', *Simulation*, **82**(9), 609–623, (2006).

[13] L. Soldatova, A. Clare, A. Sparkes, and R. D. King, 'An ontology for a robot scientist', *Bioinformatics*, **22**, 464–471, (2006). ISMB06.

[14] Leigh S. Tesfatsion, 'A constructive approach to economic theory', in *Handbook of Computational Economics*, volume 2 Agent-Based Computational Economics of *Handbooks in Economic Series*, North-Holland, (2006).

[15] Cochran WG, 'Some methods for strengthening the common chi-square tests', *Biometrics*, (10), 10–417, (1954).

[16] Uri Wilensky. http://ccl.northwestern.edu/netlogo/.

[17] Paul Windrum, Giorgio Fagiolo, and Alessio Moneta, 'Empirical validation of agent-based models: Alternatives and prospects', *Journal of Artificial Societies and Social Simulation*, **10**, (2007).

[18] Thomas H. Wonnacott and Ronald J. Wonnacott, *Introductory statistics [by] Thomas H. Wonnacott [and] Ronald J. Wonnacott*, Wiley New York,, 5th ed. edn., 1990.