Calibrating Agent-Based Models Using a Genetic Algorithm

Enrique CANESSA[1], Sergio CHAIGNEAU[2]

[1] Facultad Ingeniería y Ciencias, CINCO, Universidad Adolfo Ibáñez,
Avda. Padre Hurtado 750, Viña del Mar, Chile
ecanessa@uai.cl

[2] Centro de Investigación de la Cognición, Facultad de Psicología,
Universidad Adolfo Ibáñez,
Diagonal las Torres 2640, Santiago, Chile
sergio.chaigneau@uai.cl

Abstract: We present a Genetic Algorithm (GA)-based tool that calibrates Agent-based Models (ABMs). The GA searches through a user-defined set of input parameters of an ABM, delivering values for those parameters so that the output time series of an ABM may match the real system's time series to certain precision. Once that set of possible values has been available, then a domain expert can select among them, the ones that better make sense from a practical point of view and match the explanation of the phenomenon under study. In developing the GA, we have had three main goals in mind. First, the GA should be easily used by non-expert computer users and allow the seamless integration of the GA with different ABMs. Secondly, the GA should achieve a relatively short convergence time, so that it may be practical to apply it to many situations, even if the corresponding ABMs exhibit complex dynamics. Thirdly, the GA should use a few data points of the real system's time series and even so, achieve a sufficiently good match with the ABM's time series to attaining relational equivalence between the real system under study and the ABM that models it. That feature is important since social science longitudinal studies commonly use few data points. The results show that all of those goals have been accomplished.

Keywords: Agent-based modelling, genetic algorithms, calibration, validation, relational equivalence, complex adaptive systems.

1. Introduction

Agent-Based Models (ABMs) (among other tools) are particularly well-suited to studying Complex Adaptive Systems (CAS) [1], cf. [2]. Generally speaking, a CAS is one where numerous elements, parts, or agents (homogeneous or not) interact non-linearly with each other and with their environment such as that their properties may be modified as a result of those interactions [3]. Traditional approaches to studying CAS [4] often limit our ability to understand the full complexity of these systems, in part because the characteristics of CAS (e.g., path dependent dynamics) violate many of the statistical assumptions necessary to use those approaches (i.e., survey research, controlled experiments, game theory) [1].

The distinguishing feature of ABMs is that they are constructed in a "bottom-up" manner, by defining the model in terms of entities and dynamics at a micro-level, i.e., at the level of individual actors and their interactions with each other and with the environment [5], [6], [7]. An ABM consists of one or more types of agents, and possibly a non-agent environment (e.g. in a prey-predator ABM, the environment could be the prey's food; e.g.: if the prey are sheep and predators are wolves, the environment could be the grass.). Agent definitions include specification of their capabilities to determine particular behaviours, as well as decision-making rules and other mechanisms that agents use to choose their own behaviours. Agents may also have adaptive mechanisms that allow them to change based on their experience (e.g., changing the state of agents' memory to reflect prior interactions). While an ABM is running, agent behaviour is generated as agents choose which other agents to interact with and what to do in a given interaction. Thus, ABMs embody complex interlaced feedback relationships, leading to the non-linear, path-dependent dynamics often observed in CAS.

While ABMs and all formal models increase our knowledge about the behavior of any system consisting of similar processes, the use of such models to make inferences about particular real-world systems requires model validation (i.e., showing that model behavior or parameters are comparable to those of a real system). However, model validation is not trivial [8], [9], [10]. For exploratory CAS models, one approach to validation is to focus

on relational equivalence [11]. In general it is impossible to expect matching the detailed behavior of a CAS model to the real system [4], [8]. Thus, validation can be done by matching patterns and relationships between the model and the system being modeled, rather than matching details [11], [9].

Bearing in mind the above mentioned characteristics of the validation of ABMs that model CAS, we present a tool based on Genetic Algorithms (GA) that allows to find the combination of values for input variables to the ABM that establishes relational equivalence between the ABM and the real system it represents. The general idea is to have a GA that will deliver combinations of input parameters to a certain ABM, which will produce outputs of the ABM that match as close as possible the corresponding time series of data gathered from the real system. Because values delivered by the GA must then be judged by a domain expert to finally see whether they are reasonable and select the ones that better make sense from a substantive point of view, we prefer to refer to the GA-based tool as a calibration method, instead of a validation one.

In the next sections we discuss the limitations of current GA-based calibration methods and present a new tool that tries to lessen those drawbacks. Then, we apply the new tool to calibrate three increasingly complex ABMs and conclude that the tool works as expected, but that new improvements may be desirable.

2. State of the Art

The idea of using a GA to calibrate an ABM is not new. However, most of previous efforts have focused on applying GAs to specific models and a more comprehensive tool is still needed. For example, [12] present a GA that calibrates an ABM of oil retail markets. In [13] similar techniques are used to calibrate an ABM of financial markets. The same is done in [14] to calibrate an ABM that models financial trading in markets. In [15], an ABM of terrorist and security scenarios is explored through the use of a GA. Finally, [16] use a GA to analyze an ABM of emergency response planning. Although all those studies have their own merit, they lack the development of a general GA-based tool to calibrate different ABMs. More importantly, those proposals exhibit problems, some of which we try to lessen in the present work.

A common problem recognized by all previous studies on the automatic calibration of ABMs is the long computational time of such tools [17]. Since generally ABMs' outputs are non-deterministic time series [1], the fitness function generally uses the expected value (E) $E[\mathbf{y}_t^m - \mathbf{y}_t^r]$ and $E[\mathbf{s}y_t^m - \mathbf{s}y_t^r]$ as indicators of the match between the output time series of the ABM ($\mathbf{y}_t^m$) and the real system ($\mathbf{y}_t^r$). These expressions involve that the fitness function of the GA must calculate the expected value of the difference in mean and standard deviation between points of the time series generated by the ABM and the corresponding ones of the real system. To have a reliable value for the expectations, previous studies suggest evaluating each point of the time series around 100 times [14], [17] , which entails running 100 times the ABM for the entire time span considered. Since ABMs normally use many computational tasks for representing the behavior of many agents to run every simulation step, that process is very time consuming. Thus, a first goal of the proposed GA calibration tool is to shorten such time, but achieving a decent relational equivalence between the ABM and the system being modeled. A second and related objective is to give the researcher a means of meeting the necessary trade-off between reaching a very close relational equivalence and the computational time it will take the GA to deliver the corresponding solution. Hence, the GA-based tool should have some mechanism that could allow the experimenter to reach that required balance.

A third goal of the GA-based tool is to be easily applicable to many ABM models. All of the previous studies code the GA as part of the ABM computer program, thus increasing the coupling between the GA's and ABM´s computer code [12, 13, 14, 15, 16]. This makes it difficult to apply the GA's code to new models. Thus, the GA should be coded so that it is fully parameterized and can be used with a wide range of ABM simulation platforms.

Finally, with regard to ABMs used in the social sciences, a fourth objective is to achieve a good relational equivalence using only a few data points of the real system. This goal is important, given that in social sciences, longitudinal studies are difficult to perform and

generally two points in time are used [18]. Here again, previous studies indicate that GA-based tools are generally developed assuming that many data points of the real system will be available [12, 13, 14, 15, 16].

3. Using a GA to Calibrate an ABM

As already explained, a common problem in the approach followed by previous ABM calibration GA-based tools is the computational cost of that tool. Thus, in the present work, instead of using the expected value of outputs of the time series of an ABM to compute a fitness value, we propose to use a fitness function based on the simple aggregation of the difference $|y_t^m - y_t^r|$ for a set of data points selected by the experimenter. Additionally, in order to deal with the possible differences in scale of the time series and thus develop a tool applicable to any output of an ABM, we use the absolute value of the percentage difference, as follows:

$$fitness = \frac{1}{1 + \sum_{n=0}^{k} \sum_{t=0}^{t_n} \left| \frac{y_{nt}^m - y_{nt}^r}{y_{rt}^r} \right|} \quad (1)$$

Expression (1) implies that the user will define a set of k output variables of the ABM and real system, which will be evaluated at t_n points in time. Then the absolute value of the percentage difference for each point will be calculated and aggregated by simply summing them up. For the fitness to be normalized to the [0, 1] interval, the GA adds one to the double summation and takes the reciprocal of that value. In that form, the GA must maximize the fitness.

The fitness in (1) allows performing fewer evaluations of the ABM's time series. The more points are considered for evaluation, the better the match between the ABM's and real system's time series might be, but the slower the GA will iterate. However, an intelligent selection of the evaluation points, will allow the experimenter to balance the speed of the GA and the required precision of the match. As already discussed, because in ABM one normally intends to reach relational equivalence, a few points may achieve a sufficiently good fit, as we will show in the results. Thus, the present GA allows balancing a relatively low computational cost with an appropriate calibration of an ABM.

Another goal of this work is to build a GA-based calibration tool that should be simple to use with many types of ABMs. For that reason, we fully parameterized the code of the GA and developed it as a library that can be easily linked to any ABM written in the Netlogo [19] or any Java-based ABM platform. We used Netlogo, since it is a very good and user-friendly ABM platform, with nice Graphical User Interfaces (GUIs) and thorough user documentation, and that runs on many computers, such that it is used by many researchers, especially in the social sciences [20], cf. [2].

Table 1 shows the pseudo code of the initial GA (GA1).

Table 1. Genetic Algorithm 1 Pseudo Code

ABM Calibration Genetic Algorithm version 1 (GA1)

a. Set up:
   k = nr. of time series of ABM and real system, k = 1 to 10
   t at which series will be evaluated = $t_0, t_1, \ldots, t_n$
   y_t^r = time series of the real system
   pop-size = population size in the interval [5,200]
   p_c = cross-over probability in the interval [0,1]
   p_m = mutation probability in the interval [0,0.3]
   max-iter = maximum nr. of iterations of the GA [1,500]
   stop-fitness = fitness that when reached will stop iteration of the GA in the interval [0,1]
   L_p = lower limits for the p adjustable parameters of the ABM
   U_p = upper limits for the p adjustable parameters of the ABM
b. Generate initial population of *pop-size* chromosomes each with p genes. Sample value of each gene i from U(L_i ,U_i) (i = *1,…,p*)
c. ***do while*** (*number of iterations* ≤ *max-iter* ***and*** max (*j = 1, …, pop-size*) $fitness_j \leq$ *stop-fitness)*
d. Calculate fitness for the *pop-size* chromosomes:
   a. Evaluate k outputs of ABM at time t
   b. Compute *fitness* for each chromosome according to expression (1)
e. ***for*** j = 1 **to** *cross-over-count = (pop-size* * p_c*)/2* ***do***
f. Execute tournament selection of size 3 among population, selecting two parents
g. Execute one-point crossover for selected parents, using a randomly-chosen cross-over point
h. ***end for***
i. ***for*** j = 1 **to** (*pop-size - cross-over-count * 2*) ***do***
j. Execute tournament selection of size 3 among population, selecting one chromosome
k. Clone selected chromosome
l. ***end for***
m. ***for all*** population of chromosomes ***do***
n. ***for*** each gene i of chromosome ***do***
   a. ***if*** $u \sim$ U(0,1) ≤ p_m ***then*** replace gene i with value ~ U(L_i ,U_i)
o. ***end for***
p. ***end for***
q. ***end while***

We can see that GA1 is a continuous GA (i.e. GA in which each gene has a value in R) that uses tournament selection, with a tournament size of 3, one-point crossover, and a mutation operator that acts on randomly-selected genes of chromosomes. The stopping condition is

reaching either a maximum number of iterations or a user-defined value for the fitness function.

Some preliminary tests of GA1 showed that we could further improve it to speed up its convergence. The following changes allowed a faster convergence, as we will demonstrate in the results section. The first change focused on reducing the number of solutions (chromosomes) that must be evaluated. Remember that the most expensive computational part of the GA is the evaluation of the ABM's time series. Thus, if many similar solutions exist in the population (in terms of genes and fitness), it would be a waste of time to evaluate each of them. Hence, before evaluating each solution, the GA eliminates those that are similar to a certain degree (β), which can be set by the experimenter. This is simply accomplished by the code on lines 4 a. and 4 b. of the pseudo code listing of GA2 presented in Table 2.

Table 2. Generic Algorithm 2 Pseudo Code

ABM Calibration Genetic Algorithm version 2 (GA2)

1: Set up:
   k = nr. of time series of ABM and real system
   t at which series will be evaluated = $t_0, t_1, \ldots, t_n$
   $\mathbf{y}_t^r$ = time series of the real system
   pop-size = population size
   p_c = cross-over probability
   p_m = mutation probability
   max-iter = maximum nr. of iterations of the GA
   stop-fitness = fitness that when reached will stop iteration of the GA
   L_p = lower limits for the p adjustable parameters of the ABM
   U_p = upper limits for the p adjustable parameters of the ABM
   β = number of significant digits of values of genes and fitness in the interval [2,10]
   γ_L = lower limit of maximum allowable percentage difference between ABM and real system time series
   γ_U = upper limit of maximum allowable percentage difference between ABM and real system time series
   α = discount factor to calculate γ_t in the interval [0,1]
2: Same as in GA1 and set $\gamma_t = \gamma_U$
3: ***do while*** (*number of iterations* ≤ *max-iter* ***and*** max (j = *1, …, pop-size*) $fitness_j \leq$ *stop-fitness)*
4: Calculate fitness for the *pop-size* chromosomes:
   a. Round to β significant digits values of genes and *fitness* of each chromosome
   b. Eliminate chromosomes that have an equal value for their genes and *fitness*
   c. Evaluate k outputs of ABM at time t
      i. ***if*** for any time series j at time t : $\left| (y_{jt}^m - y_{jt}^r)/ y_{jt}^r \right| \geq \gamma_t$ ***then*** immediately terminate evaluation of time series j and assign *fitness* = 10^{-4} to corresponding chromosome
   d. Compute *fitness* according to expression (1) for all chromosomes that have not being assigned *fitness* = 10^{-4}
5 to 16: Same as in GA1
17: Update $\gamma_t = \gamma_L + (\gamma_U - \gamma_L)/ (1 + \text{number of iterations} * \alpha)$
18: ***end while***

Another improvement implemented in GA2 was to immediately discard "bad" solutions during the calculation of the fitness of the chromosomes. The code corresponding to that refinement can be seen on lines 4 c. i., 4 d. and 17 of Table 2. The improvement consists in calculating the absolute value of the percentage difference between the simulated and real data point at each specified time t (see the expression of the inner summation in (1)) and comparing it with a threshold γ_t. If the percentage difference exceeds that threshold, the ABM's current output time series is already a bad approximation to the real time series and thus, it is futile to continue evaluating the rest of the next data points. Since it would be expected that the first iterations of the GA2 would contain worse solutions than later iterations (i.e. as the GA iterates and converges, the solutions should become better), the threshold γ_t is narrowed as the GA2 iterates. The pseudo code on line 17 in Table 2 shows that γ_t is decreased by simply calculating γ_t as a function of an user-defined upper (γ_U) and lower limit (γ_L), a discounting factor α and the number of iterations performed by the GA2.

We acknowledge that discarding very similar and "bad" solutions might hinder the exploration capabilities of GA2 and overly induce exploitation. That might cause a premature convergence of GA2 and thus avoid GA2 from finding better solutions. However, remember that one of the main objectives of the designed GA is to shorten computational time at the expense of delivering solutions that achieve a very close match between the ABM's and real system's time series.

4. Results of the Calibration of Two ABMs

To show the use of GA1 and GA2 and also verify whether GA2 converges faster to a good solution than GA1, we used two ABMs. The first one corresponds to an ABM that calculates the probability of true ($p(a1)$) and illusory agreement ($p(a2)$) between two agents, according to the Concept Agreement Theory (CAT, [21]). CAT is a theory about the conditions under which individuals infer that they share their conceptualization about something (e.g., whether a certain political figure is an *authoritarian* or a *leader*). It assumes that a conceptualization C can be

described by a probability distribution of k_1 properties (i.e., things that are viewed as consistent with that conceptualization). It also assumes that people who share a conceptualization do not share exactly the same conceptual content (i.e., people accept only s_1 properties as true of *C*, where $s_1 \leq k_1$) and that alternative *Cn* conceptualizations (with their own k_2 and s_2 parameters) will share some of their coherent properties with the focal *C* conceptualization (e.g., that *determination* may be a property of an *authoritarian* figure and also of a *leader*). The number of these shared properties is the *u* parameter. To infer agreement (or lack of), an individual with *C* in mind verifies if some other individual provides evidence of sharing part of her conceptual content (i.e., says or does something which she views as acceptable content for *C*). True agreement (event *a1*) occurs when that other individual produces a property coherent with *C* and she in fact is thinking that *C* in her mind. Illusory agreement (event *a2*) occurs when that other individual instantiates a coherent property but really holds *Cn* in her mind. The probability distributions of the properties of *C* and *Cn* (which can include *u* common elements), can be empirically estimated by collecting conceptual properties from a sample of individuals. This allows the estimation of k_1, k_2, s_1, s_2 and *u*, thus allowing the probabilities of each agreement type (i.e., *p(a1)* and *p(a2)*) to be computed.

The ABM that computes these probabilities works as follows. An observer agent O looks for agreement. To compute *p(a1)*, O chooses a sample of size s_1 from a population *C* of k_1 properties (with a given distribution). Concurrently, another actor agent A chooses a sample of size s_1 from the same *C* population, and both agents check if there is a shared property in their samples. If it is so, this increases an *a1* coincidence counter by one. Computing *p(a1)* on the long run, simply amounts to getting the proportion between this counter and the total number of simulation steps. To compute *p(a2)*, the same process occurs, but A chooses a sample of size s_2 from a *Cn* population of k_2 properties (with a given distribution). Now, if there is a coincidence, this increases an *a2* coincidence counter by one. Computing *p(a2)* on the long run, simply amounts to getting the proportion between this counter and the total number of simulation steps. Given values for all the parameters, a group of simulated agents will converge to the agreement probabilities implied by those values, which will be computed using a moving average of the time series of *p(a1)* and *p(a2)*. Note that this ABM may be used to compute *p(a1)* and *p(a2)* from empirically obtained distributions of conceptual properties.

The second ABM models the strength (salience) of concepts in agents' minds by using the *p(a1)* and *p(a2)* probabilities. Here we briefly describe that ABM, giving just enough details of it to allow the reader to understand the context in which GA1 and GA2 will be tested. More details of the ABM may be found in [22]. In this second ABM, agents perform the same type of process described before, searching for confirming evidence for their conceptual content, but instead of using this to compute *p(a1)* and *p(a2)*, each time they find agreement of any type they increase their *C* concept's strength, which is represented by a *c* coefficient. Conceptual strength directly influences the probability that agents will act according to *C* when it comes their turn to be actors (A), and therefore increases the probability that they will offer confirming evidence to other agents in the simulation (i.e., the more their concept *C* demonstrates being useful to understand other agents, the more they use *C* to guide their own actions). Thus, given some input values of *p(a1)* and *p(a2)*, this ABM tracks the dynamical changes in the strength of concept *C* in a group of agent's minds. Since in a social group there may be more than one version of a concept *C*, the ABM allows setting the number of different versions of the concept that will be present in the group. Note that we currently don't have empirical measurements to calibrate this ABM, so here we will use it in calibrating synthetic data.

4.1 Calibration of a simple dynamics ABM

This experiment uses the ABM that calculates *p(a1)* and *p(a2)*. Because in this case we have empirical estimates for *p(a1)* and *p(a2)*, the GAs must find parameters that will replicate those values, and which should be close to empirically measured ones.

The ABM's two output time series converge quite rapidly to a steady-state equilibrium value, with a very short transient period. Furthermore, given that the calculations of *p(a1)* and *p(a2)* are based on a moving average, their values are very constant. Thus, the ABM's

dynamics are simple. We selected this situation, because we wanted to first assess the performance of the GAs under a simple scenario, so that we could easily verify their correct functioning. Additionally, we expected that the improvements of GA2 over GA1 for speeding up the convergence not be too significant, since the dynamics is simple. To obtain the values of the real system for *p(a1)* and *p(a2)*, we collected data from a sample of individuals, as discussed above (see Table 3). Regarding the parameters presented in Table 3, note that to reduce the amount of information we needed to handle, we calculated a stepwise probability distribution for the *C* and *Cn* elements. In our study, *C* was "male oriented professions" (a set of professions which our sample thought that they would be predominantly preferred by males, such as mechanical and civil engineering), and *Cn* was "female oriented professions" (a set of professions which our sample believed would be mostly preferred by females, such as nursing and elementary school teacher).

Table 3. Variables measured to calculate p(a1) and p(a2) for a real situation

Parameter	Description	Value
k_1	nr. elements in set *C*	61
s_1	size of sample drawn from *C*	47
k_2	nr. elements in set *Cn*	61
s_2	size of sample drawn from *Cn*	44
u	nr. of common elements shared by *C* and *Cn*	61
np_1	nr. of elements of *C* that have p_1 probability of being sampled	14
p_1	sampling probability of the np_1 elements	0.034
np_2	nr. of elements of *C* that have p_2 probability of being sampled	16
p_2	sampling probability of the np_2 elements	0.015
nq_1	nr. of elements of *Cn* that have q_1 probability of being sampled	10
q_1	sampling probability of the nq_1 elements	0.005
nq_2	nr. of elements of *C* that have q_2 probability of being sampled	19
q_2	sampling probability of the nq_2 elements	0.008
Values of *p(a1)* and *p(a2)*		
p(a1) *p(a2)*	true and illusory agreement probabilities	0.7987 0.7207

Table 4 shows the settings of GA1 and GA2 for calibrating the ABM, i.e. finding values for some of the parameters of the ABM that will produce a *p(a1)* and *p(a2)* similar to the ones already calculated using the real data shown in Table 3. The settings for those parameters were established by a trial and error process and by knowing the approximate dynamics of the output time series. From that table we can see that the GAs must find the value of several parameters, using a single value for two variables of the real system, i.e. *p(a1)* and *p(a2)*. Since the ABM produces a time series of those probabilities, one needs to set the time at which the real probabilities will be compared with the ABM-generated ones. In this case, the ABM dynamics shows that at 510 simulation steps, the ABM converges to a stable value for *p(a1)* and *p(a2)*. Table 5 presents the match between the real values of *p(a1)* and *p(a2)* and the ones calculated by the calibrated ABM using GA1 and GA2, both of which were run 10 times.

Table 4. Settings of GA1 and GA2 for calibrating the simple dynamics ABM

Settings of the GA1		
Parameter	**Description**	**Value**
k	nr. of time series of ABM and real system	2
y_t^r	time series of the real system *p(a1)* y_1: (510, 0.7987) *p(a2)* y_2: (510, 0.7207)	
pop-size	population size	30
p_c	cross-over probability	0.7
p_m	mutation probability	0.1
max-iter	maximum nr. of iterations of the GA	50
stop-fitness	fitness that when reached will stop iteration of the GA	0.95
L_p, U_p (the limits for the input parameters that the GA must adjust)	lower and upper limits for the 10 adjustable parameters of the ABM: s_1 : [30, 50] s_2 : [30, 50] np_1 : [5,20] p_1 : [0.01,0.1] np_2 : [5,20] p_2 : [0.01,0.1] nq_1 : [1,15] q_1 : [0,0.01] nq_2 : [5,20] q_2 : [0,0.01]	
Settings of the GA2: the same as those for GA1 and also:		
β	number of significant digits of values of genes and fitness	3
γ_L, γ_U	lower/upper limit of maximum allowable percentage difference between ABM and real system time series	10, 30
α	discount factor to calculate γ	0.2

From the results of Table 5, we can see that the match reached by the parameter values found by both GAs is very good, with a small error in *p(a1)* and *p(a2)*. It must be noted that GA1 and GA2 found the same best solution with a fitness of 0.96. Other solutions are not presented, although the GAs deliver the ten best solutions, so that the domain expert may choose the most appropriate one from a substantive point of view. Thus, we can see that both GAs achieve a similar calibration of the ABM.

To evaluate the relative performance of GA1 to GA2, the number of iterations and time to reach the stopping condition was recorded. The GAs were implemented in Netlogo v. 4.0.4 [19] and

run on a HP PC with Intel Core i5-2500S CPU @ 2.70GHz, 3.2 GB RAM, running under OS MS Windows 7 Enterprise, v.6.1.7601, SP 1. Table 6 presents the average and standard deviation of those figures. To fully analyze the benefits of the refinements made to GA1, we used 4 experimental treatments: GA1 (original GA), GA2-1 (GA1 with the elimination of the very similar solutions), GA2-2 (GA1 stopping the calculation of fitness for the "bad" solutions), GA2-3 (GA1 with both improvements).

Table 5. Results of the calibration of the simple dynamics ABM

Parameter	Calibrated value	Real value
s_1	47	47
s_2	44	44
np_1	12	14
p_1	0.022	0.034
np_2	12	16
p_2	0.029	0.015
nq_1	12	10
q_1	0.007	0.005
nq_2	18	19
q_2	0.007	0.008
Values for *p(a1)* and *p(a2)* (percentage error)		
Best solution's fitness: 0.96		
p(a1)	0.7925 (error = -0.8 %)	0.7987
p(a2)	0.7485 (error = 3.9 %)	0.7207

Table 6. Number of iterations and time to reach stopping condition of GA1 and GA2 for the simple dynamics ABM

	Avg. time to reach stopping condition [s]	Avg. nr. of iterations	Avg. time per iteration [s]
GA1	246.1 (36.7)	7.3 (0.95)	34.4 (7.6)
GA2-1	264.1 (37.0)	8.4 (2.41)	34.1 (12.0)
GA2-2	221.6 (28.7)	7.6 (1.71)	31.0 (9.8)
GA2-3	182.4 (13.3)	6.6 (0.84)	28.2 (5.4)

Note: Standard deviation in parentheses, $N = 10$

It can be seen that stopping the calculation of the fitness for "bad" solutions (GA2-2) shortens the average time to reach the stopping condition and that the shortest time corresponds to experimental treatment GA2-3. The results of a one-way ANOVA with GA-version as its single factor, corroborates that stopping the calculation of fitness for "bad" solutions achieves a statistically significant reduction in that time (p-value = 0.000). However, the elimination of very similar solutions (GA2-1) does not significantly shorten the time (p-value = 0.29). The same happens with the average time per iteration (corresponding p-values = 0.049 and 0.942 respectively). Regarding the average number of iterations performed by the GAs, the ANOVA shows that there are no statistically significant differences among treatments (all p-values above 0.10). Since GA2-2 and GA2-3 immediately abort the calculation of fitness of very "bad" solutions, that decreases the time of each iteration, even if those two versions of the GAs give similar number of iterations. On the other hand, the elimination of similar solutions does not shorten the processing time of the GAs, since the calculation of fitness for each chromosome takes a relatively short time, given the simple dynamics of the ABM.

4.2 Calibration of a complex dynamics ABM1

This experiment uses the ABM that traces the time series of the strength of different versions of a concept. Note that there are no empirical data available to calibrate this ABM, so we calibrated synthetic data (from here on we will refer to this series as "real synthetic"). To use a more complex dynamics time series than before, we set up the ABM1 according to the values of the parameters shown in Table 7.

Table 7. Settings and outputs of the complex dynamics ABM1

Parameter	Description	Value
nr_ver_concepts	nr. of different versions of a concept	2
nr_agents	nr. of agents that compose the group	30
c_0	initial strength of the concepts	0.5
p(a1)	probability of true agreement among agents	0.3
p(a2)	probability of illusory agreement among agents	0.5
Outputs of the ABM1		
c_1^{50} c_1^{300}	strength of version 1 of the concept at $t = 50$ and 300	0.4387 0.0173
c_2^{50} c_2^{300}	strength of version 2 of the concept at $t = 50$ and 300	0.3880 0.0280

Under these settings, the two versions of the concept decrease their strength (c_1 and c_2), reaching a relatively small and stable value in the long run. However, the dynamics that c_1 and c_2 follow is more variable than that which was described in subsection 4.1 (for a detailed discussion of the dynamics see [22] and [23]). Thus, we would expect that under this situation, GA2 would perform significantly better than GA1.

Table 8 shows the settings of GA1 and GA2 for calibrating the ABM1, which indicates that there are two input parameters to the ABM1 that the GA must adjust, namely *p(a1)* and *p(a2)*, to obtain the match of two output time series (c_1 and c_2). As before, the settings of the parameters of the GAs were established by trial

and error, based on the knowledge of the type of dynamics that the output time series exhibit.

Table 8. Settings of GA1 and GA2 for calibrating the complex dynamics ABM1

Settings of the GA1		
Parameter	**Description**	**Value**
k	nr. of time series of ABM and real system	2
$\mathbf{y}_t^r$	time series of the real system $c_1 = y_1$: (50, 0.4387), (300, 0.0173) $c_2 - y_2$: (50, 0.388), (300, 0.028)	
pop-size	population size	100
p_c	cross-over probability	0.7
p_m	mutation probability	0.1
max-iter	maximum nr. of iterations of the GA	50
stop-fitness	fitness that when reached will stop iteration of the GA	0.75
L_p, U_p (the limits for the input parameters that the GA must adjust)	lower and upper limits for the 2 adjustable parameters of the ABM: *p(a1)* : [0, 1] *p(a2)* : [0, 1]	
Settings of the GA2: the same as those for GA1 and also:		
β	number of significant digits of values of genes and fitness	3
γ_t, γ_U	lower/upper limit of maximum allowable percentage difference between ABM and real system time series	60, 30
α	discount factor to calculate γ	0.05

Table 9 presents the match between the real synthetic values of *p(a1)* and *p(a2)* and the ones calculated by the calibrated ABM1 using GA1 and GA2, both of which were run 10 times. The corresponding RMSE show that the match between the real synthetic and calibrated time series is good. Note that even slight differences in *p(a1)* and *p(a2)* produce different ABM outputs. That is one of the problems when modeling CAS: they exhibit non-deterministic outputs that are very sensitive to initial conditions, as already explained in the introduction. Because the dynamics of this ABM1 is more complex than the previous one, Figure 1 presents the graphs of the time series, so that one can visually assess the fit.

The graphs on Figure 1 confirm that the match of c_2 is better than that of c_1, as the corresponding RMSE already suggested, and that both time series achieve relational equivalence. As done before, Table 10 shows the average and standard deviation of the number of iterations and time to reach the stopping condition. Those figures were calculated for the same treatments and using the same hardware and software specifications as in the previous experiments.

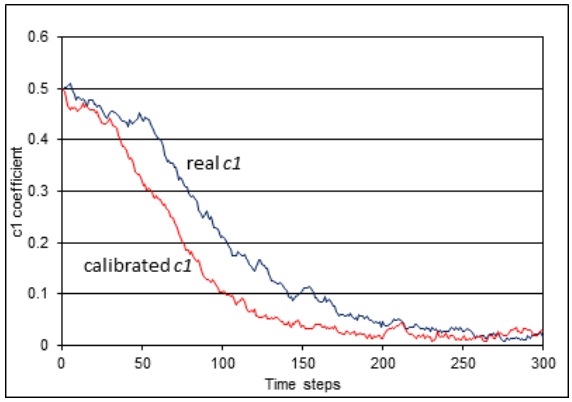
Table 9. Results of the calibration of the complex dynamics ABM1

Parameter	**Calibrated value**	**Real Synthetic value**
p(a1)	0.298	0.3
p(a2)	0.484	0.5
RMSE of c_1 and c_2		
Best solution's fitness: 0.77		
RMSE c_1	300 data points	0.0645
RMSE c_2	300 data points	0.0243

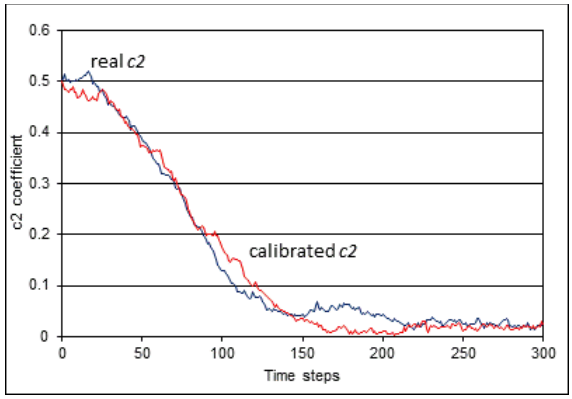
Table 10. Number of iterations and time to reach stopping condition of GA1 and GA2 for the complex dynamics ABM1

	Avg. time to reach stopping condition [s]	**Avg. nr. of iterations**	**Avg. time per iteration [s]**
GA1	711.8 (489.7)	26.5 (19.0)	27.7 (1.5)
GA2-1	287.1 (141.4)	12.8 (7.4)	23.5 (2.5)
GA2-2	360.7 (159.3)	16.3 (6.8)	21.7 (1.7)
GA2-3	69.9 (38.2)	4.3 (3.0)	18.7 (5.9)

Note: Standard deviation in parentheses, $N = 10$



(a)



(b)

Figure 1. Real and calibrated time series of complex ABM1: (a) c1 series (b) c2 series

It can be seen that stopping the calculation of the fitness for "bad" solutions and eliminating the similar solutions shorten the average time to reach the stopping condition and decreases the average number of iterations and average time to execute iterations. The best (smallest) of those figures correspond to experimental

treatment GA2-3, which combines the two improvements. The results of the ANOVA show that those differences are statistically significant at least at the 0.01 level. Since this ABM1 exhibits a rather complex dynamics, the GA may find more "bad" solutions than in a simple dynamics ABM, and thus the immediate stopping of the calculation of fitness for them is helpful. Also, in this ABM1, the calculation of the ABM1's outputs is more time-consuming, and thus the stopping mechanism saves more time. The same can be said regarding the elimination of similar solutions. Each similar solution that is eliminated saves significant computational time and also allows GA2-1, GA2-2 and GA2-3 to speed up convergence by reducing the population size.

4.3 Calibration of a complex dynamics ABM2

For this experiment we used the same ABM1, but changed some settings as shown in Table 11, and labeled it ABM2. The most important change corresponds to the new values of *p(a1)* and *p(a2)*. As discussed in [22] and [23], those new values produce a complex dynamics in the outputs of the ABM2, which is characterized by the strengthening of the concepts' coefficients.

Table 11. Settings and outputs of the complex dynamics ABM2

Parameter	Description	Value
nr_ver_concepts	nr. of different versions of a concept	5
nr_agents	nr. of agents that compose the group	30
c_0	initial strength of the concepts	0.5
p(a1)	probability of true agreement among agents	0.8
p(a2)	probability of illusory agreement among agents	0.3
Outputs of the ABM2		
c_1^{50} c_1^{300}	strength of version 1 of the concept at t = 50 and 300	0.8133 0.9933
c_2^{50} c_2^{300}	strength of version 2 of the concept at t = 50 and 300	0.7700 1.0000
c_3^{50} c_3^{300}	strength of version 3 of the concept at t = 50 and 300	0.6867 0.9967
c_4^{50} c_4^{300}	strength of version 4 of the concept at t = 50 and 300	0.7733 1.0000
c_5^{50} c_5^{300}	strength of version 5 of the concept at t = 50 and 300	0.8133 0.9967

Table 12 shows the settings of GA1 and GA2 for calibrating this ABM2, which indicates that there are two input parameters to the ABM2 that the GA must adjust *p(a1)* and *p(a2)*, to obtain the match of five output time series (c_1 to c_5).

We must note that in Table 12, the parameters γ_L, γ_U and α were not set. That means that the stopping mechanism for the calculation of fitness of "bad" solutions was disabled. We did so given that in the process of determining the settings for the parameters, we found that GA2 performance was very sensitive to the values of γ_L and γ_U. A value of γ_L and γ_U around 100 and 120, allowed a good performance of GA2, but slightly lower values, yielded a fitness for all the solutions equal to 10^{-4}, i.e. the mechanism aborted the calculation of fitness for all the solutions. Because of that, GA2 was not able to converge. On the other hand, and as expected, higher values for γ_L and γ_U did not reduce the time to reach convergence. Given that situation, we preferred to disable the mechanism, so that convergence was assured. We will further discuss this issue in the conclusions.

Table 12. Settings of GA1 and GA2 for calibrating the complex dynamics ABM2

Settings of the GA1		
Parameter	**Description**	**Value**
k	nr. of time series of ABM and real system	2
y_t^r	time series of the real system: see outputs c_1 to c_5 of ABM2 in Table 11	
pop-size	population size	100
p_c	cross-over probability	0.7
p_m	mutation probability	0.2
max-iter	maximum nr. of iterations of the GA	50
stop-fitness	fitness that when reached will stop iteration of the GA	0.197
L_p, U_p (the limits for the input parameters that the GA must adjust)	lower and upper limits for the 2 adjustable parameters of the ABM: *p(a1)* : [0, 1] *p(a2)* : [0, 1]	
Settings of the GA2: the same as those for GA1 and also:		
β	number of significant digits of values of genes and fitness	2
γ_L, γ_U	lower/upper limit of maximum allowable percentage difference between ABM and real system time series	none
α	discount factor to calculate γ_t	none

Table 13 presents the match between the real synthetic values of *p(a1)* and *p(a2)* and the ones calculated by the calibrated ABM2 using GA1 and GA2, both of which were run 10 times. The corresponding RMSE show that the match between the real synthetic and calibrated time series is good. Note that since many random processes occur in the ABM2 and its dynamics is complex, even having an exact match in *p(a1)* and *p(a2)* produces different outputs of the ABM. That is one of the problems with the modeling of CAS: they exhibit non-deterministic outputs, as already explained in the introduction.

Table 13. Results of the calibration of the complex dynamics ABM2

Parameter	Calibrated value	Real Synthetic value
p(a1)	0.8	0.8
p(a2)	0.3	0.3
RMSE of c_1 and c_2		
Best solution's fitness: 0.20		
RMSE c_1	300 data points	0.06869
RMSE c_2	300 data points	0.07288
RMSE c_3	300 data points	0.01231
RMSE c_4	300 data points	0.04279
RMSE c_5	300 data points	0.07443

Figure 2 shows the corresponding graphs of c_1 through c_5, both for the real synthetic and calibrated time series. Visually assessing the match of the time series and comparing them with their corresponding RMSE presented in Table 13, we can see that the best match corresponds to c_3, and the fit of c_4 is better than those of c_1, c_2 and c_5. Most notably, all of the time series achieve relational equivalence.

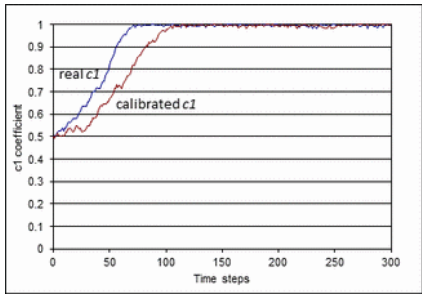
Table 14 shows the average and standard deviation of the number of iterations and time to reach the stopping condition. Those figures were calculated only for GA1 and GA2-1 and using the same hardware and software specifications as in the previous experiments.

Table 14. Number of iterations and time to reach stopping condition of GA1 and GA2 for the complex dynamics ABM2

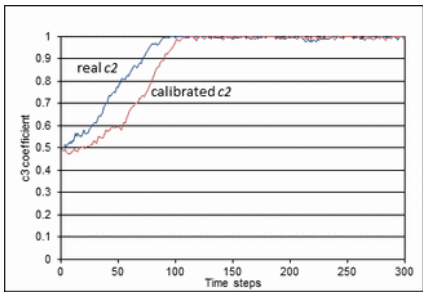
	Avg. time to reach stopping condition [s]	Avg. nr. of iterations	Avg. time per iteration [s]
GA1	738.6 (171.0)	27.2 (6.5)	27.3 (0.58)
GA2-1	489.6 (238.9)	21.4 (11.1)	23.2 (1.67)

Note: Standard deviation in parentheses, $N = 10$

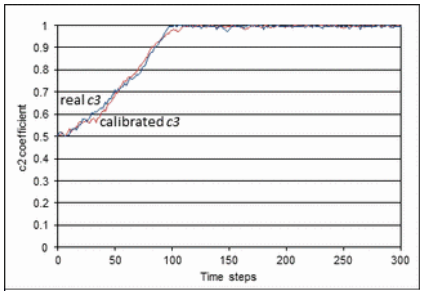
The figures in Table 14 indicate that the elimination of similar solutions significantly shortens the average time to reach the stopping condition (p-value of ANOVA = 0.015). On the other hand, although the average number of iterations is smaller for GA2-1 than for GA1, that difference is not statistically significant (p-value = 0.173). However, the shorter average time per iteration of GA2-1 compared with GA1 is highly significant (p-value = 0.000). All of that means that the shorter time of GA2-1 to reach the stopping condition is mainly due to the decrease in time per iteration. Since the elimination mechanism reduces the number of solutions whose fitness must be evaluated per iteration of GA2-1 that accounts for the highly significant decrease in the time necessary to perform iterations.



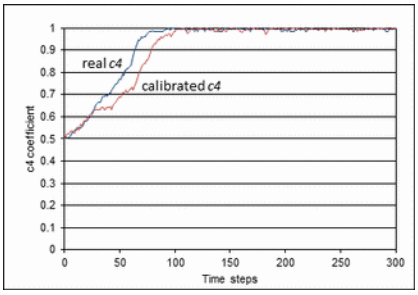
(a) c1 series



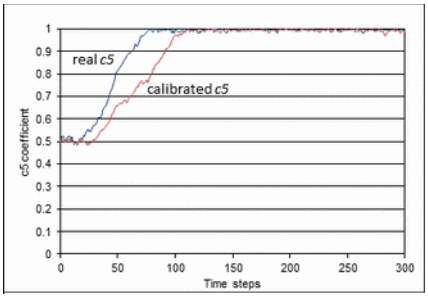
(b) c2 series



(c) c3 series



(d) c4 series



(e) c5 series

Figure 2. Real and calibrated time series of complex ABM2

5. Conclusions

The results indicate that the proposed GA delivers solutions that achieve a sufficiently good match between the ABMs' outputs and systems' time series (empirical and synthetic), reaching relational equivalence. Note that by setting the parameters that control the two mechanisms of GA2 designed to shorten computational time (i.e. β, □L, □U and α) the researcher can reach a balance between achieving an adequately short computational time and good relational equivalence. In general, a low β, □L, □U and high α will accomplish a shorter computational time and a relatively worse relational equivalence than setting a high β, □L, □U and low α; and vice-versa. Furthermore, the GA accomplishes relational equivalence even for ABMs that exhibit a relatively complex output dynamics. Additionally, the GA is able to calibrate the ABMs using a few data points of the input values, which is important for reducing the computational time of the process, and also because in social science research, longitudinal studies commonly use only two points in time of the variables of interest. We must point out that the GA delivers the ten best fitness solutions, although one could easily change the code so that the GA provides more than ten. This allows the researcher and domain expert to choose the solution that achieves relational equivalence and also makes sense from a substantive point of view. Thus, the researcher might select a solution that does not have the best fitness, but that better suits the explanation of the phenomenon under study.

The results also demonstrate that the two refinements made to the original GA help in reducing the computational time for reaching convergence of the GA. These improvements are especially noticeable for ABMs with a complex dynamics. However, care must be taken when setting up the values of the parameters that control the functioning of the corresponding mechanisms, especially in the case of □L, □U and α. If one establishes too small values for □L and □U or too high a value for α, that might hinder the GA's convergence. In this study, we established the parameters of the GAs by trial and error, based on our knowledge of the dynamics of the ABMs, so that we could avoid that problem, particularly in the more complex dynamics ABMs. However, ABMs that model CAS may present too complex dynamics to be able to correctly set up those parameters. Thus, in future work we will refine the mechanism that stops the calculation of fitness for "bad" solutions, so that □t (see line 17 of the pseudo code of GA2 in Table 2) is automatically adjusted. That might be done by monitoring the fitness of all the solutions and if all them decrease to a very small value (10-4, see line 4 c.i. of the pseudo code presented in Table 2), □t may be increased until the fitness of solutions augments and then simply keep that value of □t for the rest of the iterations of the GA2. That will assure the convergence of GA2.

Another improvement to GA2 may involve enhancing its stopping criteria. Presently, GA2 stops iterating when a maximum number of iterations is reached or when an established fitness is achieved (see line 3 of the pseudo code in Table 2). However, during the experiments with the GAs we noticed that sometimes the fitness of the best solutions remained unchanged during a large number of iterations, especially for complex dynamics ABMs. Thus, one could add a stopping criterion that computes the difference in fitness between the current iteration's solutions and the ones of a user-specified previous iteration (using a sliding window approach), and if that difference is too negative or does not exceed a certain threshold, then the GA2 stops iterating.

Acknowledgments

This work was supported by FONDECYT (Fondo Nacional de Ciencia y Tecnologia of the Chilean Government) grant Nr. 1150074 to both authors.

REFERENCES

1. CANESSA, E., R. RIOLO, **An Agent-based Model of the Impact of Computer-mediated Communication on Organizational Culture and Performance: An Example of the Application of Complex Systems Analysis Tools to the Study of CIS**, Journal of Information Technology, vol. 21, 2006, pp. 272-283.
2. QUEZADA, A., E. CANESSA, **Agent-based Modeling: A Tool for Complementing the Analysis of Social Phenomena**, Avances en Psicología Latinoamericana, vol. 28 (2), 2010, pp. 226-238.

3. MILLER, J., PAGE, S., **Complex Adaptive, Systems: An Introduction to Computational Models of Social Life**, Princeton: Princeton University Press, NJ, USA, 2007.

4. HOLLAND, J. H., **Hidden Order: How adaptation builds complexity**, Addison-Wesley, Redwood City, 1995.

5. BANKES, S. C., **Agent-based modeling: A revolution?** PNAS 99, vol. 3, 2002, pp. 7199-7200.

6. BONABEAU, E., **Agent-based Modeling: Methods and Techniques for Simulating Human Systems**, PNAS 99, vol. 3, 2002, pp. 7280-7287.

7. CONTE, R., R. HEGSELMANN, P. TERNA, **Simulating Social Phenomena**, Berlin: Springer-Verlag, 1997.

8. BANKES, S. C., **Exploratory Modeling for Policy Analysis**, Operations Research, vol. 41(3), 1993, pp. 435-449.

9. GRIMM, V., S. F. RAILSBACK, **Individual-based Modeling in Ecology**, Princeton: Princeton Univ. Press, 2005.

10. GRIMM, V., E. REVILLA, U. BERGER, F. JELTSCH, W. M. MOOIJ, S. F. RAILSBACK, H.-H. THULKE, J. WEINER, T. WIEGAND, D. L. DEANGELIS, **Pattern-oriented Modeling of Agent-based Complex Systems: Lessons from Ecology**, Science, vol. 310 (5750), 2005, pp. 987-991.

11. AXELROD, P., **Advancing the Art of Simulation in the Social Sciences**, in: R. Conte, R. Hegselmann, P. Terna eds., Lecture Notes in Economics and Mathematical Systems: Simulating Social Phenomena, Berlin: Springer-Verlag, 1997.

12. HEPPENSTALL, A. J., A. J. EVANS, M. H. BIRKIN, **Genetic Algorithm Optimisation of an Agent-based Model for Simulating a Retail Market**, Environment and Planning B: Planning and Design, vol. 34, 2007, pp. 1051-1070.

13. CAPORALE, G. M., A. SERGUIEVA, H. WU, **Financial Contagion: Evolutionary Optimization of a Multinational Agent-based Model**, Intl. J. of Intelligent Systems in Accounting and Finance Management, vol. 16(1-2), 2009, pp. 111-125.

14. ROGERS, A., P. VON TESSIN, **Multi-objective Calibration for Agent-based Models**, Proc. of 5th Workshop on Agent-based Simulation, May 2004, pp. 17-22.

15. SKOLICKI, Z., T. ARCISZEWSKI, M. HOUCK, K. D. JONG, **Co-evolution of Terrorist and Security Scenarios for Water Distribution Systems**, Advances in Engineering Software, vol. 39(10), 2008, pp. 801-811.

16. NARZISI, G., V. MYSORE, R. MISHRA, **Multi-objective Evolutionary Optimization of Agent-based Models: an Application to Emergency Response Planning**, Proc. the 2nd IASTED International Conference on Computational Intelligence, 2006.

17. CALVEZ, B., G. HUTZLER, **Automatic Tuning of Agent-Based Models Using Genetic Algorithms**, Proc of 6th Intl. Workshop on Multi-Agent Based Simulation (MABS'05), Netherlands, 2005.

18. ZWIJZE-KONING, K. H., M. D. T. DE JONG, **Auditing Information Structures in Organizations: A Review of Data Collection Techniques for Network Analysis**, Organizational Research Methods, vol. 8 (4), 2005, pp. 429-453.

19. WILENSKY, U., **NetLogo**. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.

20. MACAL, C. M., M. J. NORTH, **Agent-based Modeling and Simulation**, Proc. Winter Simulation Conference 2009, M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds., Austin, TX, Dec. 2009, pp. 86-98.

21. CHAIGNEAU, S., E. CANESSA, J. GAETE, **Conceptual Agreement Theory**, New Ideas in Psychology, vol. 30, 2012, pp. 179-189.

22. CHAIGNEAU, S., E. CANESSA, A. QUEZADA, **The Spreading and Demise of Concepts in Social Groups**, Proc. IEEE XXIX Intl. Conf. of the Chilean Computer Science Society (SCCC 2010), IEEE Computer Society, Mar. 2011, pp. 139-145.

23. CHAIGNEAU, S., E. CANESSA, **The Power of Collective Action: How Agents Get Rid of Useless Concepts without Even Noticing Their Futility**, Actas XXX Intl. Conf. of the Chilean Computer Science Society, Curicó, Chile, 7-11 Noviembre, 2011.