Finding optimal agent-based models

Center on Social and Economic Dynamics Working Paper No. 49

Ben Klemens

September 2007

This paper applies standard maximum likelihood (ML) techniques to find an optimal agent-based model (ABM), where *optimal* could refer to replicating a pattern or matching observed data. Because ML techniques produce a covariance matrix for the parameter estimates, the method here provides a means of determining to which parameters and conditions the ABM is sensitive, and which have limited effect on the outcome. Because the search method and the space of models searched is explicitly specified, the derivation of the final ABM is transparent and replicable. Hypotheses regarding parameters can be tested using standard likelihood ratio methods.

Background

More computational firepower has allowed two types of research to flourish. The first is the agent-based model (ABM), which keeps track of the decision making behavior of thousands or millions of agents. The second is optimization as a statistical method, typically involving resampling a likelihood function hundreds or thousands of times.

This paper demonstrates a method of combining the two streams of computation, by searching for the optimal ABM within the space of ABMs. Think of any statistical model as a black box, that takes a set of parameters $\boldsymbol{\beta}$ as inputs, and produces a probability value P as an output. The model could thus be written as a function $P(\boldsymbol{\beta})$, which expresses the fact that every input parameter $\boldsymbol{\beta}$ outputs a likelihood.[1] There are a number of methods for searching the space of $\boldsymbol{\beta}$ to find the value that maximizes $P(\boldsymbol{\beta})$, some of which will be discussed below.

But the internals of the $P(\cdot)$ function could be an agent-based model as easily as they could be a traditional statistical model. In such a case, the parameters $\boldsymbol{\beta}$ include both what are typically called model parameters (size of the board, for how many periods the simulation runs) and the discrete rules of the model like whether agents can move or are stationary. Save for some technical details to follow, these can be searched in a manner similar to the search for the optimal parameters in standard statistical models like the logit or probit.

But there are a number of considerations that must be addressed before maximum likelihood techniques can be applied to agent-based models. Some are engineering questions about getting good-quality estimates, and some are conceptual issues about maximizing a stochastic function whose inputs and outputs may be discontinuous.[2]

This paper develops a form for the meta-model that smooths out the details, and then gives examples of optimization searches over two disparate metamodels.

Meta-models

Let a *meta-model* be a set of rules and parameters. The meta-parameters of the meta model may include both traditional parameters (which are typically real numbers like payoffs), and characteristics which are

[1] Some authors distinguish a *probability* (odds of data X given a parameter β) from a *likelihood* (odds of β given X). I use them interchangeably here, because they are both derived from one joint distribution, $P(X, \beta)$.

[2] Computing power is *not* a concern. Optimizing a small agent-based model, like the simulation below that runs on a 30×30 grid for about fifty periods, actually requires less computation than optimizing some standard statistical models using large data sets.

typically considered to be fixed rules of the game, chosen from discrete and unordered sets like eight-way or four-way movement on a grid, or whether players will engage in a Prisoner's Dilemma or a Coordination game.

This paper tests a generic search algorithm to search for the meta-parameters of a meta-model that best fit a given criterion. That is, it searches a space of models for the parametrized model that best fits the criterion. If the criterion is that the model best fit observed real-world data, then this process is known as model validation.

The generic search algorithm borrows from the many comparable routines in the literature that find the optimum of a function. In this case, the researcher would first specify a goal for the simulation, such as maximizing the ratio of doves to hawks in the simulation below, and then specify bounds in which the algorithm will search for rules. The algorithm would then find the rules of the model that maximize the expected output.

This setup differs from the long history of searching for the best agent strategy within a fixed framework, such as the tournament of Prisoner's Dilemma strategies run by Axelrod and Hamilton [1981] and the many offspring tournaments since run. Such tournaments, evolutionary frameworks, and genetic algorithms aim to optimize a certain agent's goal (which is invariably a variant on maximizing profit or number of offspring), whereas the goal here regards the overall shape of the society, such as maximizing the balance of Hawks and Doves or minimizing the number of clashes between the two groups. Thus, the key variables here are meta-parameters of the model (which can include methods by which agents select strategies), rather than the agent strategies themselves.

There are a number of benefits to the automated optimization presented here. The first is the simple benefit of automation, such as the time-savings and precision a researcher could reap by having a computer fine-tune the model's specification and parameters. But beyond that, automation affords transparency, reducing the sense that the parameters are simply an *ad hoc* set of assumptions. Other options that the output model could have used are explicitly listed, so there is no doubt in the reader's mind about the alternatives.

Leombruni et al. [2006] suggests a standard protocol for testing an ABM, with an eye toward better transparency, that includes the recommendation that a "full exploration" of all parameters should be run. With six continuous parameters, each sliced into a grid of a hundred possibilities, a full exploration would cover a billion possible combinations. If the model takes one second to run, the full grid search would take 3,170 years. The simulated annealing method below provides a standard method of covering the space that finds the areas most likely to affect the outcome, that typically runs in a few minutes or hours.

Further, the maximum likelihood estimation (MLE) literature has developed a number of statistical techniques associated with MLEs, which can be beneficial to an ABM.

- One can develop a confidence interval around the parameters of the model, using the information matrix and the Cramér-Rao inequality. This shows to which parameters the model is sensitive, and which parameters do not significantly affect the model outcome.
- The same confidence intervals can be used for standard t or F tests to determine whether a parameter has a true influence on the outcome of the model or is merely a 'nuisance parameter'.
- The likelihood ratio test allows one to test hypotheses regarding whether a given parameter provides a statistically significant improvement over an alternative.

This paper applies such techniques directly to the parameters of agent-based models.

The space of inputs

The meta-model consists of selecting a vector of parameters $\boldsymbol{\mu}$, and then using those parameters to select a model with parameters $\boldsymbol{\beta}$, which will then map to outcome V. One could think of this as a simple model with outcome $V(\boldsymbol{\mu})$, but this paper will take the equivalent approach of assuming a meta-model and a model proper, and thus a two-step mapping $\boldsymbol{\mu} \to \boldsymbol{\beta} \to V$.

An optimization requires a goal to optimize; the examples below will use such goals as the number of hawks adjacent to doves or the number of contiguous groups of hawks. Below, the search will be done using simulated annealing, and that optimization method places certain regularity conditions on the space from which $\boldsymbol{\beta}$ is drawn. There must be a metric on the parameter space, and it is useful if the space has a topology such that any $\boldsymbol{\beta}$ is surrounded by a neighborhood of arbitrarily close alternatives.

Continuous parameters $\in \mathbb{R}$ have their standard topology and Euclidean metric, and so the meta-model can search over them without modification. For integer parameters (such as the number of periods or size of the board) the optimization can search over $\mu \in \mathbb{R}$, and the first step of the simulation would round the input to $\beta = \lfloor \mu \rfloor$ (meaning that the output V may have a discontinuity at the jump from $\mu = i - \epsilon$ to $\mu = i$, which is not a problem).

Only the discrete, unordered parameters remain. For example, in the simulation below, agents could be able to move in four directions (North, South, East, or West), or eight directions (those directions plus NE, NW, SE, and SW).[3] Let the movement parameter be β_M, and the options be $\beta_M = F$ for four-directional movement or $\beta_M = E$ for eight-directional movement.

The obvious method of generating a continuous scale over the two discrete modes of movement would be to have agents play a mixed strategy in which $\beta_M = F$ with probability p and $\beta_M = E$ with probability $1-p$. But this is unsatisfactory, because a mixture could produce a higher payoff than either purely F or E, or there could be a sharp discontinuity in outcomes between $p = \epsilon$ and $p = 0$. In short, the mixed outcomes may say nothing about the pure outcomes.

The solution is to repeat the model using pure rules fixed before the model is played. Over a hundred models, $100p$ of them will be played with a pure rule of $\beta_M = E$ and $100(1-p)$ will be played with a pure rule of $\beta_M \equiv F$. Let $E(V|\beta_M = E) \equiv K_E$ and $E(V|\beta_M = F) = K_F$; then the expected outcome given p will be $pK_E + (1-p)K_F$, and so there is a simple linear relation between the input parameter p and the output parameter $E(V)$.

If the discrete parameter could take on n values, then a choice among them would require $n-1$ meta-parameters $\mu_1, \dots \mu_{n-1}$, all $\in [0, 1]$, such that $0 \leq \sum_{i=1}^{n-1} \mu_i \leq 1$. At the outset of each model, draw a random number r from a uniform distribution and select option one if $r \leq \mu_1$, option two if $r \leq \mu_1 + \mu_2$, $\dots$, and option n if $r > \sum_{i=1}^{n-1} \mu_i$.

Mixing over pure elements in the metagame is a separate process from a game with mixed elements. The models in this paper will use only fixed elements, but one could have a series of possible parameters

- four-way movement,
- eight-way movement, and
- four way movement with probability p, eight way with probability $1-p$,

and then do a search over these three possible parameters as above.

The interpretation of derivatives of discrete choices only makes sense as the change in V from a shift from one option to another. Here, I will take the derivative with respect to option j to be the expected difference in outcome given a switch from the (arbitrarily chosen) first option to option j.[4]

To summarize, the parameters of the meta-model will correspond to the parameters of the model, but their relation will depend on the type of the parameter:

[3] Four-directional movement is sometimes called the *von Neumann* topology and eight-directional movement the *Moore* topology.

[4] In the case of a continuous derivative, the alternative is basically implicit, but one can interpret a derivative as the difference in outcome given a switch from $\beta_i = x$ to $\beta_i = x + \epsilon$.

- Continuous parameters are their own meta-parameters: $\beta_i = \mu_i$.
- Integral parameters have continuous meta-parameters, which are simply rounded or truncated to produce the parameters: $\beta_i = \lfloor \mu_i \rfloor$. Other discrete, ordered parameters can be handled similarly.
- Unordered discrete parameters that could take on n values have a set of $n - 1$ meta-parameters, each in $[0, 1]$, such that the meta-parameters sum to a value in $[0, 1]$.

Searching the space

The process described in this paper is simply an optimization, and one could use any optimization method to execute the search for the optimum, including genetic algorithms, conjugate gradient methods, simplex methods, hill-climbing, root-finding, and so on. But for a number of reasons, simulated annealing (SA) recommends itself for this situation.

Briefly, simulated annealing is an application of the Metropolis-Hastings algorithm for searching a space. It is a random walk, jumping from state to state, with two simple rules. Randomly select state $t + 1$. If the probability of state $t + 1$ is greater than the probability of state t, then the system jumps to state $t + 1$; if the probability of state $t + 1$ is less than that of state t, then the system still jumps to state $t + 1$ with a probability decreasing in t. Early on, when the probability of jumping to a less-likely state is high, the state can easily jump anywhere in the state space, but as time progresses, the system will only move to more likely states, and so approach the nearest optimum.

This method works well for the given situation for a number of reasons. First, the information matrix is a global property of the model, meaning that one needs to sample parameters from the entire parameter space to calculate it. Ackley [1987] notes that simulated annealing is often slow to converge, which in this case is a good thing, because its many iterations gather much more information for use in calculating $\mathcal{I}$, over a larger range of values, than relatively fast-converging algorithms like conjugate gradient methods. That is, SA provides a search for the optimum and a sampling routine for the information matrix at the same time.

Second, the mapping from models to outcomes is unknown. There may be a simple linear relationship among elements (as there is along the artificial linear combinations in the meta-model above), or there may be a complex series of local maxima and minima. Simulated annealing is well-suited to such a situation, because it has some likelihood of jumping from the neighborhood of any local optimum to that of any other.

Simulated annealing also offers the benefit that it is memoryless: the odds of jumping from state t to state $t+1$ in no way depends on states $t-1$, $t-2$, ... Some optimization methods build up an image of the space based on past draws, which normally improves efficiency. But with stochastic outcomes, it is possible that if some early draws are not representative, then the optimization will be led astray. With SA, an erratic draw may lead to an erroneous jump/non-jump in one step, but that may readily be corrected in later iterations.[5]

Sensitivity and confidence intervals

This paper is concerned not only with finding the optimal model, but also finding the level of confidence that one can place on that optimum, which breaks down into two subquestions.

The first is the traditional hypothesis test. The likelihood ratio test is a natural complement to maximum likelihood estimation, and requires little additional calculation. Let the likelihood of the best set of parameters be L_u, and let the likelihood of the best parameters constrained such that one parameter is fixed at a given value be L_c. Then the statistic $2\ln(L_u) - \ln(L_c)$ has a χ^2_1 distribution, and can be used to produce traditional confidence intervals and p-values.

The second means of evaluating the level of confidence one can place on a model is via sensitivity checks: as a given parameter shifts, how much does it affect the outcome measure? Again, it is easy to use the information gathered during a maximum likelihood search to produce a covariance matrix that answers such questions directly. The bumper-sticker explanation for how this is done is that the variance of a maximum likelihood estimate of a parameter achieves the Cramér-Rao lower bound (CRLB). The details of the CRLB and its underlying technical conditions are presented in the appendix.

Subjective confidence intervals In the case of an agent-based model intended to explain a data set, one could write down a traditional likelihood function, that expresses the probability of observing the output given a distribution assumed to take a certain form (typically a Normal distribution) based on the real-world mean and variance of the output measure.

For a theoretical agent-based model based on a goal that does not correspond to a real-world set of events, such as the number of doves in a hawk-dove game, one needs a subjective probability distribution.

[5]The stochastic nature of $V(\cdot)$ itself can also be mitigated by simply taking the mean of several runs. In the test below, the simulation reports the mean of twenty plays of the model for each $\boldsymbol{\beta}$.

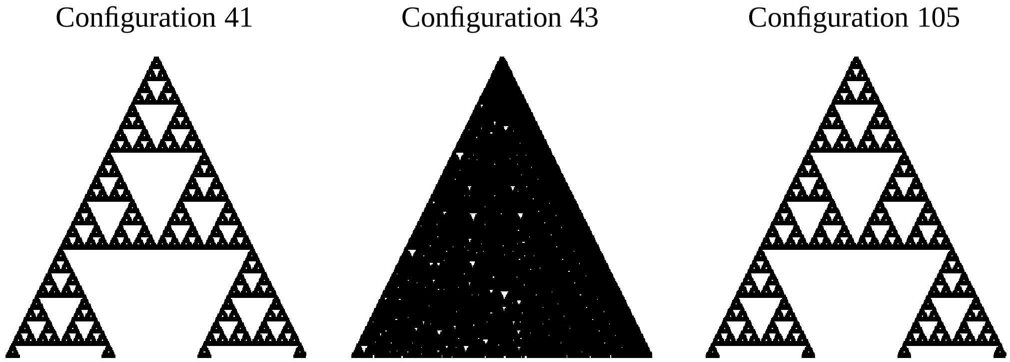

Figure 1: Configurations 41 and 105 both produce the Sierpinski triangle.

Given the outcome $V(\boldsymbol{\beta}) \geq 0$, the natural subjective probability measure is simply

$$P_{\text{subj}}(\boldsymbol{\beta}) \equiv \frac{V(\boldsymbol{\beta})}{\int_{\forall \boldsymbol{\beta}} V(\boldsymbol{\beta})}.$$

It is essential for the proofs regarding the CRLB that P integrate to a constant (i.e., one), but for most calculations it is sufficient to state that $P_{\text{subj}}(\boldsymbol{\beta}) \propto V(\boldsymbol{\beta})$.

The appendix provides a few additional technical details regarding the calculation of the information matrix using a subjective probability distribution.

An example: The Sierpinski triangle

Wolfram [2002] describes how the progress of a one-dimensional cellular automaton can describe a Sierpinski triangle, as pictured in Figure 1.

Define the parents of a point to be the three points above the given point (to its Northwest, North, and Northeast). There are eight possible configurations for the parent: (off, off, off), (off, off, on), (off, on, off), (off, on, on), ... These can be read as the binary numbers from zero to seven, by writing, say, (off, on, off) as (0, 1, 0), or more compactly, 010.

A rule in this system consists of an outcome for each of the eight parent configurations: either the child point is on or off. A rule can thus be summarized as a sequence of eight ones or zeros—another binary number, from zero to 256. For example, rule 13=0001101 specifies that if the parent configuration is 1=001, 3=011, or 4=100, then the child cell is on; else the child cell is off.[6]

[6] As an arbitrary point of æsthetics, the parent configuration of 0=(0, 0, 0) always produces no child. This ensures that the space outside the triangle remains blank.

The process of generating a configuration begins with a grid that is blank save for a single cell in the center of the top line. The second line is generated by checking each cell's parents on the first line and determining whether the rule dictates that the child on the second line should be on or off. Once the second line is generated, the third line is generated in the same way, continuing to the end of the grid.

The goal is the Sierpinski triangle, which has three key features. The sides of the triangle are generated by cells whose parents are either 1=001 on the left side of the triangle, or 4=100 on the right side. The straight horizontal lines followed below by blank spaces are a clear feature of the figure, but it is hard to tell from the low-resolution diagram above whether the triangle is a solid line or a dotted line of alternately on and off elements. If it were a straight like, then the hollow triangles would be created by the rule that if all three parent cells are on, 7=111, then the child is off; if it were a dotted line, then the rules 5=101 and 2=010 would need to be off. Configuration 41=0101001 fits all of these characteristics, and Figure 1 shows that configuration 41 does indeed produce a Sierpinski triangle.

Comparative statics The search for an optimum is not very interesting in this case. Given training information of configuration 9=0001001, each element is compared pixel-by-pixel to the output produced by rule 9, and gets a point for each pixel that matches. Ranking the configurations by score easily finds those rules that produce the Sierpinski triangle (configurations 9, 13, 41, 45, 73, 77, 105, 109; rule 11 is a near-miss that bears a resemblance but is not quite correct).

It is more interesting to ask about the comparative statics. If a mutation appeared that switched one bit in a rule that produces a Sierpinski triangle, would the mutated rule continue to produce the same pattern?

The covariance matrix of the set of games, produced via the information from the subjective likelihoods as described in the appendix, gave the following variances:

rule	**variance**
1: (0, 0, 1)	4.790
2: (0, 1, 0)	3.541
3: (0, 1, 1)	14.402
4: (1, 0, 0)	4.788
5: (1, 0, 1)	15.994
6: (1, 1, 0)	14.403
7: (1, 1, 1)	20.471

A small variance indicates sensitivity, so the rules that are most likely to destroy the output given a mutation are rules 1, 2, and 4. Rules 1=001 and 4=100 were listed as essential above. The comparative statics see what the low-resolution diagram could not show: Wolfram's cellular automata produce a Sierpinski triangle where the tops of the hollow triangles are covered by an alternating on-off pattern rather than a solid line. This is demonstrated by the fact that rule 7 is almost irrelevant to the output score, while it is very important that rule 2=010 not be on. Figure 1 shows configuration 41, then configuration 41 plus rule 2 (equals configuration 43), and finally configuration 41 plus rule 7 (equals 105). When rule 2 is added, there is a drastic change in output, while when rule 7 is added, the output is pixel-for-pixel identical.[7] When rule 5 is added, the output is generally a checkerboard pattern, much like the Sierpinski triangle with the holes filled in; as such the digression is less than when rule 2 is added, which produces sharp vertical lines.

Thus, the variances, readily derived from the subjective likelihood calculations, readily demonstrate in what ways the system is sensitive to mutation.

An example: Agent groupings

This section presents a more complex example that is somewhat less predictable than the last, which hopes to match a number of rules of the game with a number of outcome metrics.

The set of rules The basic form of the model is a set of agents on a grid of spaces, who are born, move around, die, and play some type of exchange game amongst themselves.

The choices are culled from a variety of sources, including the demographic prisoner's dilemma (as in Epstein [1998]), the Hawk-Dove game (as in Gintis [2000]), and the familiar coordination game.

The complete specification of the meta-model consists of a listing of rules, like the seven possible on/off rules for the Sierpinski Triangle example, although these rules have no neat ordering and many are continuous or multi-valued. The rules listed as TBD (to be determined) are the free parameters for whose values the optimization will search.

	D	H
D	(4, 4)	(-6, 6)
H	(6,-6)	(-5, -5)

Prisoner's dilemma

	D	H
D	(4, 4)	(-5, 6)
H	(6,-5)	(-6, -6)

Hawk-dove

	D	H
D	(2, 2)	(-1, -1)
H	(-1,-1)	(2, 2)

Coordination

Figure 2: The payoff matrices that may be used by the model.

[7] Notice, in fact, that the configurations that produce the desired figure divide into a set of configurations with rule seven off—9, 13, 41, and 45—and that exact set repeated with rule seven ($2^{7-1} = 64$) on—73, 77, 105, and 109.

- The grid is an $n \times n$ torus (top connects to bottom; left connects to right), where n is TBD.
- The sequence of play is: all agents move, all play a game with one neighbor (if any), all possibly spawn a new agent, all who meet certain criteria die.
- All agents are one of two types. Borrowing from the hawk/dove game, they are named H or D.
- Initially, agents are placed at random on the grid with a density TBD. These initially-placed agents have equal odds of being H or D.
- Each agent moves one space at a time to a randomly selected adjacent space. If no adjacent spaces are free, the agent does not move. The simulation here uses the 8-direction topology; leaving the choice of topologies TBD is left as an extension for future work.
- Each agent randomly selects a neighbor (if any) with which to play. The payoff will be based on one of the games in Figure 2; which table is used for a given run is TBD.
- If the agent reaches a given wealth level (currently fixed at 12 units), then the agent spawns a child. The child will have the same type as the parent, unless there is a mutation, which occurs with likelihood TBD. The child is placed in a random empty space adjacent to the parent; if no such space exists, the parent can not spawn.
- There are a number of rules by which agents can die. The first is taken from Conway's Game of Life. [Gardner, 1983].
  - Loneliness: agents have zero or one neighbors.
  - Old age: agents are over 11 periods old.
  - Poverty: agent wealth falls below zero.

  A model may use between zero and three of these rules. The simulations here used the poverty and old age rules; allowing the use of all three rules TBD is left as a future extension.

The goals As in Figure 3, different rules lead to different patterns in outcomes. The figure shows three types of outcome: large clumps of each type, large clumps of one type surrounded by the other type, and a general intermixing. Formally, these general forms provide three outcome metrics:

- Clumping. This can be read as 'edge minimization': the goal is to minimize the ratio of (edges between an agent and another agent or open space) to (total number of agents).

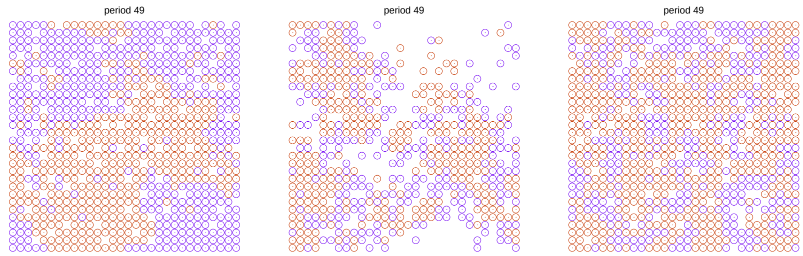

Figure 3: Different games lead to different patterns of outcomes. Left: the coordination game leads to two solid blocks [edges wrap around]. Center: the Prisoner's dilemma leads to clumps of doves surrounded by hawks. Right: the hawk/dove game leads to a closer mixing of types.

- Surrounding. Maximize the percentage of agents who are adjacent to both a blank square and an agent of opposite type.
- Intermixing. Maximize the ratio of (edges between agents of different types) and (number of agents).

The entire metagame includes a great many free parameters, from relatively small details like the age of death to the basic structure of the payoff table. It may be impossible to predict before the fact what rules will have a significant impact—perhaps the initial density really is the determining factor in the structure of outcomes. For the purposes of the testing here, the additional degrees of freedom make it less-than-trivial for an optimization routine to find the best point in the space.

The results

Above, the paper discussed three outputs to the likelihood search: the most likely outcome itself, the sensitivity of the outcome to each metaparameter, and likelihood ratio tests regarding whether a constraint on the system produces a real improvement. The Sierpinski triangle example showed how the variance matrix can be used to glean information about the models; this section will look at the other two outputs.

The most likely outcome The most clear and simple benefit from a maximum likelihood search is that it saves the researcher the pain of finding optimal parameters manually.[8]

Figure 4 shows the results of the search for the configuration with the least edges, given the Coordination payoff table. The game is played for 26 periods, on a small board.[9] The initial density is 21%. The mutation

[8]The data used in this section was produced using approximately 1.8 million runs of the game, which took several hours using a typical high-end PC.

[9]The board is fixed to be of size 10 or greater, so the size here can be read to mean 'as small as possible.'

periods	26.780228
boardsize	10
density	0.214842
passed_on	0.050099
mutation	0.013070

Figure 4: The optimal configuration for an edge-minimizing coordination game.

rate is only 1% and the amount that a parent passes on to its child is only 5%—both amounts that are lower than what one typically sees in an ABM of this type.

It is hard to imagine a sequence of manual tests that would arrive at the values above, especially given that modifications in one variable can readily offset others. As above, a thorough and direct grid search could take literally years. Further, the last two variables took on values that a manually-searching researcher may not have even considered.

Likelihood ratio tests The typical setup for a likelihood ratio (LR) test involves an unconstrained and a constrained model. The null hypothesis is that the constraint is irrelevant, meaning that the log likelihood of the system given the constraint is the same as the log likelihood of the system without the constraint. The statistic for such a likelihood ratio test is $S \equiv 2(L_u - L_c)$, where L_u is the likelihood at the global unconstrained optimum, and L_c is the likelihood at the optimum given a constraint. The statistic has a χ^2_1 distribution [Vuong, 1989].

Figure 5 presents the results of the LR tests. For the edge minimization outcome, the coordination payoff table is the most likely (using the subjective likelihood function), meaning that it is the unconstrained optimum. This verifies the observations of Epstein [2006, pp 221–224], but with the added bonus of confidence intervals. If the system is constrained to use the Hawk-Dove payoff table, then the likelihood drops with 98.8% confidence. The Prisoner's dilemma table causes the likelihood to fall with 89.2% confidence, which is generally considered to be not statistically significant, but which can be read to provide modest evidence that the PD payoff table will produce a lower outcome than the Coordination payoffs.

The edge maximization outcome shows a similar pattern: the Hawk-Dove game is most likely in this case, and the other two payoff tables can be said to reduce the outcome measure with 92.6% and 84.4% confidence. The bordering outcome measure found little difference in outcome given the three payoff tables.

To read the LR tests in the opposite direction, if we observed a world where agents tended to clump

Payoff table	$L(\cdot)$	S	conf.
	Edge minimization		
Coordination	0.869537		
Prisoner's dilemma	-0.429038	2.597	89.2%
Hawk-Dove	-2.30259	6.344	98.8%
	Edge maximization		
Hawk-Dove	-0.574383		
Coordination	-1.58324	2.018	84.4%
Prisoner's dilemma	-2.17568	3.20	92.6%
	Surrounding		
Prisoner's dilemma	1.3282		
Coordination	0.945448	0.766	61.8%
Hawk-Dove	0.776576	1.103	70.6%

Figure 5: LR test results

together, the LR tests indicate that it is unlikely that that world was the product of agents using a Prisoner's dilemma or Hawk-Dove payoff table. It is certainly possible that the agents used any of a multitude of other tables not tested here. No statistical test of any sort can ever prove causation, but taken with other evidence regarding the situation, the LR test can be read to provide supporting evidence to the claim that the Coordination payoff table causes an edge-minimized outcome, while the other two payoff tables do not.

Conclusion

A maximum likelihood search provides more than just the most likely outcome. It also provides information about the sensitivity of the outcome to changes in parameters and fodder for traditional statistical hypothesis tests. These results from the statistical literature can be directly applied to the problem of designing agent-based models to fit observed patterns, thus producing models that better fit the patterns—and how well they fit can be measured and tested.

Appendix: Calculating the covariance matrix

The variance of a maximum likelihood estimate of a parameter achieves the Cramér-Rao lower bound (CRLB). Given a maximum likelihood estimate based on n data points $\hat{\beta}(y_1, \ldots, y_m)$, a log-likelihood function L, dlog-likelihood vector S (often called the score, and of the form

$$[\partial L(\boldsymbol{\beta})/\partial\beta_1 \quad \partial L(\boldsymbol{\beta})/\partial\beta_2 \quad \cdots \quad \partial L(\boldsymbol{\beta})/\partial\beta_1]),$$

then:

$$\begin{aligned}\mathrm{var}(\hat{\beta}) &= (n\,\mathrm{var}(S))^{-1} \\ &= -\left(nE\left(\frac{\partial^2 L}{\partial\beta^2}\right)\right)^{-1} \\ &\equiv -(n\mathcal{I})^{-1}\end{aligned}$$

There are only two assumptions underlying the derivation of the CRLB. The first is that the integral of the probability distribution over the entire space is constant. In the context here, this means that a subjective probability will have to be normalized to integrate to one.

The second condition for the above result is that

$$\int \frac{\partial p(x,\beta)}{\partial\beta} d\beta = \frac{\partial \int p(x,\beta) d\beta}{\partial\beta}. \tag{1}$$

This too is a very general statement, that is satisfied by any function that demonstrates uniform convergence, which in turn is satisfied by any member of an exponential family. The definition and use of exponential families will not be discussed here, but it is worth noting that Barron and Sheu [1991] demonstrate that any PDF can be approximated arbitrarily closely (measured by Kullback-Leibler distance) by a sum of exponential family distributions.

Finally, the result only makes sense when the derivatives and second derivatives are defined almost everywhere. The framework above makes a point of meeting this criterion: there is a linear mapping from discrete parameters to outcomes, and a linear function clearly satisfies the differentiability requirements above. In most cases, the optimum will be where only one model parametrization is used and the parameter-to-outcome mapping is nondifferentiable, but the CRLB is a global property, so it does not require differentiability at the optimum, only differentiability almost everywhere.

References

David H Ackley. *An Empirical Study of Bit Vector Function Optimization*, pages 170–204. 1987. In Davis [1987].

Robert Axelrod and William D Hamilton. The evolution of cooperation. *Science*, 211(4489):1390–1396, March 1981.

Andrew R Barron and Chyong-Hwa Sheu. Approximation of density functions by sequences of exponential families. *The Annals of Statistics*, 19(3):1347–1369, September 1991.

Lawrence Davis, editor. *Genetic Algorithms and Simulated Annealing*. Research Notes in Artificia Intelligence. Pitman Publishing, 1987.

Joshua M Epstein. *Generative Social Science: Studies in Agent-Based Computational Modeling*. Princeton University Press, 2006.

Joshua M Epstein. Zones of cooperation in demographic prisoner's dilemma. *Complexity*, 4(2):36–48, 1998.

Martin Gardner. *Wheels, Life, and Other Mathematical Amusements*. W H Freeman, 1983.

Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2000.

Roberto Leombruni, Matteo Richiardi, Nicole J Saam, and Michele Sonnessa. A common protocol for agent-based social simulation. *Journal of Artificial Societies and Social Simulation*, 9(1), 2006.

Quang H Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57 (2):307–333, March 1989.

Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.