# Finding Forms of Flocking: Evolutionary Search in ABM Parameter-Spaces

Forrest Stonedahl and Uri Wilensky

Center for Connected Learning and Computer-Based Modeling
Northwestern University, Evanston, IL, USA
forrest@northwestern.edu, uri@northwestern.edu

**Abstract.** While agent-based models (ABMs) are becoming increasingly popular for simulating complex and emergent phenomena in many fields, understanding and analyzing ABMs poses considerable challenges. ABM behavior often depends on many model parameters, and the task of exploring a model's parameter space and discovering the impact of different parameter settings can be difficult and time-consuming. Exhaustively running the model with all combinations of parameter settings is generally infeasible, but judging behavior by varying one parameter at a time risks overlooking complex nonlinear interactions between parameters. Alternatively, we present a case study in computer-aided model exploration, demonstrating how evolutionary search algorithms can be used to probe for several qualitative behaviors (convergence, non-convergence, volatility, and the formation of vee shapes) in two different flocking models. We also introduce a new software tool (BehaviorSearch) for performing parameter search on ABMs created in the NetLogo modeling environment.

**Key words:** parameter search, model exploration, genetic algorithms, flocking, agent-based modeling, ABM, multi-agent simulation

## 1 Motivation

Agent-based modeling is a powerful simulation technique in which many agents interact according to simple rules resulting in the emergence of complex aggregate-level behavior. This technique is becoming increasingly popular in a wide range of scientific endeavors due to the power it has to simulate many different natural and artificial processes [1, 3, 22]. A crucial step in the modeling process is an analysis of how the system's behavior is affected by the various model parameters. However, the number of controlling parameters and range of parameter values in an agent-based model (ABM) is often large, the computation required to run a model is often significant, and agent-based models are typically stochastic in nature, meaning that multiple trials must be performed to assess the model's behavior. These factors combine to make a full brute-force exploration of the parameter space infeasible. Researchers respond to this difficulty in a variety of ways. One common approach is to run factorial-design experiments that either explore model behavior only in a small subspace or explore the full space

but with very low resolution (which may skip over areas of interest). A second common approach is to vary only a single parameter at a time, while holding the other parameters constant, and observe the effect of changing each parameter individually. However, because ABMs often constitute complex systems with non-linear interactions, these methods risk overlooking parameter settings that would yield interesting or unexpected behavior from the model.

As an alternative, we argue that many useful model exploration tasks may instead be productively formulated as *search problems* by designing appropriate objective functions, as we will demonstrate by example in the domain of simulated flocking behavior. In this paper, we introduce a new software tool (*BehaviorSearch*) that we have created for the purpose of searching/exploring ABM parameter spaces. Using *BehaviorSearch*, we offer a case study showing how search-based exploration can be used to gain insight into the behavior of two ABMs of flocking that have been implemented in the NetLogo modeling environment [21, 18]. We also provide a comparison of the performance of three different search algorithms on several exploratory tasks for these two ABMs. In particular, we will show how genetic algorithms and hill-climbing can be used to discover parameter settings for these models that yield behaviors such as convergence, non-convergence, volatility, and specific flock shape formation. This approach can be useful for researchers to better understand the models they have created, the range of behavior their models are capable of producing, and which parameters have large impact on which behaviors. Flocking behaviors were chosen for this case study because flocking is a well-known example of a successful agent-based model, and can demonstrate a wide range of behaviors depending on the controlling parameters.

## 2 Related Work

Rather than using a full factorial experiment design for sampling points in the space, several more sophisticated sampling algorithms exist (e.g. Latin hypercube sampling, sphere-packing). These algorithms stem from the design of experiments (DoE) literature or more specifically the more recent design and analysis of computer experiments (DACE) literature (see [14] for a discussion of applying DACE methodology to ABMs). While appropriate experimental designs provide efficient sampling of the space in some situations, this is a separate direction from the search-oriented approach that we are pursuing here. In particular, we are interested in the use of genetic algorithms [7] (GAs) to search the ABM parameter spaces for behaviors of interest. Genetic algorithms have proven to be quite successful on a wide range of combinatorial search and optimization problems, and are thus a natural meta-heuristic search technique for this task. There is prior work on parameter-search and exploration in ABM, and considerably more on the problem of parameter-search in general.

Calvez and Hutzler have previously used a genetic algorithm (GA) to tune parameters of an ant foraging model [4], and discuss some of the relevant issues for applying GAs to ABM parameter search. However, in this case, the GA's

performance was not compared to any other method, and the effectiveness of GAs for the ABM parameter search task has not been thoroughly investigated. Our present work contributes toward this goal. Specifically, we compare the performance of a genetic algorithm against a stochastic mutation-based hill-climber, as well as uniform random search, to serve as a baseline for comparison. We also explore a different domain (i.e. flocking models rather than ant foraging), and thus provide another perspective on the issue of automated model exploration.

Genetic algorithms have also been used to attempt to calibrate agent-based models with aggregate-level equation-based models as part of the SADDE methodology [15] for designing ABMs. Our research places an emphasis on exploration, as opposed to calibration or model design. The modeler may pose a question about the model's behavior which are potentially interesting, and the distribution of search results should answer that question, and may give additional insight into the interaction between parameters as well.

Other methods of exploration (besides genetic algorithms) have previously been considered. Most notably, Brueckner and Parunak proposed a meta-level multi-agent system to adaptively select points in the parameter-space to evaluate [2]. This swarm-based approach resembles particle swarm optimization [8] in that it uses a population of agents that combine global and local information to choose a direction to move in the search space, but it also considers whether to run additional simulations to improve the confidence of results at locations in the space. Brueckner and Parunak also mention in passing that genetic algorithms would be an appropriate choice for this type of search problem, but they did not follow this path, and only offer results from the novel multi-agent optimization algorithm they proposed. A comparison of genetic algorithms with this, and other swarm-based approaches, would be an interesting area for future work.

Genetic algorithms have also been employed in parameter-search problems which are not ABM, but closely related fields. For instance, genetic algorithms have been applied to search for rules in cellular automata (CA) that will produce a certain behavior (e.g. density classification) [11]. Cellular automata models could be considered a highly restricted case of agent-based models, and the cell state transition rules could perhaps be considered the *parameters* of such models, in which case this would constitute searching the parameter space. However, agent-based simulations more typically have numeric parameters, and whereas CA rules are naturally represented by binary switches, and the density-classification task is closer to a multi-agent system coordination problem, rather than an agent-based simulation.

Our present investigation is also inspired by Miller's work on active non-linear testing [9], which demonstrated the use of meta-heuristic optimization (genetic algorithms and hill climbers) for searching the parameter-space of the *World3* simulation, a well-known system dynamics model (SDM). Our work departs from Miller's in two respects: 1) model stochasticity (which is less frequently present in SDMs) is not addressed in those experiments, and 2) the characteristics of search spaces produced by agent-based models likely differ from those which are produced by aggregate equation-based models.

## 3 Methods

### 3.1 Flocking Models Overview

For our case study we explore the parameter-space of two agent-based models, searching for a variety of target behaviors. The two ABMs are the Flocking model [20] (denoted as *Flocking*) and the Flocking Vee Formations model [23] (denoted as *Flocking VF*). While the parameters of these two models are discussed briefly below, an in-depth discussion of these models is beyond the scope of this paper. Thus, we invite interested readers to examine the models themselves, which are both available in the NetLogo models library.

*Flocking* closely resembles the seminal ABM of swarming behavior in artificial birds (playfully dubbed "boids") that was introduced by Reynolds as a way to create life-like cinematic animation of flocking birds or other flying/swimming/swarming creatures [13]. The behavior of each "boid" is influenced by three basic rules, which provide impetus toward alignment, coherence, and separation. The relative influences of each are controlled by the parameters max-align-turn, max-cohere-turn, and max-separate-turn, respectively. Additionally there are parameters controlling the distance at which birds have knowledge of other birds (vision), and the minimum distance of separation which birds attempt to maintain (minimum-separation). For this first model, exploratory search tasks include the discovery of parameters that yield quick directional convergence (Section 4.1), non-convergence (Section 4.2), and volatility of the aggregate flock's heading over time (Section 4.3).

*Flocking VF* is based loosely on an extension of Reynolds' work that was proposed by Nathan and Barbosa [12], attempting to produce the vee-shaped patterns often observed in large migratory birds, such as Canada geese. *Flocking VF* has 8 controlling parameters, which account for fine-grained control over bird vision (vision-distance, vision-cone, obstruction-cone), takes into account benefits of "updraft" from nearby birds (updraft-distance, too-close), as well as flying speeds and acceleration (base-speed, speed-change-factor, and max-turn). The final exploratory search task is to seek parameters that best yield V-shaped flock formations, in both *Flocking* and *Flocking VF* (Section 4.4).

### 3.2 Search Algorithms

For each search task, we tested three different search algorithms: uniform random search (RS), a random-mutation hill climber (HC), and a genetic algorithm (GA). For all of the search methods, each ABM parameter's value was encoded as a sequence of binary digits (bit string) using a Gray code[1], and all the parameters' bit strings were concatenated to create a string that represents one point in the parameter-space. A bit string is evaluated by decoding it into the ABM parameter settings, and running the model with those parameters.

[1] A high-order binary encoding requires flipping 4 bits to change from 7 ($0111_2$) to 8 ($1000_2$). In a Gray code, consecutive numbers only require a single bit flip, thus creating a smoother mapping from numbers into binary search spaces.

The RS method simply generates one random bit string after another, and in the end chooses the one that best elicited the desired model behavior. RS is a naive search techniques, which we included as a baseline for comparison, to determine whether using more sophisticated meta-heuristics (such as the HC and GA) were indeed helpful.

Our HC is primarily a local search algorithm. It starts with a random bit string ($s$). A new string ($s_{new}$) is generated from $s$ (each bit of $s$ gets flipped with probability 0.05, which is the *mutation-rate*). If $s_{new}$ is better than $s$ (generates behavior that judged closer to the desired target behavior), then the HC chooses $s_{new}$ as the new $s$, and the process repeats. If the HC becomes stuck (after 1000 unsuccessful move attempts), it will restart at a new random location in the search space, which makes this a quasi-local search method.

Our GA is a standard generational genetic algorithm [7], with a population size of 30, a crossover rate of 0.7, and a mutation rate of 0.05, using tournament selection with tournament size 3. The GA is a more sophisticated search mechanism than HC or RS, and there are several reasons to believe that it might perform better. First, the GA is population-based, which allows it to explore multiple regions of the space simultaneously (more of a global search technique). Second, genetic algorithms have previously been shown to perform well on a variety of nonlinear and multi-modal search/optimization problems. Third, genetic algorithms (like the biological processes of evolution that inspired them) often have a way of coming up with creative or unexpected solutions to a problem, which humans would not have considered. However, depending on the how the search space is structured, simpler approaches may be more effective. For example, it was shown that a HC performed better on a problem that was specifically designed with the expectation that GAs would work well on it [10]. One important consideration, is whether there are so-called *building blocks* in the solution-space, which the GA is able to discover and combine (via genetic crossover) to form better solutions. Phrased at the level of the agent-based model, this question becomes: are there certain combinations of several parameter settings, each of which partially produce desired target behavior, and when combined together produce that behavior even more strongly? If so, the GA may be able to take advantage of that structure in the search space to efficiently find solutions.

The objective function (or "fitness function" in the parlance of evolutionary computation) was always averaged across 5 model runs (replicates) with different random seeds, to reduce variability stemming from model stochasticity. While this variability is essentially "noise" from the search algorithm's perspective, it is simply a reflecting the fact that running the ABM results in a range of behavior depending on the initial placement of the birds. Our objective functions are attempting to characterize the presence or absence of a certain behavior *on average*, and short of running the simulation with every possible initial condition (which is impossible), there will always be some uncertainty about the objective function measure. Taking the average value from several replicate runs of the simulation, however, reduces this uncertainty and smooths the search landscape.

The objective functions were different for each task, and will be discussed individually in each of the investigations below (Sections 4.1-4.4). For efficiency, objective function values were cached after being computed.[2] The search algorithms were stopped after they had run the ABM 12000 times. Each search was repeated 30 times (except for the *volatility* exploration in Section 4.3, which was repeated 60 times for improved statistical confidence), to evaluate search performance and ensure that search findings were not anomalous.

### 3.3 BehaviorSearch

To perform these searches, we developed a new tool called *BehaviorSearch* [16], which was implemented in Java, and interfaces with the NetLogo modeling environment, using NetLogo's *Controlling API*. *BehaviorSearch* is an open-source cross-platform tool that offers several search algorithms and search-space representations/encodings, and can be used to explore the parameter space of any ABM written in the NetLogo language. The user specifies the model file, the desired parameters and ranges to explore, the search objective function, the search method to be used, and the search space encoding, and then *BehaviorSearch* runs the search and returns the best results discovered, and optionally the data collected from all of the simulations run along the way. *BehaviorSearch* supports model exploration through both a GUI (see Figure 1), and a command line interface. A beta-release of *BehaviorSearch* is freely available for download[3]. The software design purposefully resembles that of the widely-used *BehaviorSpace* [19] parameter-sweeping tool that is included with NetLogo. Our intent is to make advanced search techniques accessible to a wide range of modelers so that the methods and ideas discussed in this paper can be put into practice.

## 4 Explorations

### 4.1 Investigation 1: Convergence

The convergence of swarm-based systems is one potential property of interest, and has been formally studied for some theoretical cases [5]. Thus, the first behavior of interest for the *Flocking* model was the ability of birds starting at random locations and headings to converge to be moving in the same direction (i.e. directional, not positional, convergence). In order to make the search process effective, we must provide a quantitative measure to capture the rather qualitative notion of convergence. This quantitative measure (the objective function) will provide the search with information about how good one set of parameters is, relative to another, at achieving the goal. Specifically, we would like to find

[2] The goal of caching is to avoid repeating expensive computations. However, because the model is stochastic, re-evaluating points in the search space could lead to different results than the cached values, meaning that the search process is affected by caching. For further discussion of noise/uncertainty and fitness caching, see [17].

[3] Available at: `http://www.behaviorsearch.org/`

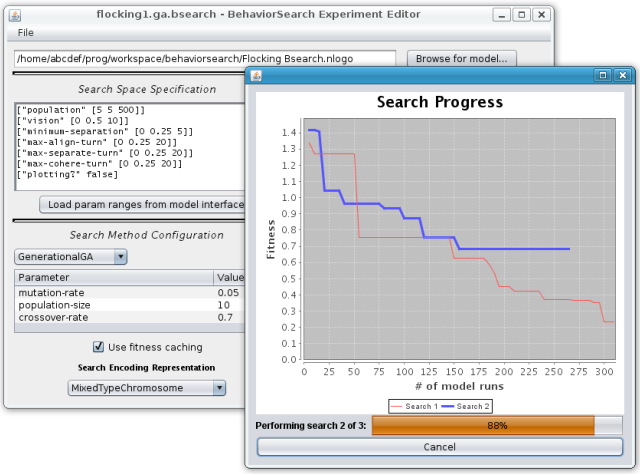


**Fig. 1.** Screenshot of the BehaviorSearch GUI, displaying search progress.

parameters that yield very little variation between birds' headings. Thus, we will attempt to minimize the following objective function:

$$f_{nonconverged} = stdev(\{v_x(b) \mid b \in B\}) + stdev(\{v_y(b) \mid b \in B\}) \qquad (1)$$

where $v_x(b)$ and $v_y(b)$ are the horizontal and vertical components of the velocity of bird $b$, and $B$ is the set of all birds. The standard deviation ($stdev$), which is the square root of the variance, serves as a useful measure of the variation for velocity, and we must apply it in both the $x$ and $y$ dimensions. A value of $f_{nonconverged} = 0$ would indicate complete alignment of all birds. We measure $f_{nonconverged}$ after 75 ticks (model time steps). While 75 ticks is effective here for finding parameter settings that cause the flock to quickly converge, if we were instead interested in the long-term behavior of the system, a longer time limit would be more appropriate.

The plot of search progress (Figure 2) shows that on average the HC may have found better model parameters early in the search, but in the end the GA's performance was superior (t-test, $p < 0.01$). Both GA and HC significantly outperformed random search. The best parameters found in each run (Figure 3) shows us that it is crucial for birds to have long-range vision, and that even a small urge to cohere is detrimental to convergence. The wide spread for max-separate-turn suggests that convergence is not very sensitive to this parameter (given the other parameter settings). Figure 3 also shows one possible converged state from running the model using the best parameters found by the GA.

### 4.2 Investigation 2: Non-convergence

Next, we probed for parameter settings that cause the birds not to globally align. For this task, we simply maximized the same objective function we min-

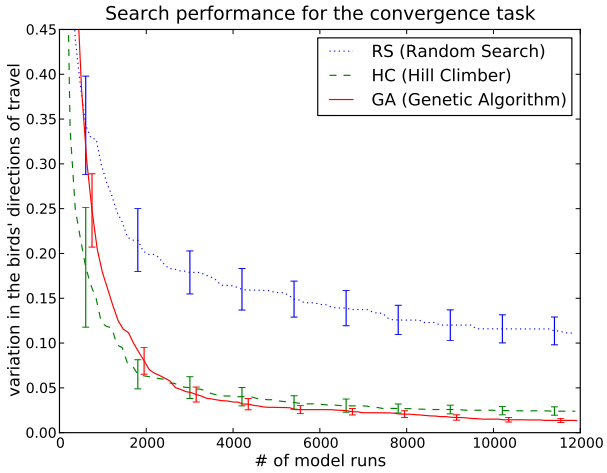


**Fig. 2.** Search performance for the convergence task, comparing how efficiently the GA (*genetic algorithm*), HC (*hill climber*), and RS (*random search*) can find parameters that cause the flock to quickly converge to the same heading. (Error bars show 95% confidence intervals on the mean.)

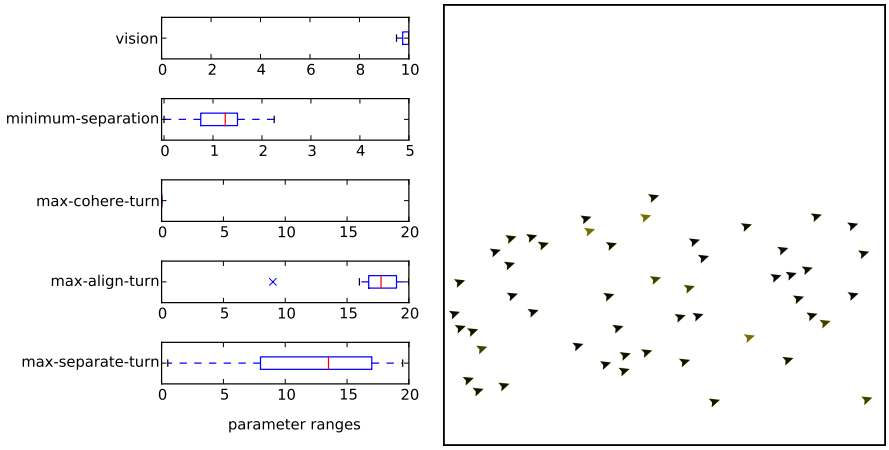


**Fig. 3.** *LEFT:* Distribution of model parameter settings found to cause quickest convergence in each of the 30 GA searches. All box-and-whisker plots presented in this paper show the median line within the lower-to-upper-quartile box, with whiskers encompassing the remainder of the data, apart from outliers which are marked with x's. *RIGHT:* Visualization of the flock (after 75 model steps) using the best parameters the GA discovered.

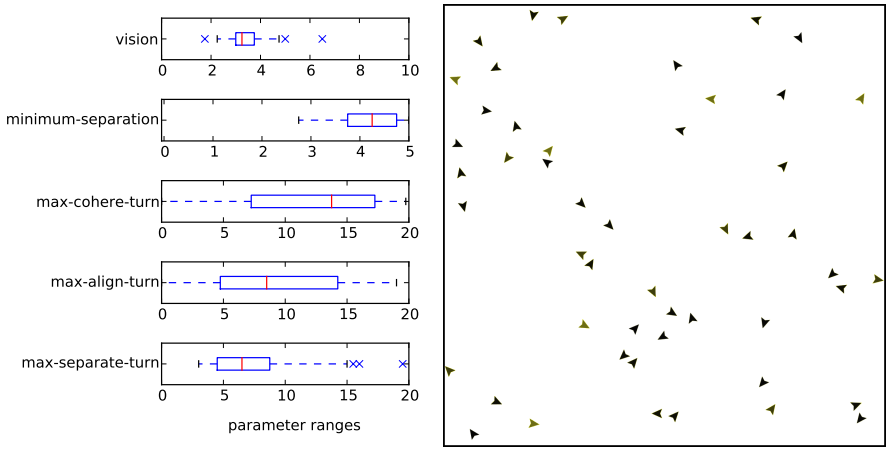


**Fig. 4.** *LEFT:* Distribution of model parameter settings found to cause non-convergence in each of the 30 GA searches. *RIGHT:* Visualization of a non-converged flock using the best parameters the GA discovered.

imized in Section 4.1. This task turned out to be rather trivial, as all three search methods (GA, HC, and RS) very quickly found parameter settings that yielded little or no flock alignment. That such behavior is rather common in the parameter space is illustrated by Figure 4, which shows a wide distribution of best parameters. The results suggest that for non-convergence, it is helpful for birds to have a low-to-medium vision range, desire a large amount of separation from each other (minimum-separation), and act to achieve the separation (non-zero max-separate-turn). Digging deeper, the results tell us that it is the relationship between parameters that matters; if minimum-separation is larger than vision each bird will seek to avoid any other bird as soon as it sees it, as separation takes precedence over the align/cohere rules.

### 4.3 Investigation 3: Volatility

Our third experiment sought parameters for the *Flocking* model that would yield the most volatility (or changeability) in global flock heading, by attempting to maximize $f_{volatility}$, as defined in (4).

$$\overline{v_x}(t) = mean(\{v_x(b) \mid b \in B\} \text{ at tick } t \tag{2}$$

$$\overline{v_y}(t) = mean(\{v_y(b) \mid b \in B\} \text{ at tick } t \tag{3}$$

$$f_{volatility} = stdev(\overline{v_x}(t) \text{ for } t = 400..500) + stdev(\overline{v_y}(t) \; for \; t = 400..500) \tag{4}$$

Again, on average the GA was slightly more successful than the HC in eliciting flock heading volatility, and both significantly outperformed random search (Figure 5). Only 5 out of the 60 GA searches' best parameter settings had a non-zero value for minimum-separation, indicating that birds flying close together is a

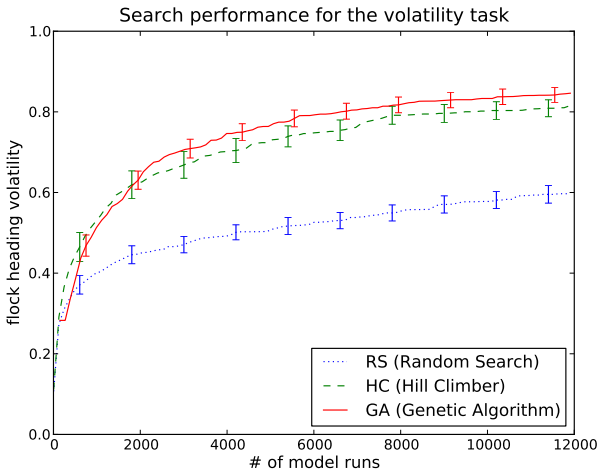


**Fig. 5.** Comparison of search algorithm performance for the flock heading volatility task. The final mean performance of the GA was better than the HC (t-test, $p < 0.05$), but not substantially so. (Error bars show 95% confidence intervals on the mean.)

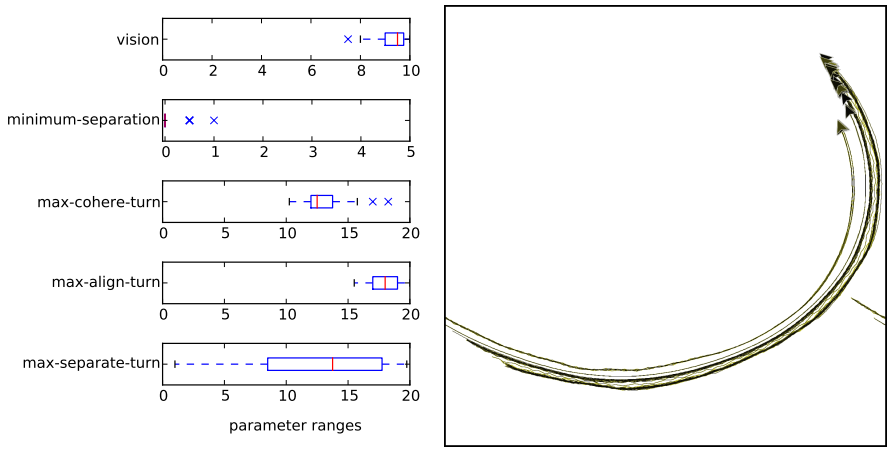


**Fig. 6.** *LEFT:* Distribution of model parameter settings (from each of the 30 GA searches) found to cause the most volatility in flock heading. *RIGHT:* Visualization of the flock after 500 model steps (also showing each bird's path over the last 100 steps), using the best parameters found by the GA.

key factor for maximal volatility. Long-range vision, and large effects of max-align-turn and max-cohere-turn are also important (see Figure 6). The flight pattern of a flock exhibiting considerable volatility is shown in Figure 6. The single bird positioned at the left side in the rear is at least partially responsible for shift in flock heading, because of the strong coherence parameter.

Despite taking the average of 5 replications, noise due to model stochasticity was still significant. For example, the search reported finding settings yielding 0.99 volatility, but averaging 1000 runs at those settings showed true volatility of 0.41. This fact could bias the search toward parameters that occasionally yield very high volatility, over those that consistently yield moderately high volatility. Both goals are potentially interesting for model exploration; however, appropriate noise reduction methodology is a worthy subject for future research.

### 4.4 Investigation 4: Vee Formations

The final experiment was to search both the *Flocking* and *Flocking VF* models for a more complex behavior, which we shall refer to as *veeness*. *Veeness* measures the degree to which birds are flying in vee, or more generally, echelon formations. Our specific questions are: 1) Do any parameter settings cause *Flocking* to exhibit *veeness*? 2) How much better can *Flocking VF* do? and 3) What parameters are most important for the best vee/echelon creation?

To calculate *veeness*, we first cluster all the birds in the world into separate flocks, according to proximity (within 5 distance units of another bird in the flock) and directional similitude (less than 20 degrees angular difference in heading). A flock with less than 3 birds is assigned a flock veeness score of 0. Otherwise, it is calculated by choosing the optimal *point bird* and left/right echelon angles (which must be between 25 and 50 degrees, comprising a mid-range of echelon angles observed in nature [6]) for the flock. For a given point bird, the left and right echelon angles are calculated separately, by first dividing flockmates into those to the right or left, relative to the point bird. The echelon angles are then chosen such that they minimize the mean-squared-error difference between the echelon angle and the angle between the point bird and all following birds on that side. Flock groupings with echelon angles and flock veeness scores can be seen in Figure 9. The flocking score for the flock is the reciprocal of the mean-squared-error value for the best "point" bird, rescaled so that a flock in perfect echelon/vee formation has a score of 1.0. Overall *veeness* is a weighted average (by flock size) of the veeness scores of individual flocks. *Veeness* was measured every 100 model ticks, between 1000 and 2000 ticks. Searches for both *Flocking* and *Flocking VF* used 30 birds and the same veeness metric.

Unlike in previous experiments, the HC search method performed slightly better than the GA (see Figure 7), but the difference was not statistically significant. For the Flocking model, RS was not far behind the GA and HC, but was considerably worse than the other methods for the Vee Flocking model.

The results show that *Flocking* can create formations that appear only mildly vee-like at best, but *Flocking VF* can (as expected) create much better vees (as shown in Figure 9). For *Flocking VF* to produce the best vees (according to our

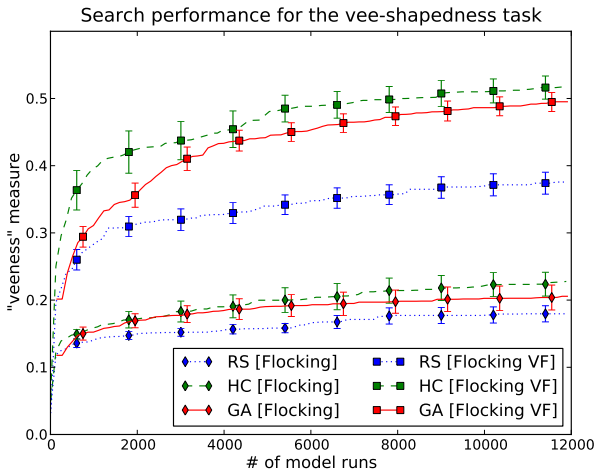


**Fig. 7.** Comparison of search performance for the vee-shapedness task on both the Flocking and Flocking Vee Formation models. (Error bars show 95% confidence intervals on the mean.)

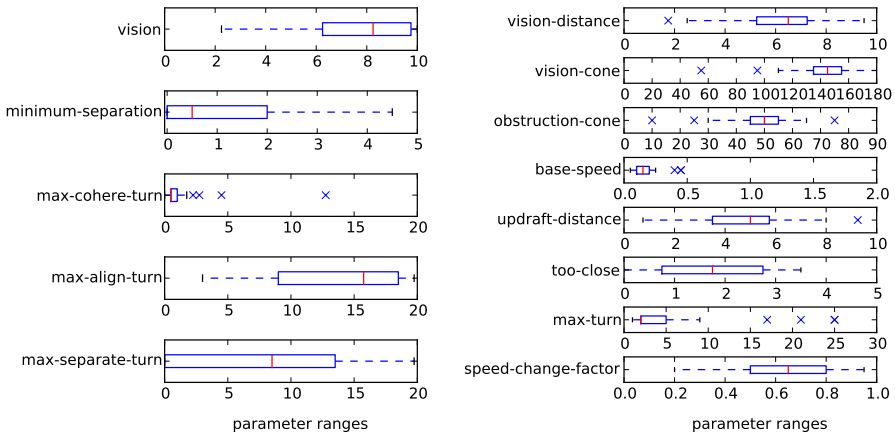


**Fig. 8.** Distribution of model parameter settings found to yield the best vees in the Flocking model (*left*), and the Flocking Vee Formation model (*right*), in each of the 30 HC searches.

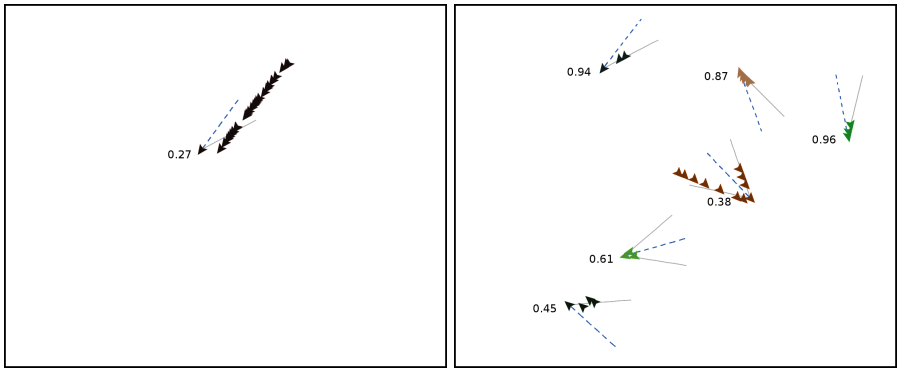


**Fig. 9.** Visualization of a run of the Flocking model (*left*), and the Flocking Vee Formation model (*right*), using the best "vee-forming" parameters found by the 30 HC searches. Birds are shaded by flock group, dashed lines show average flock heading relative to the "point" bird, and gray lines show best-fit angles for right and/or left echelons of the vee formation. The numeric "veeness" measure for each individual flock is also shown.

chosen *veeness* metric), the vision-cone angle should be large, perhaps roughly 3 times larger than the obstruction-cone angle, the bird's base-speed and max-turn angle should generally be low, but the speed-change-factor should not be too small. We will not elaborate on specific implications of these findings for the *Flocking VF* model here, but broadly argue that findings such as these can lead modelers to a better understanding of their model by cognitively linking changes in model parameters with the qualitative behavior being investigated.

## 5 Conclusion and Future Work

Beyond the specific results concerning the behavior of two particular agent-based models (Flocking and Vee Flocking), there are several more general conclusions that may be drawn from this case study. First, evolutionary algorithms such as the GA and HC are indeed effective means of exploring the parameter space of ABMs. Their performance was vastly superior to RS, except in the cases where the task was too easy (e.g. nonconvergence) or too hard (veeness in *Flocking*) to make substantial progress. (The difficulty of the search task relates to how dense or sparse the desired target behavior is in the search space: parameter settings that cause the Flocking model to not converge are plentiful in the parameter space, whereas parameter settings that cause good vee formations are either extremely rare or nonexistent.) Second, by running multiple searches on a stochastic model and looking at the distribution of best-found parameter settings, rather than just the single best setting for the parameters, we can uncover trends (or at least postulate relationships) about the interactions between model parameters and behavior. One interpretation is that we are implicitly performing a type of sensitivity analysis on the *search process* for a particular *behavior*,

but that the results of that analysis can tell us something about the model. Note that the trends we find are unlikely to be global (characterizing the whole parameter space), but apply only to a local view that is focused on regions of the parameter space where the target behavior is expressed mostly strongly.

These results also suggest several important areas for future work. First, it is unclear what circumstances favor the use of a genetic algorithm over a simpler hill climbing search mechanism. Second, the performance results presented here may be dependent on any number of *search algorithm parameters* (not to be confused with *model parameters*), such as the population size, mutation rate, crossover rate, elitism, or chromosomal representation. While we attempted to choose reasonable values for these search parameters, it is likely that by tuning these parameters, the algorithms' efficiency could be improved. Furthermore, poorly chosen search parameters could lead to worse performance than random search, and should thus be avoided. Also, in future work, we would like to investigate the use of other search algorithms (such as simulated annealing, and particle-swarm optimization). Finally, additional consideration should be given to the treatment of model stochasticity and noisy objective functions; while running fewer replicates of model runs takes less time for searching, large quantities of noise can inhibit search progress. In general, the prospects seem bright for using meta-heuristic search, such as genetic algorithms, to improve model exploration and analysis. It is our hope that these promising prospects will encourage ABM practitioners to flock toward, and eventually converge on, new methodologies for model parameter exploration that take advantage of these ideas.

**Acknowledgments** We especially wish to thank William Rand for constructive feedback on this research, Luis Amaral for generously providing computational resources to carry out our experiments, and the National Science Foundation for supporting this work (grant IIS-0713619).

## References

1. Bankes, S.: Agent-Based Modeling: A Revolution? PNAS 99(10), 7199–7200 (2002)
2. Brueckner, S.A., Parunak, H.V.D.: Resource-aware exploration of the emergent dynamics of simulated systems. In: AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems. pp. 781–788. ACM, New York, NY, USA (2003)
3. Bryson, J.J., Ando, Y., Lehmann, H.: Agent-based modelling as scientific method: a case study analysing primate social behaviour. Philosophical Transactions of the Royal Society B: Biological Sciences 362(1485), 1685–1698 (2007)
4. Calvez, B., Hutzler, G.: Automatic Tuning of Agent-Based Models Using Genetic Algorithms. In: MABS 2005: Proceedings of the 6th International Workshop on Multi-Agent-Based Simulation (2005)
5. Cucker, F., Smale, S.: Emergent behavior in flocks. IEEE Transactions on automatic control 52(5), 852–862 (2007)
6. Heppner, F., Convissar, J., Moonan Jr, D., Anderson, J.: Visual angle and formation flight in Canada Geese (Branta canadensis). The Auk pp. 195–198 (1985)

7. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975)
8. Kennedy, J., Eberhart, R., et al.: Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks. vol. 4, pp. 1942–1948. IEEE, Piscataway, NJ (1995)
9. Miller, J.H.: Active nonlinear tests (ANTs) of complex simulation models. Management Science 44(6), 820–830 (1998)
10. Mitchell, M., Holland, J., Forrest, S.: When will a genetic algorithm outperform hill climbing? In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems, vol. 6, pp. 51–58. Morgan Kaufmann, San Mateo, CA (1994)
11. Mitchell, M., Crutchfield, J.P., Das, R.: Evolving cellular automata with genetic algorithms: A review of recent work. In: Proceedings of the First International Conference on Evolutionary Computation and Its Applications. Russian Academy of Sciences, Moscow, Russia (1996)
12. Nathan, A., Barbosa, V.: V-like formations in flocks of artificial birds. Artificial life 14(2), 179–188 (2008)
13. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques. pp. 25–34. ACM, New York, NY, USA (1987)
14. Sanchez, S.M., Lucas, T.W.: Exploring the world of agent-based simulations: simple models, complex analyses. In: WSC '02: Proceedings of the 34th conference on Winter simulation. pp. 116–126 (2002)
15. Sierra, C., Sabater, J., Augusti, J., Garcia, P.: SADDE: Social agents design driven by equations. In: Methodologies and software engineering for agent systems. Kluwer Academic Publishers (2004)
16. Stonedahl, F.: BehaviorSearch [computer software]. Center for Connected Learning and Computer Based Modeling, Northwestern University, Evanston, IL. Available online: `http://www.behaviorsearch.org/` (2010)
17. Stonedahl, F., Stonedahl, S.: Heuristics for sampling repetitions in noisy landscapes with fitness caching. In: GECCO '10: Proceedings of the 12th annual conference on genetic and evolutionary computation. ACM, New York, NY, USA ([in press])
18. Tisue, S., Wilensky, U.: NetLogo: Design and implementation of a multi-agent modeling environment. In: Proceedings of Agent 2004. pp. 7–9 (2004)
19. Wilensky, U., Shargel, B.: BehaviorSpace [Computer Software]. Center for Connected Learning and Computer Based Modeling, Northwestern University, Evanston, IL. `http://ccl.northwestern.edu/netlogo/behaviorspace` (2002)
20. Wilensky, U.: NetLogo Flocking model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. (1998)
21. Wilensky, U.: NetLogo. Center for Connected Learning and Computer-based Modeling, Northwestern University, Evanston, IL (1999)
22. Wilensky, U., Rand, W.: An introduction to agent-based modeling: Modeling natural, social and engineered complex systems with NetLogo. MIT Press, Cambridge, MA (in press)
23. Wilkerson-Jerde, M., Stonedahl, F., Wilensky, U.: NetLogo Flocking Vee Formations model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. (2010)