

# Resource-Aware Exploration of the Emergent Dynamics of Simulated Systems

Sven A. Brueckner  
Altarum

3520 Green Court  
Ann Arbor, MI 48105-1570, USA  
+1 (734) 302 4683

sven.brueckner@altarum.org

H. Van Dyke Parunak  
Altarum

3520 Green Court  
Ann Arbor, MI 48105-1570, USA  
+1 (734) 302 4684

van.parunak@altarum.org

## ABSTRACT

The emerging science of simulation enables us to explore the dynamics of large and complex systems even if a formal representation and analysis of the system is intractable and a construction of a real-world instantiation for the purpose of experimentation is too expensive. A computer simulation model can be run for many more configurations and the accumulated observations deepen our understanding of the system's operation, but it is very important that we have tools that help us manage the huge numbers of experiments that need to be run and the massive data sets that are collected. Furthermore, as we explore vast parameter spaces of simulation model, we need guidance in finding regions of interest in a resource efficient way.

In this paper we use a model of agent-based graph coloring to introduce a software infrastructure for the systematic execution of experiments across large regions of parameter space (parameter sweep). Furthermore, we present a multi-agent system that searches large parameter spaces automatically for regions of interest specified by a fitness function. The fitness function captures the researcher's interest in certain system dynamics. We specify a function that searches for overlap regions that accompany phase changes in the simulation model. The agents search the parameter space by executing simulation experiments in regions of high fitness. As a consequence, the use of computational resources is minimized.

## Keywords

Multi-Agent Coordination, Simulation, System Dynamics, Tools and Methods

## 1. INTRODUCTION

Large multi-agent systems may express very complex dynamics even if the individual agents and their interactions are simple and easy to represent and analyze. This emergent complexity may be even higher if the agents are embedded in a real-world environment that introduces additional constraints and dynamics.

Consider for example an agent that controls a small segment of a material handling system. The segment has a number of entry

and exit points and the agent's only task is to transfer incoming material to one of the exits sequentially. The agent's decision process is extremely simple. It cyclically decides whether to move the next item from an entry to an exit and if it does, it prefers to move items of the same type to the same exit as it did in the past. This very simple agent behavior leads to the emergence of very robust and flexible material sorting dynamics if multiple segments (and their agents) are joined into a larger system [3, 4]. While an implementation of the agent system is very simple and straightforward, a formal analysis of the emerging dynamics and expected sorting performance is far from trivial.

In the case of the emergent material sorting as well as for many other agent systems it is possible to construct a sometimes abstract simulation of the agent and environmental dynamics even if a formal model of this system is not attainable. This simulation approach is the middle ground between the two traditionally chosen approaches in science – formal analysis and experimentation with real systems. These traditional approaches are no longer feasible for the type of systems we are confronted with. The emergent dynamics of large multi-agent systems quickly become intractable to any non-trivial formal representation and analysis and their sheer scale results in prohibitive costs for a real-world implementation for experimental purposes. Thus we see an emergence of a science of simulation [9] to which this paper hopes to contribute.

In experiments with a software simulation we can observe the dynamics of a system in many more scenarios than in experiments with a real system. Therefore we can collect much more data and gain a deeper understanding of the operation of the system. Of course, as it is also the case with real-world experiments, two important issues have to be addressed in the simulation approach: bookkeeping and experimental design.

*Bookkeeping* addresses the management of the collected data, which includes the tracking of the executed experiments w.r.t. the used parameter configuration, software version, applied metrics, the observed data, and the analyses that had been performed on this data.

*Experimental design* is concerned with the higher-level question of which regions of the vast parameter space need to be explored next to confirm or disprove current hypotheses that we may have constructed to explain the previously observed data. These hypotheses often describe specific structures in the system dynamics across regions of the parameter space (e.g. phase changes), and we perform experiments to refine our description of these structures.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'03, 7, 2003, Melbourne, Australia.

Copyright 2003 ACM 1-58113-000-0/00/0000...\$5.00.

In a recent research project we explored the dynamics of a large agent system in which agents coordinate to solve a global problem (graph-coloring) by locally exchanging information. This exchange of information among the agents in our model is constrained by the environment in which the agents are embedded. Just as it is the case in many systems embedded in a physical environment, there is a delay in the transmission of any message (communication latency). In a separate paper [5] we discuss in detail the emerging phase structure of the agent system under varying problem, solution, and environmental parameters.

In this paper we use the distributed graph-coloring model as an example for the systematic simulation approach to the analysis of complex emergent system dynamics. After a brief description of the agent model, we present a software infrastructure that we developed to perform and manage hundreds of thousands of individual simulation experiments to “sweep” selected regions of the model’s parameter space systematically.

With our parameter sweep infrastructure we address the bookkeeping issue, but we still have to select manually the next configurations that we had to explore in the search for specific structures in the system’s dynamics. Furthermore, many of the experiments in a particular sweep were performed at configurations that were not “interesting”. Thus we wasted a lot of computing resources to map a selected region completely.

In this paper we present an agent-based system for the automatic design of experiments based on a fitness function that formally defines a level of interest and confidence at any point in parameter space (configuration) of the model based on the experiments that have been performed so far. The fitness function spans a multi-dimensional multi-peaked landscape in which the agents perform their search for the most interesting regions.

In the process of searching the landscape, the agents initiate additional simulation experiments to gather more data and thus increase the confidence in their findings. Thus the agents focus the expenditure of computational resources on simulations in interesting regions of the parameter space. In this paper we report results from initial experiments with a prototypical implementation of our agent-based search infrastructure that found previously uncharted phase changes in the graph-coloring model with only one sixth of the computational effort than a systematic sweep would have required. We estimate that a refined version of the distributed search would reduce the effort by one or two orders of magnitude compared to a full sweep and thus we could find structures significantly faster or search much larger spaces.

Why is this paper important for the agent community? The answer is twofold. First, it presents a new multi-agent algorithm for emergent optimization of the use of computational resources in a parallel search in a high-dimensional space. But secondly, the paper also presents a new tool for the analysis of complex emergent dynamics of large multi-agent systems with vast parameter spaces. Thus we use agent technology to perform agent research.

The remainder of this paper is structured as follows. In section two we present our generic software infrastructure for systematic parameter sweeps and in section three we briefly introduce the graph-coloring model that we explored with this infrastructure. In section four we specify a fitness function that formalizes our interest in creating new experiments based on previously collected data, in section five we present a multi-agent system that searches the resulting fitness landscape for interesting regions, and in section six we present our initial experimental results. We conclude in section seven.

## 2. SYSTEMATIC PARAMETER SWEEPS

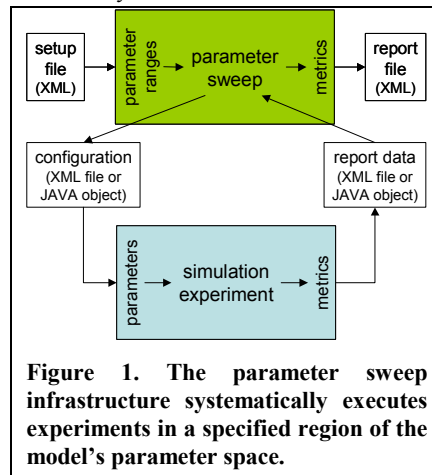
In several recent research projects we chose to explore the complex emergent dynamics of an agent system in a software simulation. In the course of these projects we developed a generic software infrastructure that enables us to configure and execute simulation experiments automatically for ranges of the various model parameters and to collect and archive observations of the system’s operation in specified metrics. Furthermore, the infrastructure may farm out individual experiments to be executed in parallel on multiple computers in a local network.

Figure 1 shows the general architecture of the parameter sweep infrastructure. We implement the respective simulation model to be configured either through an XML setup file or through an initialization data object that is provided to the constructor of the simulation experiment. With the XML setup we can manually configure and execute individual experiments outside the sweep infrastructure, while the initialization object permits the infrastructure to spawn new experiments efficiently without saving the data into the file system. For simulation models that are not able to receive an initialization object, maybe because they are not implemented in JAVA or because they are executed on a separate computer, the infrastructure configures experiments through the file system.

We observe the execution of a simulation experiment through various metrics. In our multi-agent systems, these metrics may trace data from the internal processes or communication activity of the individual agent, or they may automatically aggregate such raw data across agent populations or across time using statistical or other compression methods.

In the course of the increasingly detailed investigation of the dynamics of an agent system, we develop a wide range of such metrics, which may focus on various aspects of the system. Therefore, in any given parameter sweep experiment, we do not want to report all available metrics. Rather, as part of the configuration process of a simulation run, our infrastructure initializes only those metrics that we specified for this sweep.

This specification of the required metrics is part of the configuration of the parameter sweep, which is described in an XML setup file. We archive the file with the version of the simulation model and the reported data to be able to reproduce any experiment. The setup file also specifies the respective parameter values that are to be



explored. These values may either be fixed, an explicit enumeration of values, or they may be taken from an interval at regular steps. A future extension of the infrastructure may generate parameter values from the functional combination of values of other parameters.

For non-deterministic simulation models, the infrastructure executes a fixed number of experiments (replicas) with varying random seeds for all combinations of parameter values specified in the setup file of the parameter sweep. The data reported from these replicas form the basis of a Monte-Carlo analysis of the system dynamics at these configurations.

In the integration of a simulation model with our infrastructure we may choose whether the report data of the selected metrics in an individual experiment is stored into the file system or if it is handed to the infrastructure in a report object. To avoid running out of internal memory we often choose the route through the file system, especially for metrics that trace individual agent activity over time and thus generate lots of data. But if the data across multiple replicas should be processed further during the parameter sweep, we may need to return the report directly. The adaptive search for interesting regions reported in this paper is an example for such a dynamic post-processing.

The infrastructure generates a final report of all data collected across all experiments and saves it in XML format in the file system. The internal structure of nested XML elements in the report file reflects the structure of the region of the parameter space that had been explored in this sweep. Each level of nesting refers to a specific model parameter. Therefore subsequent analyses know the configuration of the model parameters that resulted in the respective data sets.

We archive the final report with the setup of the experiment and we use various filters to transform the data into formats used by our post-experiment analyses software. We currently use Mathematica, Microsoft Excel, and specifically tailored JAVA programs to analyze and graphically display our experiments such as those reported in [5].

We have used our parameter sweep infrastructure in several research projects to explore the dynamics of agent systems. Currently, we run experiments with an agent-based supply-network simulation implemented in the Swarm package, a JAVA implementation of a distributed formation flying mechanism for robotic planes, and we explore the emergent dynamics of a swarming path planning algorithm.

### 3. AN EXAMPLE SIMULATION MODEL

In the following we present a model of a population of agents collectively solving the graph-coloring problem. In a recent research project we chose this model to analyze the emergent dynamics of multi-agent coordination in resource-restricted environments. Using our parameter sweep infrastructure, we explored the parameter space of the model and found distinct phases of system-level behavior. In this paper we demonstrate the use of an agent-based experimentation infrastructure that searches

**Table 1. Graph-Coloring Model Parameter**

Model Parameter	Description
N	Number of Nodes in Graph
K	Number of Neighbors per Node
G	Number of Colors Available
GC	Mechanism to Construct Graph
AL	Probability to Activate Color Decision
CL	Time of Message Transfer to Neighbor
MD	Constraint on Change of Local DoC
CS	Mechanism to Select Color

efficiently for phase changes in the graph-coloring model by automatically generating simulation experiments.

### 3.1 The Model

The graph-coloring problem is a fundamental challenge problem to which many other coordination tasks may be reduced. In this paper as well as in more detail in [5] we analyze the dynamics of a particular approach to this problem that has been

proposed by researchers at the Kestrel Institute in [7].

In its general form the graph-coloring problem seeks to assign one color out of a globally fixed set of size  $G$  to each node in an undirected graph so that the number of edges that connect nodes of the same color is minimized. Soft, real-time distributed graph coloring assigns an agent to each node in the graph that needs to be colored. Thus there are  $N$  agents (one for each node in the graph) in the multi-agent system and, according to the undirected edges among the nodes, each agent has a number of direct neighbors to whom it communicates changes of its color.

In our experiments we typically considered random graphs in which each node has a fixed number of neighbors ( $K$ ). We implemented multiple ways (indexed by the  $GC$  parameter) of sequentially constructing such graphs, each of which results in graphs with specific characteristics. For instance, in one graph construction mechanism, we randomly distribute the nodes on a unit square and assign each node those  $K$  nearest neighbors that do not yet have their complete set of neighbors assigned. This mechanism typically produces graphs that may be embedded in low-dimensional spaces. Another mechanism selects randomly among those nodes that have the least number of neighbors assigned already and connects the chosen one to another of these most incomplete nodes. This mechanism tends to yield graphs with a very short characteristic path length. In this paper we present how our agents automatically found a phase change in graphs constructed by one mechanism after we had manually confirmed the phase change in graphs constructed by another mechanism.

Any agent in the random graph cyclically performs steps in a local hill-climbing mechanism in which it uses various rules (indexed by the  $MD$  and  $CS$  parameters) to select a new color that reduces the local Degree of Conflict ( $DoC$ ) metric. The local  $DoC$  is the node's main performance metric. For any (assumed or real) color of the node, it is the number of neighbors that share this color divided by the overall number of neighbors of this node. The rate at which the agent reconsiders its color choice is determined by the global Activation Level ( $AL$ ) parameter. At fixed intervals the agent decides probabilistically whether to execute another hill-climbing step. The  $AL$  parameter sets this probability.

If the color that the agent selects is different from the current color, the agent communicates the change to all its direct neighbors in the graph. In our model we delay the arrival of the change messages at the neighboring nodes by a globally fixed time specified in the "Communication Latency" ( $CL$ ) parameter.

Table 1 lists all available model parameters that may be varied in the exploration of the emergent system dynamics. We implemented the graph-coloring model and integrated the simulation with the parameter sweep infrastructure.

One of our metrics, called Option Set Entropy (OSE), estimates the guidance in the currently available local information as it is used by the agent’s decision process. The OSE is the normalized Shannon (or Information) Entropy [10] applied to the probability of an agent’s selecting a particular color in a decision cycle at a specific point in time. This probability is determined by the currently known colors of the node’s neighbors, the constraints on the change of the local DoC, and the chosen color selection mechanism.

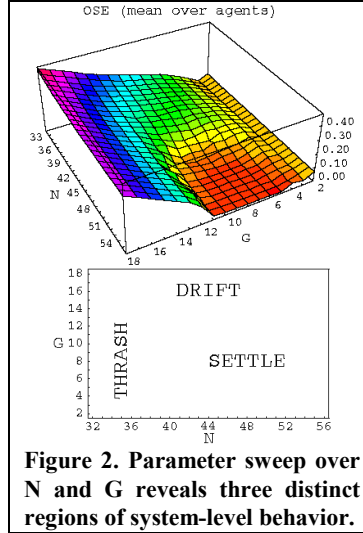
### 3.2 A Phase Change

As we report in [5], in this graph-coloring model we find three main regions of system-level behavior that drastically influence the problem-solving performance of the agent population. Figure 2 maps these three regions in a space spanned by the two parameters N (number of nodes) and G (number of colors) observed through the OSE metric. We find two regions of good performance, one, in which the system performs better than a random color selection process and another where the problem becomes so easy that the performance of the random process asymptotes towards the agent system performance.

The third region is found at low values of N and G. In this region the graph-coloring problem is very hard, because there are only a few colors available and most nodes in the graph are direct neighbors (N approaches K). Therefore, it is critical for the individual agent to use correct information when making its next color choice. But in the combination of the fixed decision rate (AL parameter) and the delay of messages (CL parameter), at some point in parameter space the system falls into thrashing behavior. At this point too much false information has made its way into the decision process and agents change their node’s color to resolve a conflict that had been solved by one of their neighbors already and thus they recreate the conflict.

Figure 3 shows the result of a parameter sweep across values of the AL parameter while keeping all other parameters fixed. As we plot measurements from individual experiments (rather than the mean over all replicas as in Figure 2), we discover that the transition into thrashing behavior is sudden rather than gradual and accompanied by a robust overlap region, where we find systems attracted either to the thrashing or to the benign behavior. The overlap is due to a so-far unknown graph characteristic that selects the performance attractor in this critical region. As we will argue in the next section, the existence of such an overlap region helps us to identify the location of the phase change solely based on measurements from one configuration rather than having to compare neighboring configurations in parameter space.

Determining the location of the phase change is very important in the deployment of the agent system in a real-world scenario. On



**Figure 2. Parameter sweep over N and G reveals three distinct regions of system-level behavior.**

the one hand, the decision rate determines the speed at which a problem is solved and typically we are required to find a solution as fast as possible. On the other hand, if the system falls into thrashing because the decision rate is too high for a given problem and the communication latency of the specific environment, the system will not find a good solution. Thus, for a given deployment scenario we will have to define the globally largest AL parameter for which none of the typically encountered problems leads to thrashing.

We may find this critical value by specifying what graph structures and problems are typically encountered in a specific deployment scenario and what the expected communication latency will be and then systematically sweeping the parameter space until we find the transition into thrashing. But

this approach may take a long time and waste a lot of computing resources before it finds the interesting region. Rather, we propose to apply an automated search of the parameter space that efficiently hones in on the phase change guided by a fitness function that represents our interest in an overlap region.

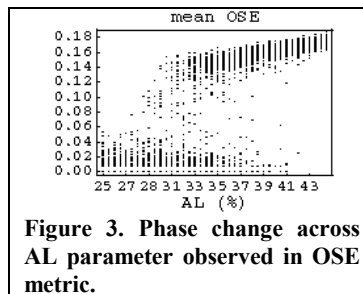
### 4. FORMALIZING INTEREST

The first step in an automated search for interesting structures in the observed dynamics of a simulation model is to specify formally the level of interest generated by a set of data points sampled at a single configuration in parameter space. This function then becomes the fitness function, which guides the search of our agents in the parameter space of the underlying simulation model.

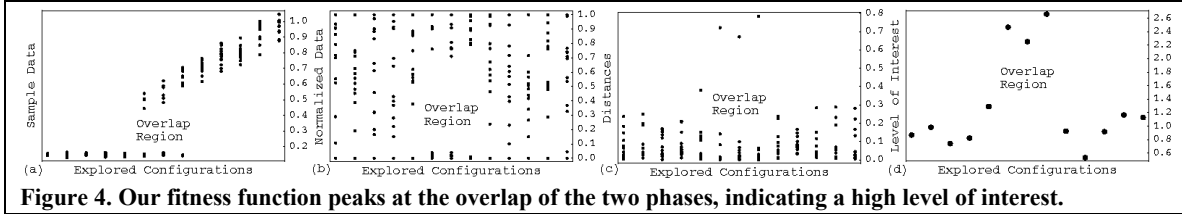
In our example of the graph-coloring model we are interested in finding the location of the transition from benign into thrashing behavior. We observe this transition in many different metrics. In the OSE metric the thrashing behavior results in high values while in the benign region the observed values are significantly lower (see Figure 3). The non-deterministic nature of the model results in a variation of the actually observed values around a statistical mean determined by the phase in which the respective experiment falls.

In the graph-coloring model chosen as an example in this paper we find that at the location of the phase change there is a small region of coexistence in which some critical characteristic of the specific problem graph decides into which phase a specific experiment will fall. Previous parameter sweep experiments over extremely long periods of simulation time confirmed the robustness of this overlap.

We specify a fitness function that for a variable number of OSE observation points provides a single level-of-interest value, which is high, if the provided data points seem to be part of two clusters and low if the values could be explained by just one random distribution. Obviously, as we operate on samples of random functions, our confidence in our fitness evaluation grows with the number of samples taken into account.



**Figure 3. Phase change across AL parameter observed in OSE metric.**



**Figure 4. Our fitness function peaks at the overlap of the two phases, indicating a high level of interest.**

The following procedure determines the fitness of a configuration based on a set of observed OSE values from multiple runs at a single location in parameter space: 1) normalize the observations so that the smallest value maps to zero and the largest value maps to one, 2) sort the normalized set, 3) consider the set of distances between neighboring values in the sorted set of normalized observations. In a set with samples from two clusters we expect to find a large number of similar distance values from samples within the respective cluster and very few large distance values as we compare samples from different clusters. The statistical function Coefficient-of-Variation applied to the set of distance values is high if a few values stand out and low if the set is homogeneous. The function is defined as the standard deviation divided by the mean of a set of data points.

The plots in Figure 4 illustrate the application of the fitness function to a set of artificially constructed sample values from a range of configurations. The first plot (4.a) shows the observed data values. At the left side the observed system falls clearly into phase one, which is characterized by a low constant mean (0.15) and a small variance (0.0075). The samples on the right side of the plot are all taken from a second phase with a linearly increasing mean and a larger variance (0.05). For configurations in the center of the plot we find both phases overlapping. At each configuration we took 13 samples.

The second plot (4.b) shows the normalized sample values. For each configuration we proportionally scale the data points so that the smallest maps to zero and the largest maps to one. For samples that are taken from only one distribution, the shared variance results in an approximately even spread of data points. Samples from two distributions with a different mean should be less homogeneous. The normalization step is necessary since the mean and the variation of the two clusters shifts as we move across the parameter space, but we need a fitness function that is independent of the range of values collected by the chosen metric. The OSE metric is already limited to the interval  $[0,1]$ , but other metrics may provide samples from a different domain.

In the third step of our fitness estimation process we sort the normalized data points and compute the distance between direct neighbors. We need at least three samples for this step. The plot 4.c shows the distance values. An even spread of normalized values results in lower to medium range distances, while increased heterogeneity leads to a mix of very small and very large distance values.

As we are interested in the degree of heterogeneity of our sample data points (overlap region), we compute the statistical Coefficient-of-Variation across the distance values. As expected, this measure peaks for the configurations in the overlap region (plot 4.d). Thus, our fitness function is the Coefficient-of-

Variation applied to the distances among sorted members of the normalized set of observed OSE values.

In the following section we present a system of autonomous agents that coordinate their exploration of a given region in parameter space based on this fitness function to find the interesting region of phase change without wasting computational resources for simulations in regions of low interest.

## 5. AGENTS GUIDING SEARCH

In Figure 2 we explored a two-dimensional parameter sub-space, but the graph-coloring model has nine mostly independent parameters. The traditional approach to searching such a space is a predefined search, based on an experimental design. Factorial designs that exhaustively sweep the relevant ranges (for example, using a tool such as Drone [2]) quickly become computationally prohibitive, while designs such as Latin Squares that combine the exploration of different factors in a single run are blind to interaction effects. The problem is exacerbated when the phenomenon we wish to detect (such as phase coexistence) requires multiple simulation runs at each point in parameter space. APSE (Adaptive Parameter Sweep Environment) is a distributed, agent-based search mechanism that has the potential of significantly reducing the effort involved with such a search.

The fitness function presented in the previous section assigns a single value (level of interest) to each configuration in the model's parameter space. The resulting fitness landscape is noisy, has many peaks and ridges, and it is initially not known as it only emerges through repeated execution of simulation experiments at a configuration.

Fitness landscapes of this size and nature are best searched in parallel, using heuristics such as genetic algorithms. In this paper we propose a fundamentally similar approach: local hill climbing of individual agents combined with a global exchange of guidance information. An immediate advantage of the agent approach is the potential for distribution of the search over multiple processors.

Each Searcher agent represents one thread of the parallel search for peaks in the fitness landscape spanned over the model's parameter space. In a particular search experiment, we deploy as many Searchers as we can afford, given our computational resources. Since each agent will execute simulation experiments with the underlying model, we may only be able to deploy a few agents per processor.

The population of Searchers creates and searches the fitness landscape at the same time. They are deployed in a supporting environment that collects the samples taken by the agents at the various locations (configurations). An individual sample is very expensive computationally and therefore it is important that no effort is duplicated.

### 5.1 Simulate or Move?

A Searcher has two objectives. For any location in parameter space that it occupies it seeks to increase the confidence in the local fitness estimate by executing additional runs, but at the same time it wants to be located at a configuration that has a high fitness. In the tradeoff of these two objectives across the agent population a resource-aware search emerges.

Figure 5 illustrates the different situations in which an agent may find itself. It may be located at a configuration for which no samples have been collected yet. In this case, nothing is known about this location’s fitness and the agent must execute a minimum number of simulation experiments. At the other end of the spectrum the agent may find itself at a configuration with a sufficiently high confidence (maximum number of samples). Again the agent has no choice. It may not execute more experiments, and so it must find itself a different location.

If the agent’s location has a confidence higher than the minimum requirement but lower than the maximum, the Searcher considers the location’s current fitness estimate to make its choice. In this case the general rule is that the higher the fitness estimate the higher is the likelihood that the agent adds another sample to the local collection of data points. Therefore more experiments are executed in regions of high interest, while in less interesting locations fewer resources are spent. The agent applies the general rule probabilistically with a weighted coin-flip and therefore even low-fitness regions have a non-zero chance of being explored.

### 5.2 Simulate!

If a Searcher agent decides to sample the dynamics of the underlying simulation model to increase the confidence of the fitness estimate at its current location, it uses the same interfaces to the model as the parameter sweep infrastructure. It creates a configuration object or file that includes a random seed that has not been used before at this location. Then it triggers the execution of the model, which either returns a report object or dumps its results into the file system.

The fitness estimate requires that the emergent dynamics of the simulation run be characterized by a single value. In our experiments with the distributed graph-coloring model, we used the Option Set Entropy (OSE) metric averaged over all agents and over a period of time towards the end of the simulation for this characterization. Other metrics may be more informative in other models, depending also on the system-level feature that we consider interesting in the particular search.

After the simulation run is complete, the Searcher agent extracts the characterizing value from the report data and adds it to the collection of sample values in the search environment (database). Then it computes the fitness estimate for the new sample set. At this point the agent returns to its simulate-or-move decision.

### 5.3 Move!

Distributed optimization mechanisms generally execute a local hill-climbing mechanism (deterministic or probabilistic) for the individual representations of the current solution set. For

instance, an agent in the distributed graph-coloring model considers all its options (available colors) and it prefers to select one that reduces its local Degree of Conflict with its neighbors. Ants in a food-foraging mechanism sample pheromone concentrations in their local neighborhood and tend to move towards higher concentrations.

A Searcher agent in our dynamics finder mechanism may only move to neighboring configurations in the discretized parameter space of the underlying model. At the outset of a search experiment we specify the range of parameter values accessible to the Searcher population and for those parameters that we permit to change, we also specify the step size of the change. This prior knowledge reduces the complexity of the agents’ search task.

An agent’s movement decision is always a probabilistic choice across the set of neighboring locations. The agent first integrates various driving forces into selection probabilities for the respective location and then spins a roulette wheel with segment sizes proportional to these probabilities.

The probability of the selection of a neighboring location is a combination of local and global information. The local information is the fitness of the location if it has been established already or zero if the configuration has not been sampled yet. This information is provided by the agents’ environment, which keeps track of the sample data accumulated at the various locations. Initially it is very unlikely that at any of the neighboring locations the minimum confidence has been built and thus there is no guidance for the agent to be found in this local information.

Global attracting forces between the Searcher agents temper the local hill climbing. All agents at locations with a higher fitness attract an agent that currently executes a movement decision. This attraction is intended to focus the search of the population on one region of high fitness rather than spreading the agents out too thin.

The attracting forces are interpreted as vectors that point to the respective attracting agent and whose length equals the difference in the fitness. The sum of the vectors to all attracting agents is the global information that increases the selection probability of the neighboring locations whose direction is similar to this vector at the expense of those that would lead the agent into the opposite direction.

The probabilistic integration of local and global guidance information leads the agent to climb towards local optima in the fitness landscape but, if the local optimum is not as high as other optima discovered somewhere else, the agent may abandon the local ascent for a more promising pasture. The initial absence of fitness information leads to globally guided agent movement. But over time fitness at more and more locations is mapped and the individual agents focus more on local information.

### 5.4 Comparison with Previous Research

APSE adapt the search of the parameter space dynamically during the simulation process. Antecedents to this work include exploratory modeling at RAND [1], the EvCA (Evolving

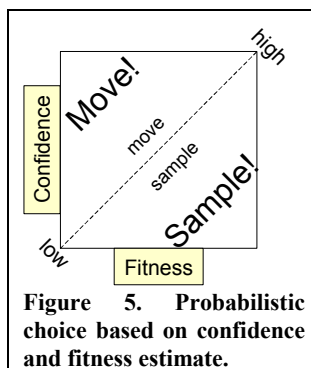


Figure 5. Probabilistic choice based on confidence and fitness estimate.

Cellular Automata) group at the Santa Fe Institute [6] and Miller at CMU [8].

The RAND work outlines the potential for the sort of exploration we are conducting, but does not solve the critical problem of developing fitness functions that capture dynamic phenomena of interest.

Miller uses evolution over parameter spaces to verify models in the social sciences, seeking parameter values that break a model. In contrast, we are searching the parameter space against two criteria: 1) What are the bounds on performance that a given approach can achieve? 2) Where might there be interesting discontinuities in behavior (e.g., phase transitions) that require further study?

The EvCA group’s use of evolution is closer to ours. They evolve update rules for one-dimensional cellular automata to find rules that will let the automaton compute a given function of its initial state (e.g., setting all cells equal to the state of the majority of the initial cells). We are searching for a much more complex structure, and the target of our search will be much more difficult to capture.

The key to adaptive search is defining an appropriate fitness function against which to evaluate successive results. This requirement is not onerous if we are searching for performance bounds, since we simply use adaptive search to drive the performance as high (or low) as it can, and examine the slope of the performance landscape to detect leveling-off. In searching for discontinuities, the appropriate fitness function can be much more elusive. Such discontinuities are traditionally recognized by visual inspection of plots of experimental results (e.g., Figure 3). Our experiments show the promise of using the entropy of the normalized distance between sample points.

## 6. EXPERIMENTAL RESULTS

Figure 6 shows the result of a preliminary experiment with a first untuned implementation of the Searcher population and its support infrastructure.

In previous parameter sweep experiments we found a phase change into thrashing for graph-coloring problems in which the graph was constructed with the MinimumNeighbor method, which tends to produce graphs with very short characteristic path length and which could only be embedded in very high-dimensional spaces. At this point we were wondering whether lower-dimensional graphs (e.g., graphs on a 2D surface) would trigger the same phenomenon. Thus we implemented a graph construction method that randomly positioned the  $N$  nodes on a unit-square surface and then sequentially connected each node with its  $K$  nearest neighbors that had not yet  $K$  neighbors themselves.

Finding the answer to our question through manual sweep experiments would have taken up too much computational and human resources that we needed to use in other aspects of the project. So we put our Searcher agents to work in the background.

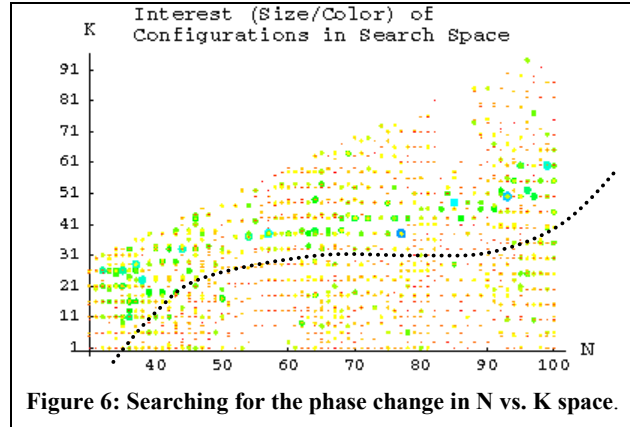


Figure 6: Searching for the phase change in  $N$  vs.  $K$  space.

We asked our agents to map the location of a phase transition on a  $N$  vs.  $K$  region of the model that would have required 121810 simulation runs in a systematic exploration. The plot in Figure 6 shows the level of interest divided by the degree of confidence after 20046 runs. We clearly find high-interest regions with only one-sixth of the effort. Subsequent sweep experiments of smaller parameter spaces confirmed the existence of a phase change at the locations of high interest.

Given the experimental nature of our implementation and the lack of any fine-tuning of the agent decision process (weights etc.) we expect to achieve at least a ten-fold reduction in effort with more “serious” implementations.

## 6.1 Future Research

Our current results open two major directions for future research. On the one hand we see a number of ways in which the distributed agent-based search mechanism may be improved. On the other hand, we are in the process of developing concepts by which the agents may automatically balance the computational load on a network of processors as they execute massive simulation runs.

Currently, the movement of the agents through the parameter space is only guided by the fitness and confidence values at the various configurations. We believe that a major performance improvement may be gained if the agents take existing domain (model) knowledge into account. For instance, two agents searching for a phase change should be attracted towards each other, if the values that they observe in the reports of the simulation runs (here OSE metric) are very different at the two locations. The closer the agents are in parameter space, the stronger is the attraction for the same observed difference, since we may suspect at least a drastic change in the system’s dynamics, which might be accompanied by an overlap region. Other improved fitness metrics may be defined on neighboring sets of locations rather than just on one model configuration.

The decision of an agent to move or to run another simulation experiment is currently only guided by the observed fitness and confidence values, but it does not take into account the number of simulations currently under way at the local processor of the agent. We envision a future enhancement that includes this information into the agents’ decision process. Furthermore, we believe that permitting the agents to switch their processing node may further enhance the balancing of the computational load across the network.

## 7. CONCLUSION

In this paper we presented two related tools that we developed to support our systematic exploration of complex emergent system dynamics of multi-agent systems. The first tool – a parameter sweep infrastructure – automatically configures and executes simulation experiments across specified ranges of the various model parameters and it aggregates the reports of metrics that had been selected for the respective sweep. Supported by this tool we

analyzed several simulation models of multi-agent systems for various applications.

Emergent system-level dynamics of large multi-agent systems are often very complex even though the individual agent may still be simple in its decision and interaction processes. For instance, in an agent system for distributed graph coloring we discovered qualitatively distinct phases of system-level behavior and a robust overlap region in the transition from one phase to another. Detailed results from this analysis are reported in [5]. In this paper we use the agent model as an example for the automated search for specific emergent dynamics in the vast parameter space of a simulation model.

In the development of our second agent research support tool presented in this paper we chose a multi-agent systems approach. Thus we use agents to research agents! Manually searching a model's parameter space for the location of interesting dynamical features may be a time consuming process if the formal dependency of the dynamics on the model parameters is not known. Even with our parameter sweep tool we could spend days in large collections of simulation runs only to find "boring" dynamics.

A formal estimate of the level of interest for a set of observations collected from multiple simulation runs is the key to the implementation of an automated search process that guides the use of computational resources to regions in configuration space that promise interesting dynamics. In this paper we present an example of such a fitness function that peaks if the observed dynamics indicate an overlap region of two system attractors.

The fitness function spans a noisy, multi-peaked and multi-dimensional fitness landscape that is searched best in parallel. We deploy a population of Search agents in an infrastructure that tracks simulation results. At any point in time an agent is "located" at a specific configuration of the underlying model and it decides whether to move (change the model configuration) or if it should initiate another simulation run with the current configuration to improve the confidence in the fitness evaluation. The agent is most likely to move it is at a high-confidence/low-fitness location.

The movements of the agents tend to take them towards regions of higher fitness. Collectively, the agents minimize the use of computational resources for simulation runs of low interest and maximize those for locations of high interest. In a prototypical implementation we have been able to find previously unknown locations of phase change in the graph-coloring model with one sixth of the effort that would have been required to completely sweep the parameter region that we had considered.

Our tools focus on two important issues that need to be addressed when following the simulation approach in systems' research. With a computer simulation model we are able to explore the emerging system dynamics in many more scenarios than we could with experiments with real-world systems. Thus it is very important that we have tools that support bookkeeping of the experiments and that help us to focus our resources on interesting regions in parameter space.

Our multi-agent solution to the distributed search problem addresses a novel problem: searching vast spaces with only

limited resources available to evaluate the quality of the individual solution. Given the restrictions of this problem, the agents combine a local hill-climbing mechanism for solution optimization with a global attraction mechanism to coordinate their resource usage.

## ACKNOWLEDGMENTS

This work is supported in part by the DARPA ANTS program, contract F30602-99-C-0202 to Altarum, under DARPA PM Vijay Raghavan. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government. We gratefully acknowledge the cooperation of Stephen Fitzpatrick at Kestrel Institute for consultations on their graph-coloring model for sensor allocation.

## 8. REFERENCES

- [1] S. Banks and J. Gillogly. Exploratory Modeling: Search through Spaces of Computational Experiments. In *Proceedings of Third Annual Conference on Evolutionary Programming*, pages 353-360, World Scientific, 1994.
- [2] T. C. Belding. Drone 1.01 User's Guide. 1996. HTML, <http://pscs.physics.lsa.umich.edu/Software/Drone/doc/drone.html>.
- [3] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.
- [4] S. Brueckner. Software Demonstration in Illustration to the Paper: Ant-Like Missionaries and Cannibals - Synthetic Pheromones for Distributed Motion Control. In *Proceedings of Autonomous Agents 2000*, 2000.
- [5] S. Brueckner and H. V. Parunak. Information-Based Phase Changes in Multi-Agent Coordination. In *Proceedings of AAMAS'2003 (submitted)*, 2003.
- [6] EvCA Group. Evolving Cellular Automata. 2000. Web Site, <http://www.santafe.edu/projects/evca/>.
- [7] S. Fitzpatrick and L. Meertens. Soft, Real-Time, Distributed Graph Coloring using Decentralized, Synchronous, Stochastic, Iterative-Repair, Anytime Algorithms: A Framework. Technical Report KES.U.01.5., Kestrel Institute, 2001.
- [8] J. H. Miller. Active Nonlinear Tests (ANTs) of Complex Simulation Models. *Management Science*, 44(6 (June)):820-30, 1998.
- [9] S. Rasmussen and C. L. Barrett. Elements of a Theory of Simulation. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Editors, *Advances in Artificial Life. Third European Conference on Artificial Life, Granada, Spain, June 4-6, 1995.*, vol. 929, *Lecture Notes in Artificial Intelligence*, Springer, Berlin, 1995.
- [10] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. Urbana, IL, University of Illinois, 1949.