

# Information-Driven Phase Changes in Multi-Agent Coordination

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

Large systems of agents deployed in a real-world environment face threats to their problem solving performance that are independent of the complexity of the problem or the characteristics of their specific solution mechanism. One such threat is the degrading of the quality of agent coordination mechanisms when faced with delays in the flow of critical information among the agents introduced by communication latencies. In this paper we demonstrate in a simple model of locally interacting agents that the emerging system-level performance may degrade very suddenly as the rate of individual decision making increases against the availability of up-to-date information. We present results from extensive simulation experiments that lead us to select a locally accessible metric to adapt the agent's individual decision rate to values that are below this phase change. Given the generic nature of the coordination mechanism that is analyzed and the information-theoretic metric, the adaptation mechanism may increase the deployability of large-scale agent systems in real-world applications.

## Keywords

Multi-Agent Coordination, Emergent Phase Structure, Adaptation and Learning, Simulation Experiments, Tools and Methods.

## 1. INTRODUCTION

In many real-world applications (e.g., manufacturing control, intelligent sensor networks, fine-grained robotics) multi-agent systems comprise many entities with sometimes severely restricted resources (i.e., processing, memory, bandwidth). These entities often exchange information to coordinate their individual activities and to achieve meaningful system-level behavior.

The interactions of the agents transmit information that influences the agents' decision processes. Thus the performance of the individual agent as well as the whole agent system directly depends on the timely and accurate flow of information. But, in systems of resource-limited agents this requirement may not be met since communication requires time and may be noisy.

A simple graph-coloring model enables us to explore systematically the impact of the speed of the information flow on the quality of the decision processes in a large-scale agent system. The complex emerging dynamics of the decision processes in-  
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.

clude a robust phase change in the system performance driven by the latency in individual communications. Knowing the location of this phase change is very important in the tradeoff between the speed in which a solution is achieved by the system and the quality of the solution.

In deployed systems the communication latency is determined by many factors and it may vary over time as well as over the space inhabited by the agents. Furthermore, we seldomly are in a position to control the speed of the information flow and thus our solution-time vs. solution quality tradeoff has to be determined by the rate of the individual agent's decision process.

Based on the analysis of the phase structure of the system's performance and its reflection in metrics that are accessible to the individual agent we present a generic local control mechanism that permits the agents to determine their currently optimal decision rate to achieve a good solution in the shortest time possible. Our local control mechanism induces meta-level coordination in that it observes the performance of the system through local indicators and guides the individual agent activity for the greater good. These are the basic characteristics of the cognitive functions of introspection and learning, realized in our case through emergent sub-symbolic reasoning.

The remainder of this paper is structured as follows. In section two we present our agent-based graph-coloring model. In section three we discuss our experimental approach and a software infrastructure that we developed to explore large regions of a model's parameter space automatically. In section four we explore the complex dynamics and the emergent phase structure of the graph-coloring model. In section five we introduce a generic local control mechanism that realizes system-level introspection and learning and we demonstrate the improvement of the performance achieved in the graph-coloring example. We conclude in section six.

## 2. DISTRIBUTED GRAPH COLORING

The graph-coloring problem is a fundamental challenge problem to which many other coordination tasks may be reduced. In its general form it 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.

We directed our attention to graph-coloring in a multi-agent coordination context in the DARPA ANTS program, where the team from Kestrel Institute explored the dynamics of a distributed sensing and tracking system controlled in real-time by agents that are severely restricted in their processing and communication capabilities [8]. In the application, an agent corresponds to a small robotic radar sensor that can light one of three segments of the sky at any time. It takes at least three nearby sensors to focus on the same region of the airspace above to track an object. The graph-

coloring problem in this case maps nodes to regions in space in which at least three sensors overlap and edges connect regions that share the same sensor. The number of colors relates to slots in a repetitive schedule of the sensors that need to be coordinated to track objects discovered by at least one sensor.

Our research in this program was concerned with the general dynamics of distributed resource allocation. Phrased in this context, each node in the graph represents a task, each color represents a resource, and an edge between two nodes (tasks) indicates that a single resource cannot service them simultaneously. Thus, graph-coloring models (spatio-)temporal coordination tasks that are very common in multi-agent applications in general.

In this paper we do not present a new approach to solving the graph-coloring problem. Rather we use the distributed solution algorithm proposed by the Kestrel team to demonstrate the impact of delayed information flow on the solution quality and speed. Soft, real-time distributed graph coloring [5] 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 resulted in graphs with specific characteristics. For instance, in one graph construction mechanism, we randomly distributed the nodes on a unit square and assigned each node those  $K$  nearest neighbors that did 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. All experiments reported in this paper were conducted with graphs constructed by this mechanism, but the observations and conclusions also hold for other GC parameters.

In the chosen graph-coloring algorithm, any agent cyclically decides whether to reconsider its color choice or not. The so-called activation decision is taken non-deterministically with a fixed probability, which we call the agent’s Activation Level (AL). The randomized activation of the nodes in the graph is intended to (statistically) prevent simultaneous color changes among neighboring nodes.

Figure 1 sketches the basic decision process that a node executes if it is activated. At

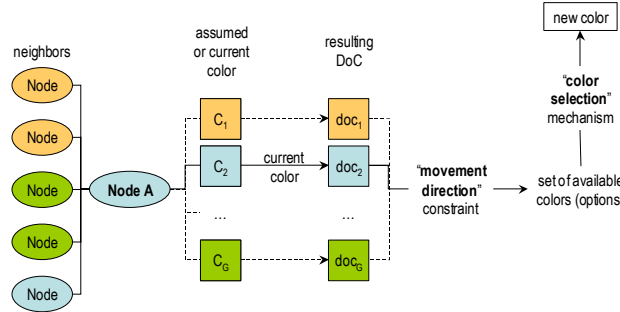


Figure 1. Node A's agent's color selection process.

first, taking into account the currently known colors of the node’s neighbors, it computes the local Degree of Conflict (DoC) for each available color. 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 agents attempt to solve the coloring problem across the graph by individually minimizing this local metric.

After determining the expected consequences of each available option, the agent selects its color in two steps. First, it may reduce the option set depending on the permitted change in the local DoC. The “Movement Direction” (MD) parameter determines, whether only colors that improve (reduce) the local DoC (MD=“OnlyUp”), colors that do not increase the conflict (MD=“LateralAndUp”), or all colors (MD=“Any”) may be selected. The agent keeps its current color if no color meets the respective constraint.

In a final step the agent selects its new color from the remaining option set. We experimented with three different mechanisms determined by the model’s “Color Selection” (CS) parameter. Either the agent selects the next color randomly from the reduced option set (CS=“Random”), or it chooses probabilistically with the individual probabilities inversely proportional to the expected local DoC (CS=“Roulette”), or the agent chooses randomly only among the colors that promise the largest improvement in the local DoC (CS=“Best”).

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.

Our model also includes a simple noise process. Any node has a fixed probability to “fail”. In each cycle, independent of whether the color decision process is activated or not, the node may randomly select any color with a probability specified in the “Reset Probability” (RP) parameter.

Table 1 lists all available model parameters that may be varied in the exploration of the emergent system dynamics. In the following section three we present a software infrastructure that supports a systematic exploration of this parameter space.

### 3. SYSTEMATIC PARAMETER SWEEPS

Even though the individual decision processes and agent interactions may be very simple as they are for instance in the distributed graph-coloring model, formally analyzing the complex emergent dynamics of a large system of these agents quickly becomes intractable. But it is exactly the dynamics in the large that primarily interest us.

Constructing a software simulation of the respective model and then experimentally

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 Random Graph
AL	Probability to Activate Color Decision Process
RP	Probability to Randomly Change Color
CL	Time of Message Transfer to Nearest Neighbor
MD	Constraint on Permitted Change of Local DoC
CS	Mechanism to Select Color from Option Set

exploring selected regions of the model’s parameter space provides a solution to this dilemma that is chosen these days by many researchers. But there are many pitfalls to be avoided when experimenting with complex non-deterministic models with large parameter spaces. For instance, the introduction of artifacts by the system’s random-number generator has been discussed extensively. Another important issue is finding interesting regions in vast parameter spaces while only having limited time and computational resources available. We propose an approach to this issue in another paper [2].

Another non-trivial problem in following the experimental approach is plain bookkeeping. Mapping out the phase structure of the distributed graph-coloring model requires the setup, execution and analysis of hundreds of thousands of experiments – a task that cannot be achieved manually.

We developed a parameter sweep infrastructure that enables us to explore the dynamics of a given simulation model efficiently. The infrastructure automatically sets up experiments either through XML configuration files in the file system or through JAVA objects directly handed over to the simulation thread. The use of XML configuration files permits us to run individual experiments manually at any time and it also allows us to farm out experiments on to any available computer in our network that has access to the fileserver.

After a simulation experiment has been completed successfully, it stores its results in an XML report file in the file system. In any given parameter sweep we may be interested in a number of metrics such as traces of individual agent states or statistical measures aggregated over agent populations and many simulation cycles. But in general we do not require all possible metrics to be recorded at all time. Therefore we specify with the configuration of a parameter sweep, which of the measurements should be reported.

Reporting our metrics in XML format enables the parameter sweep infrastructure to aggregate the results of many experiments efficiently into one report whose structure reflects the dimensions of the region of the parameter space that had been explored. Additional filters applied after the completion of the sweep translate the XML file into any format required by our chosen analysis software. We currently use Mathematica, Microsoft Excel, and specifically tailored JAVA programs to analyze and graphically display our experiments such as those reported in this paper.

The integration of the parameter sweep infrastructure with a new simulation model is very simple. We configure a parameter sweep using a meta-level XML setup file, which specifies which parameters are to be “swept” and how they are communicated to the underlying simulation software. We also separate the collection of our report data from the simulation itself similar to the Swarm package’s Observer Swarm concept [7].

We have used our parameter sweep infrastructure in several research projects to explore the dynamics of agent systems. Currently, it supports 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 the emergent dynamics of a swarming path planning algorithm.

## 4. INFORMATION-DRIVEN PHASE CHANGES

The dynamics of the distributed graph-coloring system may be explored in many dimensions and they are reflected in various

local and global metrics. In the following we map the general layout of the dynamics along the major dimensions and then focus on the sudden transition from good to bad performance.

### 4.1 Parameter Space

Table 1 lists the parameters of our agent model of distributed graph coloring. These parameters may be grouped into problem parameters, solution parameters, and environmental parameters.

**Problem parameters** specify the particular graph-coloring task. These include the number of nodes (N), their arrangement into an undirected random graph (K, GC), and the number of colors available (G). Varying these parameters while keeping the ones of the other groups fixed explores the algorithm’s performance under various problems.

**Solution parameters** configure the operation of the distributed graph-coloring algorithm. This group includes the permitted change of the local DoC (MD), the method of selecting a new color (CS), and the average decision rate of a node (AL). Varying these parameters explores the performance of different classes of the solution approach for the same problems. The parameters of the solution algorithm are the ones that need to be adapted in the deployment of the agent system to solve a certain range of graph-coloring problems in a particular environment.

**Environmental parameters** define the conditions under which a particular solution mechanism attempts to solve a given problem. We model the delay in communication among the nodes (CL) and the rate of node failure (RP). Exploring these dimensions helps us understand the impact of a particular deployment scenario on the solution of a given coordination problem.

### 4.2 Metrics

The agents in the distributed graph-coloring problem interact locally to solve a global problem without being explicitly aware of their joint task. Therefore we consider the global solution to be emergent.

The emerging dynamics of the solution mechanism applied to a specific problem in a particular environment may be observed in a variety of metrics. These metrics may be based on processes inside the individual agent, observations across the agent population at a given point in time, or aggregations over time.

The primary performance metric is the **global Degree of Conflict**, which measures the quality of the solution to the graph-coloring problem at a particular point in time. Similar to the local

**Table 2 . Additional Graph-Coloring Model Metrics**

Metric	Description
Agent Activation Count (AAC)	Number of Agents Activated in a Simulation Cycle
Changed Color Portion (CCP)	Percentage of Nodes that have Changed their Color in the Last Cycle
Color Change Entropy (CCE)	Information Entropy in Distribution of Recent Color Change Events in the Graph – Computed over whole Graph and Multiple Cycles
Graph Characteristics (GC)	Watts' Graph Metrics - Characteristic Path Length and Clustering Coefficient (REFERENCE)
Random Option Size (ROS)	Number of Colors among which an Agent chooses Uniformly Random
Time to Solution (TTS)	Estimate of the Number of Cycles it takes for the Network Dynamics to Approximate a Fixed Location in State Space

DoC metric, it is defined as the portion of edges in the overall graph that connect nodes of the same color. It is the goal of the agent population to minimize this metric.

To measure the quality of improvement provided by any solution mechanism, we compare the observed performance with the expected performance of a random search mechanism. In our case we assume for the random baseline that every agent selects its color uniformly random. In this case the expected global and local DoC equals  $1/G$ . Therefore, we usually plot the DoC observed in experiment multiplied by  $G$  to show the change in the performance compared to random agent behavior. For configurations in which the product of observed DoC and  $G$  is smaller than one, our algorithm outperforms random behavior.

Any agent makes its color decision based on local information provided by its neighbors. It is the general approach of the distributed graph-coloring mechanism in our model to optimize the global DoC by individual local improvements. In analyzing such individual hill-climbing mechanisms it is typically useful to consider the “steepness” of the hill, which determines the guidance that an agent receives from the information that is used by its decision process.

A metric 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 [14] 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 (MD parameter), and the chosen color selection mechanism (CS parameter).

We compute the OSE for an individual agent based on the selection probabilities computed by the agent in a decision cycle. All colors that are excluded by the MD constraint are assigned a probability of zero. If the agent uses the random color selection mechanism, all remaining colors share the same probabilities. If CS equals Roulette, the probabilities of the MD-reduced option set are those used in the roulette wheel selection. Finally, if the agent only chooses among the best options, all but the best colors have a probability of zero and the best ones have equal (non-zero) probabilities.

We compute the OSE as  $-\sum_{i=1}^G p_i \ln(p_i) / \ln(G)$  where  $p_i$  is the probability of the agent to select the  $i$ -th of the  $G$  colors in this decision cycle. The metric reaches its maximum value of one if the agents select randomly among the  $G$  colors.

OSE is a local metric that an individual agent may compute without any additional information. This is an important characteristic to qualify a metric as a candidate for individual agent learning.

A third important metric on the operation of the distributed graph-coloring mecha-

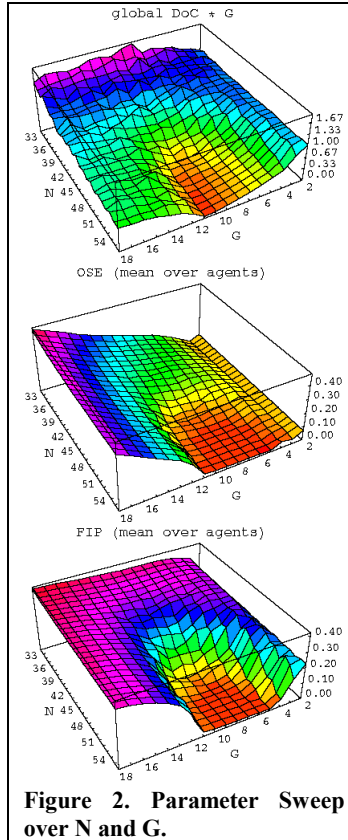


Figure 2. Parameter Sweep over N and G.

nism is the amount of false information that is used by an agent in a color decision cycle. The **False Information Percentage (FIP)** measures the proportion of neighbors for whom the node assumes the wrong color when it makes its decision.

False information is a direct result of the communication latency introduced by the real-world constraints of the deployment of the agent system. At the time an agent chooses its color, its neighbors may have already changed their colors but the message sent by those neighbors has not yet reached this node. Thus, the agent uses stale color information.

The FIP metric is non-local in time in the sense that the agent can only determine the percentage of false information after the fact. But it is local to the agent and thus, by keeping track of the assumed color values that went into a decision process and subsequently arriving (time-stamped) messages that falsify the assumptions, the agent may compute the FIP of a decision cycle CL time-steps later. In our simulation environment we are of course able to take the global view and measure the FIP of an agent directly, but we never let the agent access this information.

Table 2 lists additional metrics that we found useful to measure during our exploration of the model’s parameter space, but which we will not report in this paper.

### 4.3 Mapping the Problem Space

Any undirected graph has a unique chromatic number, which is the minimum number of colors required to color the nodes without any conflict. In terms of our model, it is the smallest  $G$  for which there exists a color assignment of the graph that results in a global DoC of zero.

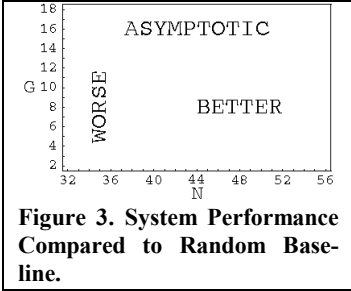
Our exploration of the emergent dynamics of distributed graph coloring was part of a larger project [12] in which we researched the general dynamics of resource allocation. Considering the graph coloring problem as an instance of resource allocation, we focus on the transition from an underloaded state, in which we have more resources (colors) available than required, to an overloaded state, in which there are not enough colors to resolve all conflicts.

No algorithm is known to determine the chromatic number of an arbitrary graph analytically. As a consequence we are not able to determine the location of the transition from an underloaded to an overloaded system just on the basis of our problem parameters. Furthermore, previous experiments with a generalized Minority Game model [13] showed that the characteristics of the solution mechanism strongly influence the emergent dynamics near the problem transition, which may lead to complex patterns of sub-optimal performance.

Figure 2 shows our three metrics (DoC, OSE, FIP) for varying configurations of the problem parameters  $N$  and  $G$ . Each coordinate in a plot corresponds to the statistical mean of the respective metric over 32 simulation experiments of 2500 cycles with

Table 3. Observed metrics by region.

Metric	Region		
	asymptotic	better	worse
DoC * G	medium	low	high
OSE	high	low	medium
FIP	high	low	high



**Figure 3. System Performance Compared to Random Baseline.**

lation experiments of 2500 cycles with varying random seed. The data used in computing the metrics is gathered beginning at cycle 2000, to avoid initial transients. We vary N from 32 to 56 nodes and G from 2 to 18 colors. All other model parameters are fixed to K=30, GC=MinimumNeighbors,

AL=33%, RP=0%, CL=1, MD=Any, CS=Best.

Considering the DoC\*G plot, we find that there is a region near  $N > 44$ ,  $G = 10$  in which the agents clearly outperform the random baseline. We also find that the system performs worse than random for similar values of G but smaller values of N. Finally, we see the system performance approaching the random baseline as we increase the number of colors available. Figure 3 sketches the location of these three performance regions and Table 3 categorizes the observed values in our metrics by region.

How may we explain the structure of the observed system performance in comparison with the random baseline? The answer for the **asymptotic** region can be found right away: Increasing G beyond the transition from the overloaded to the underloaded region in problem space makes the graph-coloring problem so easy that even a random assignment already provides a good solution. Thus, it is not the case that the performance of the agent system fails, but rather, the performance of the random baseline improves with increasing colors available.

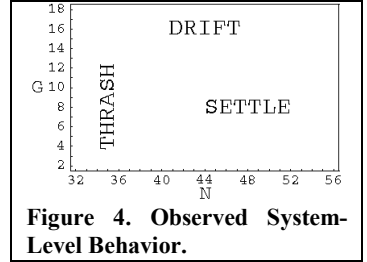
Considering the OSE metric in the asymptotic region we notice a slight decrease of the guidance (increasing entropy) for the average agent as we move to very large values of G. This observation supports our hypothesis, since in the asymptotic region many colors may provide the perfect solution and thus the local hill-climbing algorithm of the agents has many equally perfect options available even on “top of the hill”.

We also observe higher values of FIP in this region. This is due to the fact that even at the optimal local no-conflict state, there are multiple colors that are as good as the current one and thus the CS=Best mechanism may select randomly among them.

As long as the agent changes the color of its node there will be messages sent to the node’s neighbors, whose delay result in false information used by the neighbors. The use of false information does not have a negative impact on the system’s DoC metric, since most choices are good ones anyway. However, the continuous change of color uses processing and communication resources without improving the solution.

In the region where the observed performance performs **better** than random, the agents also quickly find a stable color assignment (low FIP). Especially for larger values of N we see that the algorithm stabilizes even though a conflict-free color assignment is not found ( $DoC * G > 0$ ). The hill-climbing algorithm of the individual agent (MD=Any, CS=Best) drives the overall system into a good local optimum from which it cannot escape.

The plot of the FIP metric supports this hypothesis. The individual agent quickly finds a color assignment, which it cannot improve. Therefore only few color changes are communicated, which in turn reduces the portion of false information (FIP) in the agents’ decisions.



**Figure 4. Observed System-Level Behavior.**

In contrast, in the region in which the system performs **worse** than random, the FIP metric remains high but the OSE metric is low. This indicates that the system does not settle into a local optimum (high DoC) because the agents take well-guided decisions (low OSE) based on outdated information (high FIP) – the system is thrashing.

We hypothesize that the descent into thrashing behavior is caused by the increased connectivity of the overall graph. In all configurations the number of neighbors per node (K) is fixed to 30. Thus, in a system with only 31 nodes, each node must be the neighbor of all other nodes and thus it has to coordinate its color directly with every node in the graph. Once we increase the number of nodes and therefore decrease the K/N ratio, a node has to coordinate only with a subset of the graph and the problem becomes easier.

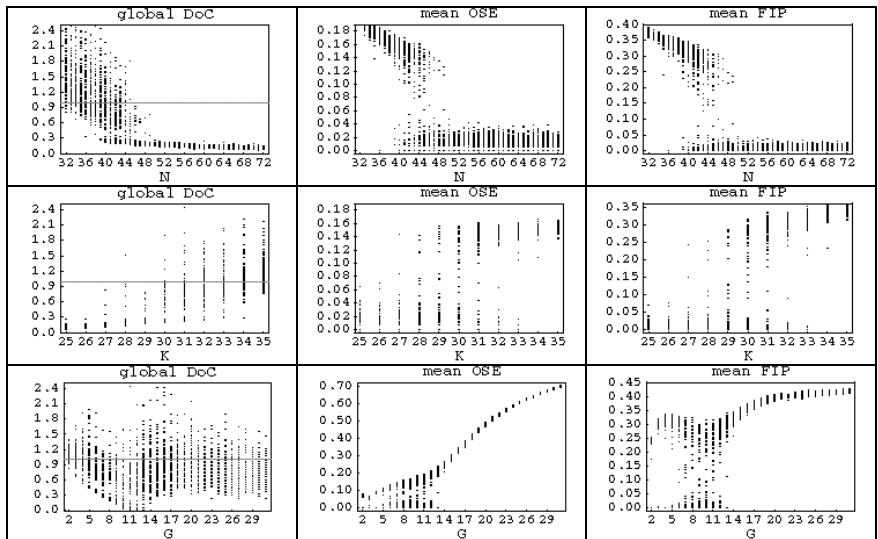
Assume a state in which the nodes have a high probability to change their color. With increasing connectivity of the graph, the probability that a node’s direct neighborhood includes one that just changed its color increases. Thus, high rates of color change produce increasing FIP values for decreasing N. Using outdated information to solve a conflict that was already solved may actually reinstate the conflict. This is particularly true if the number of available colors is small. Therefore we expect thrashing behavior of the distributed graph-coloring mechanism in the low-N low-G region of the problem space.

Figure 4 shows the three different system-level behaviors that we observe in the distributed graph-coloring model.

### 4.4 Phase Changes

The goal of the agent population to color their graph with the

**Table 4. Plotting individual experiments reveals a phase shift into thrashing.**



lowest degree of conflict is fulfilled in the asymptotic region and in the better-than-random region. Only in the region of low  $N$  and low  $G$  we find that thrashing induced by the delay of critical information leads to less-than-optimal performance.

The communication latency parameter is the main driver of the performance decrease in our model. In a deployed system communication latency is an environmental parameter that usually may not be influenced by the agents or the agent programmer. To improve the performance of the agent system for low- $N$  low- $G$  problems, we have to find a solution parameter that counteracts the delay in communication.

Thrashing occurs because the agents make color decisions before critical information has arrived. In our model it is therefore the rate at which decisions are taken – determined by the agents’ activation level parameter (AL) – that needs to be adapted to the particular problem and environmental parameters. In section five we introduce a local adaptation mechanism that provides this capability, but first we analyze the transition into the thrashing region in more detail.

The plots in Table 4 show a series of parameter sweep experiments that vary the three problem parameters  $N$ ,  $K$ , and  $G$  independently while keeping the remaining other parameters fixed to  $N=44$ ,  $K=30$ ,  $G=4$ ,  $AL=33\%$ ,  $RP=0\%$ ,  $CL=1$ ,  $GC=MinimumNeighbors$ ,  $MD=Any$ ,  $CS=Best$ . In all diagrams we plot the respective metric (DoC, OSE, FIP) for each of the 100 individual experiments of 10,000 cycles per configuration, beginning with cycle 9,500.

Plotting results from individual experiments rather than the mean over all experiments at one configuration as we did in Figure 2 shows that the dynamics of our model do not change gradually as we move into the thrashing region. Rather, thrashing is a qualitatively different type of system behavior that either occurs, or not.

Consider, for instance, a move from high to low values of  $N$ . Initially, we are in the region of better-than-random behavior, characterized by low values of DoC, OSE, and FIP. But suddenly, at  $N=49$ , we encounter systems that do not fit this profile, since the observed values in our metrics are significantly higher. Especially in OSE and FIP we find an extremely large jump.

As we further decrease the values of  $N$ , both types of dynamics co-exist for a number of configurations. This overlap region is robust even as we continue to run our experiments for extremely long periods of time. Therefore we conclude that the selection of the respective attractor in system behavior (thrashing or benign) must be driven by the fine-structure of the graph that must be colored. At this point we have no data to explain, which graph-theoretic metric would best describe the boundary of the basins of attraction.

Finally, for even lower values of  $N$ , we find that all experiments result in robust thrashing behavior.

Our description of these changes as a “phase change” relates to the notion of a “phase transition” in physics. Many physical systems can exist in distinct phases and exhibit discontinuities in passing between them. Common examples are melting and evaporation, or ferromagnetic transitions at the Curie temperature. Our results are particularly reminiscent of physical first-order phase

transitions such as the melting and evaporation of water, in which multiple phases can coexist. Physical phase transitions have been a fruitful source of inspiration in computer science, particularly in studying issues of computational complexity [4, 6, 10]. In keeping with usage in the physics community, we believe it is important to reserve the term “phase transition” for a point of nonanalyticity in the behavior of a system, and describe our observations as “phase changes” or “phase shifts.” That is, a “transition” is characterized on the basis of the *mathematical* behavior of a *model* of the system, while a “change” is an *empirical* observation based on experimentation. It will often be the case that the two correspond, but careful use of vocabulary supports the important distinction between empirical and theoretical results.

## 5. INTROSPECTION AND LEARNING

Two of the system-level behaviors exhibited by the distributed graph coloring model (better than random, asymptotic to random) correspond to good problem-solving performance, either because the agent population is “intelligent” enough to solve a complex problem, or because the problem is so simple that even a random solution is likely to be good already. In the third region in parameter space, thrashing prevents the agents from finding and maintaining a good solution. To perform well in this region, agents must possess introspection (the ability to detect that they are thrashing) and be able to learn from their experiences.

The distributed graph-coloring model represents a class of agent systems in which the agents are only able to interact locally based on potentially incomplete or outdated knowledge. Such systems occur for instance in real-world applications that deploy large numbers of agents in a physical environment with limited resources available to the individual agent (e.g., swarming robotics, sensor networks). As a consequence of the real-world constraints, any mechanism that is supposed to detect and counteract the emergence of thrashing behavior must operate at the individual agent level in regards to its input data and its influence on the system dynamics. Just as thrashing is an emergent phenomenon, thrashing prevention must be an emergent functionality as well to be implemented in a completely distributed and decentralized fashion.

In our discussion of the regions of performance we identified the delay of critical information as the main driver for the emer-

gence of thrashing behavior and we suggested that the rate of decision making (parameter AL) may be the most suitable solution parameter that needs to be adapted to prevent thrashing. Figure 5 plots another sweep around the general configuration chosen for the plots in Table 4. In this sweep we only vary the AL parameter, but we again find that the systems eventually fall into the thrashing attractor for large enough AL. In this set of experiments neither the complexity of the problem nor the characteristics of the environment change. It is only the rate at which the agents take their color decisions that determines whether the system is thrashing or not. Thus, we conclude that for any configuration there exists a critical maximum AL value, above which thrashing will occur.

Any system with a globally fixed AL parameter faces a dilemma. If AL is too high, some possible scenarios (problem and environmental parameters) will lead to thrashing, while if it is too

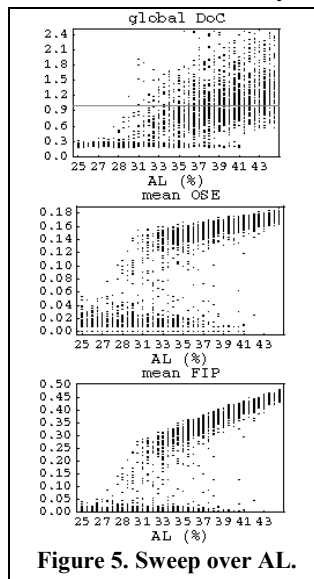


Figure 5. Sweep over AL.

low, response will be slower than would otherwise be possible. Real-world applications are often driven toward this dilemma. Economic pressures on a resource allocation system tend to keep the number of resources low, corresponding to low  $G$ , where random choice is ineffective, while time pressures will tend to force decision rates up, corresponding to high  $AL$ .

We address this dilemma by expanding the individual agent behavior with a local “AL-Learning” mechanism. Our proposed mechanism causes the agent to skip probabilistically some decision cycles that it normally would have executed given its preset  $AL$  parameter. Thus, we are able to slow an agent’s decision rate down until the thrashing stops. With this learning mechanism in place we may preset the  $AL$  parameter to high values that permit short solution times, relying on the agents to modulate their own activity to avoid thrashing.

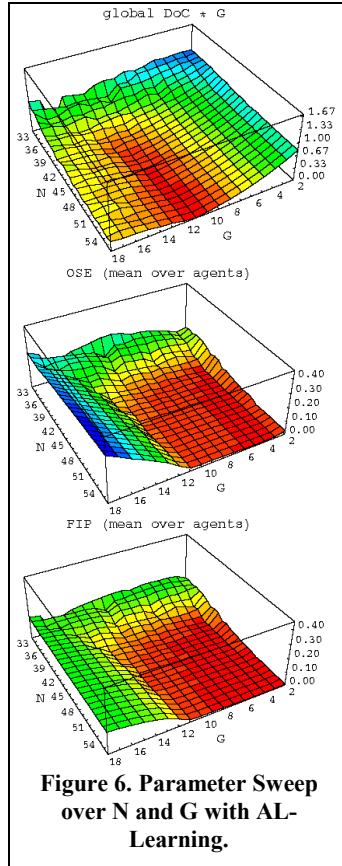
The change in the behavior of an individual agent is driven by an additional parameter, which we call the agent’s No-Action Pheromone ( $NAP$ ). It determines the likelihood that an agent’s color decision is skipped and therefore an agent with a  $NAP$  of zero operates at the decision rate specified by the  $AL$  parameter, while  $NAP$  values larger than zero reduce the decision rate.

Initially, any agent’s  $NAP$  is zero, but over time the agent collects evidence that may increase the parameter. It is the rate and strength of the locally observed evidence that drive the increase in  $NAP$ .

With the accumulation of knowledge (evidence) by the agent, we face a truth-maintenance problem. Over time the agent may observe states that may be interpreted as evidence for thrashing, but that are just part of the normal operation. Also, for instance through changes in other agents’ behavior, old evidence may no longer justify a decreased decision rate. Therefore, following the truth-maintenance approach of natural agent systems (e.g., insect colonies), an agent immediately starts to forget any knowledge that it receives unless it is continuously reinforced.

We find that the observed percentage of false information is a superior source for the extraction of evidence for thrashing behavior at the individual agent level. As the plots in Table 4 show, we generally observe low  $FIP$  values when the global performance (degree of conflict) is high and high  $FIP$  values if the  $DoC$  is high. As mentioned before, the individual agent cannot observe the portion of false information in its current color decision directly, but by keeping track of incoming and used color information, it may determine the  $FIP$  with a delay equal to the communication latency of the system. Our experiments show that using this delayed  $FIP$  is sufficient to remove the system from the thrashing region.

We treat the current  $NAP$  value as a concentration of a digital pheromone as we used them in several previous applications ([11], [3], [1]). Consequentially, we deposit additional pheromone into  $NAP$  in proportion to the strength of the current evidence, which is the locally observed delayed  $FIP$ . At the same time we continually “evaporate” the pheromone, reducing its concentration through multiplication with a fixed No-Action Probability Evaporation ( $NAPE$ ) factor between zero and one.



**Figure 6. Parameter Sweep over  $N$  and  $G$  with AL-Learning.**

Based on our formal analysis of the dynamics of digital pheromones [1], we know that a repeated evaporation and deposit of the strongest evidence ( $FIP=1$ ) drives the  $NAP$  parameter to approximate a fixed point of  $1/(1-NAPE)$ . Treating this fixed point as the upper limit, the agent skips a color decision cycle with a probability of  $NAP*(1-NAPE)$ .

Figure 6 repeats the parameter sweep experiment conducted for Figure 2 with the emergent introspection and learning mechanism at a  $NAPE$  value of 0.9. For easier comparison we scaled the z-axes to the same interval as in plots in Figure 2. We see that the thrashing region is completely gone (no performance worse than random) and almost completely replaced by good performance in the other two regions.

A companion paper [9] analyzes this approach to calming hyperactive agents in more detail, and shows how it can be generalized to more concrete resource application problems and to reasoning about approaching deadlines.

## 6. CONCLUSION

In this paper we present a model of distributed coordination among agents with limited resources in a real-world environment. The distributed graph-coloring mechanism, proposed in [5], deploys agents in an environment that delays the transfer of information in the local interactions of neighboring agents in an undirected graph. The agents exchange information to as-

sign colors from a finite set to their respective node in the graph in a way that globally minimizes the number of neighbors with the same color. The graph-coloring problem is an abstraction of many important agent coordination problems.

The high degree of complexity of the emergent system-level dynamics suggests an experimental approach to the analysis of the model, especially for large systems as they occur in real-world applications. To support systematic experimentation and gathering of data from individual experiments we developed a generic software infrastructure that executes for a specified subset of the model’s parameter space multiple replica of the model with different random seeds. The infrastructure enables us to explore the dynamics of an agent simulation model efficiently.

The parameters of the distributed graph-coloring model fall into three different classes. Problem parameters configure the specific graph-coloring problem, solution parameters specify the features of the agent’s color decisions, and environmental parameters characterize the deployment scenario in which the agents solve the particular problem.

Analyzing the dynamics of distributed graph coloring in a restrictive environment we find that there are three distinct regions of system behavior. On the one hand, the system may show good performance either because the agent population is sufficiently intelligent to settle on a low-conflict solution, or because the problem is so easy that almost any randomly selected configuration results in a low degree of conflict. On the other hand, the agents may be prevented from finding and maintaining a good solution because critical information is delayed and the system falls into thrashing behavior.

Detailed analysis of the transition into the thrashing region reveals that thrashing is a qualitatively different phase in behavior space. Changing any of our model parameters to transition into this phase takes us through an overlap region in which systems may or may not fall into the thrashing attractor. This observation leads us to conclude that there must be an implicit parameter, such as a characteristic of the randomly created graph, that determines the attractor selection in the sensitive overlap region.

Thrashing significantly reduces the problem-solving performance of the system. We find that for any configuration there is a maximum AL value, above which the system falls into the thrashing attractor. The individual agent may manipulate the solution parameter AL and thus we propose an adaptive mechanism based on the local metric FIP (false information percentage) that dynamically adjusts the effective decision rate of the agent to prevent thrashing. Our experiments verify the performance improvements.

The addition of the emergent introspection and learning mechanism drastically improves the performance of distributed systems of agents coordinating in a limiting environment. Rather than being forced into very conservative global decision rates to guarantee that thrashing cannot occur in any possible scenario, we may start the agents with a higher decision rate that leads to faster solutions and slow them down only as thrashing occurs.

## 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.

## 7. REFERENCES

- [1] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.
- [2] S. Brueckner and H. V. Parunak. Resource-Aware Exploration of Emergent Dynamics of Simulated Systems. In *Proceedings of AAMAS'2003 (submitted)*, 2003.
- [3] S. A. Brueckner and H. V. D. Parunak. Swarming Agents for Distributed Pattern Detection and Classification. In *Proceedings of Workshop on Ubiquitous Computing, AAMAS 2002*, pages (forthcoming), 2002.
- [4] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI-91*, pages 331-337, Morgan Kaufmann, 1991.
- [5] 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.
- [6] T. Hogg, B. A. Huberman, and C. Williams. Phase Transitions and the Search Problem. *Artificial Intelligence*, 81:1-15, 1996.
- [7] C. Langton, R. Burkhart, and G. Ropella. The Swarm Simulation System. 1997. <http://www.swarm.org>.
- [8] L. Meertens and S. Fitzpatrick. Peer-to-Peer Coordination of Autonomous Sensors in High-Latency Networks using Distributed Scheduling and Data Fusion. Technical Report KES.U.01.09, Kestrel Institute, 2001.
- [9] H. V. D. Parunak, S. Brueckner, R. Matthews, and J. Sauter. How to Calm Hyperactive Agents. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, pages (submitted), 2003.
- [10] H. V. D. Parunak, S. Brueckner, J. Sauter, and R. Savit. Effort Profiles in Multi-Agent Resource Allocation. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS02)*, pages 248-255, 2002.
- [11] H. V. D. Parunak, S. A. Brueckner, J. Sauter, and J. Posdamer. Mechanisms and Military Applications for Synthetic Pheromones. In *Proceedings of Workshop on Autonomy Oriented Computation*, 2001.
- [12] H. V. D. Parunak, R. Savit, S. A. Brueckner, and J. Sauter. A Technical Overview of the AORIST Project. ERIM, Ann Arbor, MI, 2001. URL [http://www.erim.org/cec/projects/aorist/AORIST\\_Snapshot\\_0104.pdf](http://www.erim.org/cec/projects/aorist/AORIST_Snapshot_0104.pdf).
- [13] R. Savit, S. A. Brueckner, H. V. D. Parunak, and J. Sauter. Phase Structure of Resource Allocation Games. *Physics Letters A.*, (forthcoming), 2002.
- [14] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. Urbana, IL, University of Illinois, 1949.