An Experimental Study of the Emergent Behaviors of Self-Organized Agent Communities

Shufeng Wang
National Laboratory for Parallel and Distributed Processing, Changsha, 410073, China
shufeng.wang@gmail.com

Hong Zhu
Department of Computing, Oxford Brookes Univ., Oxford, OX33 1HX, UK
hzhu@brookes.ac.uk

***Abstract*—Emergent behavior is an essential feature in multi-agent systems and plays a significant role in the applications of agent technology. Because of the huge gap between individual agents' behaviors and those of the whole system, specifying and reasoning about emergent behaviors are notoriously difficult. Simulation has been the essential method to study emergent behaviors in multi-agent systems. In this paper, we report an experimental study of the emergent behaviors of self-organized agent communities, in which emergent behaviors play a crucial role. The experiments confirmed the results of a theoretical analysis of agent communities using a formal theory called Scenario Calculus. It further provided insight into the dynamic features of the system that were very difficult to obtain by using formal logic, such as the speed of convergence to the emergent states and the relationships between the convergence time and various parameters of self-organized agent communities systems.**

***Keywords:* Agent communities, Emergent behavior, Simulation, Reachability, Stability, Convergence**

I. INTRODUCTION

Emergent behavior is a common phenomenon in complex systems. In both natural and artificial complex systems, individual components perform their actions and make decisions based on local information, while the whole system demonstrates properties and behaviors that have strong global features [1]–[4]. Ant colony is a typical example in nature which consists of a number of ants that act with very simple rules while the whole colony can accomplish complex tasks that in some cases far exceed a single ant's capability [5]. The Amalthaea system developed by MIT Media Lab is [6], a multi-agent system for web information retrieval and filtering [6]. It consists of a collection of relatively simple agents that each of them only retrieves or selects one particular type of information on the web. However, when organized into an evolutionary ecosystem, they can provide information that suits a user's interests even when the user's interests are changing from time to time. Similar emergent behaviors also exist in resource allocation in a distributed environment [7], e-commerce (such as online auctions), peer-to-peer network [8], and many other application areas. However, in the development of software for complex systems, it is very difficult to understand how the system as a whole will behave because there is a huge gap between individual's autonomous behaviors and those of the whole system. Therefore, understanding emergent behaviors is essential to the development of multi-agent systems.

In general, there are two approaches to gain insight into such emergent behaviors [9]. The first is to conduct experiments and to simulate the system and observe the emergent behaviors. This approach has been widely used in the research on emergent behaviors [5], [10], [11]. Simulation tools and environments such as StarLogo [12] have been developed. However, it is difficult to cover all possible scenarios and operating conditions of multi-agent systems. Thus, results obtained are less conclusive because they are more or less empirical laws. Existing multi-agent simulation tools such as StarLogo provide strong supports to the design of behavior rules of the agents in complex systems and the visual representation of the evolution processes to demonstrate their emergent behaviors. However, they support little about repeated experiments with random initial states of the systems with different distributions and the automatic collection and statistical analysis of data. The second approach is to analyze the system with formal theory and methods. Theoretically speaking, formal analysis based on sound theories can provide more conclusive results than empirical laws. However, this approach has only started recently [13]. Few formal theories and methods have been proposed in the literature because specifying and reasoning about emergent behaviors are notoriously difficult. Few formal theories and methods have been proposed in the literature. Among the very few theories is the Scenario Calculus recently proposed in [13]. It is desirable to conduct experiments to validate the theory by examining whether the results of theoretical analysis match the reality. Moreover, simulation experiments can also explore some of the intrinsic features of complex systems that are currently unable to obtain by applying formal methods.

Agent communities are typical multi-agent systems in which emergent behaviors play a crucial role. Community is a common phenomenon in natural ecosystems and human societies. With the wide usage of Internet, it can easily bring people or entities together situated at distributed locations to form kinds of virtual communities. The considerably increasing dimensions and complexity of contemporary Internet-based communities require a substantial amount of management work to organize and administrate proper groups in a large-scale distributed environment. A series of computational techniques have been proposed to automate this process. A common method to organize communities is to cluster entities according to their similarity [14], [15]. Because this kind of method employs pre-defined features and computing models to create communities, it involves a

significant amount of computation for constructing and re-clustering in dynamic situations. Communities can also be formed according to the associated links between entities [16], [17]. This approach needs a full acknowledgement of the associations between entities to perform community formation. This prerequisite sometimes makes the community formation inapplicable to achieve in large-scale and dynamic applications. The self-organizing communities approach proposed by Wang in [18] presented a novel solution to form communities in a decentralized way. It takes the advantage of emergent behaviors of autonomous agents to form communities. It was proven that the average formation time of self-organized communities may increase linearly with the log of the number of users and also linearly with the number of middle agents [19].

A formal theory called Scenario Calculus for the specification of and reasoning about emergent behaviors was proposed in [13] based on the SLABS language for the formal specification of multi-agent systems [13], [20]–[23]. In [24], it is applied to the study of the emergent behaviors in a variety of self-organized agent communities, and in particular the autonomous formation of the communities. It further extended the work reported in [18] and examined a variety of subtle variants of the algorithm proposed and studied in [19]. In particular, it formally proved the logical properties of the reachability of community formation, the stability of the communities, and the convergence of self-organized communities for the variants of the algorithms and their combinations.

In this paper, we will apply the experiment method to the study of the emergent behaviors of self-organized agent communities. The goals of the study are two-fold. First, the experiments are aimed at validating the results obtained in our previous theoretical analysis. The second is to explore the features of community formation that are difficult to obtain by theoretical analysis but suitable for experimental study. Such features include the times required by an agent community to reach an emergent state from various initial settings, and the relationships between the convergence time and various parameters of the self-organized agent community systems. To achieve these goals, an experiment environment is developed to enable us to perform systematic repeated executions of agent community systems and to collect and analyze data efficiently.

The remainder of the paper is organized as follows. Section 2 is an overview of self-organized agent community. Section 3 describes the experiment environment and presents experiment results. Section 4 concludes the paper with a summary of the results and a discussion of future work.

II. Self-Organized Agent Community

This section describes the structure and operations of self-organized agent communities. A formal specification of the system can be found in [24].

1. Structure and Operation

In a system of self-organized agent communities, there are two types of agents: members and organizers. Each organizer organizes a community and keeps a registry of the members of its community. Each member is registered only to one organizer at any time. Each member is interested in a particular category of knowledge and has certain knowledge on a set of specific topics of the category.

A member R may raise a question to its organizer G about a specific topic of its interested category. The organizer G will then search for a member in its community who knows this topic to answer the question. If such a member S is found, the organizer will introduce S to R. And the member S will then respond to the query. If the organizer G cannot find such a member within its community, it will ask for help from another organizer H by making a query on the topic. If the organizer H finds a member T in its community who knows the topic, it will pass T's identity to organizer G. The organizer G will then introduce T to R, and T will then answer R's question.

At any time, any member can raise a question on any topic of its interested category as long as it doesn't know. It is possible that the same member asks the same question many times and the questions may cover all topics in any order. An organizer can search a member within its community and query another organizer also in random order.

The performance of an organized community heavily depends on the configuration that members are grouped in communities. It is more efficient if a question raised by a member can be answered within the community. If communities can reconfigure themselves so that any member can get the answers to its questions within its own community, the overall performance of the system will be the best.

There are various ways that communities can reconfigure through members' autonomous behaviors in moving from one community to another without global information in order to achieve optimized efficiency. Assume that, at the beginning, a member with some knowledge of a category is registered to an organizer at random. Therefore, the efficiency of the system cannot be guaranteed. Reconfiguration of the communities is necessary, which is achieved by members changing their memberships to the communities. A member moves from one community to another by deregistering from one organizer and then registering to another. In this model, members are autonomous to decide when or where to move, which is not controlled by the organizers or the system.

A member will make a decision about whether to move to another community according to the community's attraction to it when it communicates with a member of the community. These situations include: when a member as a requester raises a question which can not be answered locally by any member of its community, but can be answered by a member of another community; or when a member as a server answers a question raised by a member outside its community. In other words, after a question is answered both the requester and the server will make a decision about whether to move. The community's attraction can be derived from the community itself or some member within.

Suppose that a member R raises a question on a topic, which is not known by any member of its community. While a member T of another community provides a successful service of answering R's question. Then, members T and R

will try to be in the same community. This can be achieved by either member T moving into member R's community or member R moving into member T's community. A simple rule to decide which member will move is that the one who is in the more attractive community will stay while the one who is in the less attractive community will move. When the agents calculate a community's attraction in the same way, it is certain that one of them will stay and the other will move, thus they will be together after the actions. The situation is more complicated if agents calculate the attraction differently. In such cases, it may happen that both of the requester and the server move to the other community simultaneously, thus they may still be separated after taking the actions. In all cases, the question is whether agents' moving between communities will lead to an optimal configuration.

2. Varieties of Agent Communities

According to the different definitions of the community's attraction from a member's point of view, four types of members were studied in [24].

- *CAKM (Community's amount of knowledge of the category)*: The agent measures a community's strength of attraction according to the total amount of knowledge of its interested category hold by the agents registered to the community.
- *CASM (Community's number of agents in the specific category)*: The agent measures a community's strength of attraction according to its the number of agents that have the same category.
- *PARM (Personal amount of knowledge of the service provider)*: The agent measures a community's strength of attraction as according to the amount of knowledge that the specific service provider has in the category.
- *PAEM (Personal attribute irrelevant to its knowledge)*: The agent measures a community's strength of attraction according to an attribute of the specific service provider, where the attribute is irrelevant to its knowledge.

Therefore, we have five types of systems.

- **CAKM**: Systems only contain CAKM members.
- **CASM**: Systems only contain CASM members.
- **PARM**: Systems only contain PARM members.
- **PAEM**: Systems only contain PAEM members.
- **Hybrid**: Systems contain several types of members.

3. Emergent Behavior

An emergent behavior of a system of organized communities is that the members of the communities will gradually grouped in a way so that members of the same category come together in one group and are registered to the same organizer. For the sake of simplicity, in the sequel, a community that is organized by organizer G will be referred to as community G.

A. Notions and notations

In order to define clearly the emergent behavior, we first introduce some notions and notations.

At time moment t, the *population* of the members of a category C in a community G is denoted by $P_t^G(C)$. The overall population of the members of a category C in the whole system is denoted by $P_t^*(C)$.

At time moment t, the *domain of knowledge* in category C in a community organized by G is denoted by $D_t^G(C)$. The domain of knowledge of category C in the whole system is denoted by $D_t^*(C)$.

At time moment t, a community organized by G is *complete* with respect to the knowledge of category C, if and only if $D_t^G(C) = D_t^*(C)$.

A world of organized communities is a *closed world*, if its population does not change. A category C of knowledge is *non-trivial*, if $D_t^*(C) \neq \emptyset$.

B. Emergent states and their properties

The result of a process of an agent community's emergent behavior is that the system reaches a state in which the operation of the system is optimized. In [24], two emergent states of self-organized agent communities were recognized and studied. They are defined as follows.

The state of a world of organized communities is *mature*, if for every non-trivial category C of knowledge, there is a complete community with respect to C. And we say the categories are *mature* too.

The state of a world of organized communities is *optimal*, if every member is in a complete community of its category.

A system may demonstrate different dynamic properties of emergent behaviors. An emergent state is called *reachable* if for every execution, the system will reach the emergent state if execute the system in a time long enough. The state is called *stable*, if the system will stay in the state whenever it reaches the state. If the state is reachable and stable, we say that the state is *convergent*. In [24], the properties of the emergent states in five different agent community systems were investigated. The results are summarized in Table 1, where √ means the emergent state is reachable (or stable) and × means the emergent state is not always reachable (or stable).

TABLE 1
RECURRENCE PROPERTIES OF EMERGENT BEHAVIORS

	MATURITY		OPTIMALITY	
	Reachable	Stable	Reachable	Stable
CAKM	√	√	√	√
CASM	√	×	√	√
PARM	√	×	×	√
PAEM	√	×	×	√
Hybrid	×	×	×	√

III. THE EXPERIMENTS

In this section, we report the experiments that confirmed the theoretical results and the main findings of the experiments that were unknown before.

1. Experiment Environment

To enable the experiments, we developed an experiment environment to simulate the execution of agent communities. We also provide a graphical user interface tool to setup simulation experiments and collect data for statistical analysis. Figure 1 is the snapshot of its interface.

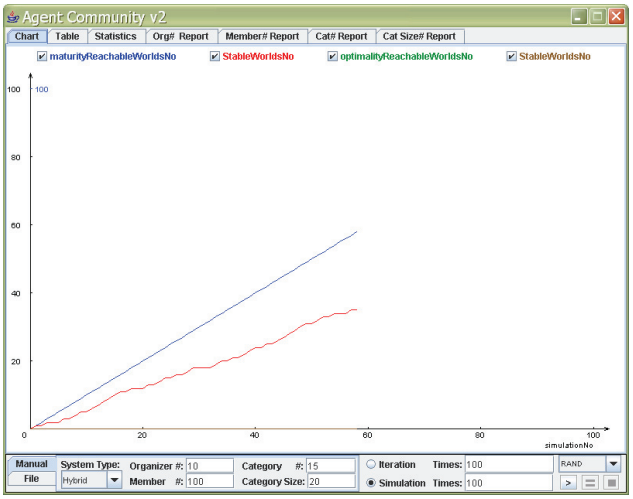


Figure 1. Interface of the experiment environment

As shown in Figure 1, the experiments take four parameters as the input to the agent community simulator:

- **k:** the number of organizers,
- **m:** the total number of member agents in the system,
- **c:** the number of categories of knowledge, and
- **s:** the size of each category of knowledge.

For each given set of parameters, an initial setting of the agent communities is generated at random according to the uniform distribution. Thus, each agent is initially assigned at random with a category, a non-empty set of topics and an organizer. The randomly generated settings are non-trivial in the sense that every organizer has at least one member and every category has at least one member. The following constraints are also imposed on the parameters unless explicitly stated otherwise: $2 \le k \le 100$, $2 \le m \le 100$, $1 \le c \le 100$, $c \le m$, $1 \le s \le 70$. Figure 2 is a screen snapshot showing an initial setting on the left and the setting after several iterations in an execution of the same system on the right.

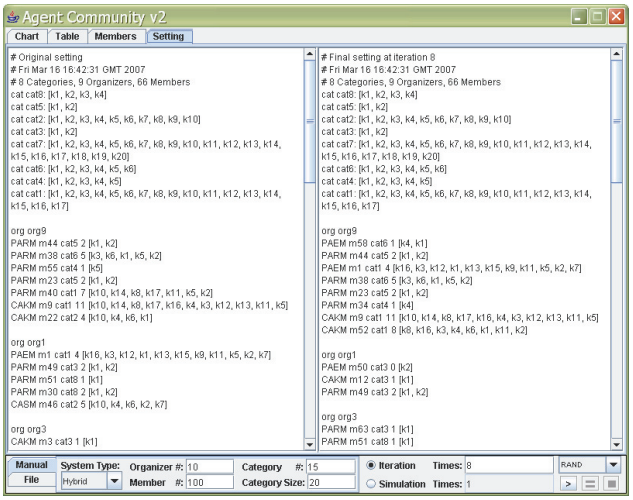


Figure 2. Example of initial setting and final setting

The user can select one of the five types of agent community system to perform an experiment. Two basic types of experiments are supported by the tool. The first is to run a system for once and to collect the data of the execution and display them in various windows. The second is to repeat the execution on an initial setting for a number of times and to collect the data, display the data in a graphic user interface and perform statistical analysis of the data.

In each execution, the algorithm is run for a certain number of iterations of user's choice. In each iteration, there is a random number of members to raise questions, get their answers if exist, and then to make movement according to the members' behavior rules. Each member only raises one question in one iteration cycle. After each iteration cycle, the system's global state is checked to see if it is in an emergent state of mature or optimal. Figure 1 is the screen snapshot in the middle of an execution for repeating simulation for 100 times that displays the statistical data.

2. *Experiment Results*

Using the tool, we systematically carried out a large number of simulations of agent communities with a wide range of parameters. The experiments not only validated the results of our previous theoretical analysis, but also enabled us to observe new phenomena of agent communities. The following reports the main findings.

A. Validation of theoretical results

To validate the theoretical results [24] summarized in Table 1, we set a large number of iterations for each execution of the system. Initial settings were generated for randomly selected parameters in the following ranges: $2 \le k \le 100$, $2 \le m \le 100$, $1 \le c \le 30$, $2 \le s \le 30$ and the number of iterations in each execution $n=500$. For each of the five types of agent community systems, a total of 1000 initial settings were generated at random. The system on each initial setting was run repeatedly for 100 times.

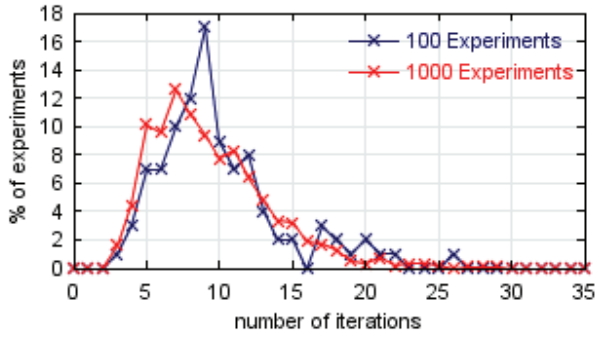
The experiments confirmed the theoretical analysis results in the following sense.

- For an emergent state that is reachable in a type of agent community system, it is confirmed by experiments if and only if in all runs of the agent community system on every initial setting the system is in the emergent state after some iterations.
- For an emergent state that is not always reachable, the property is confirmed if and only if there is an initial setting on which there is at least one run of the agent community system that does not reach the state in all iterations within the set number of iterations.
- For an emergent state that is stable, the property is confirmed by the experiments if and only if for all initial settings and every execution of the agent community system on the setting the system reached the state after a certain number of iterations and the system is in the state for all iterations after that in the same execution.
- For an emergent state that is not stable, the experiments confirm the property if and only if there is at least one initial setting and at least one execution of the agent community system on the setting such that the system reaches the state at a certain iteration and for at least one iteration after that the system is not in the state in the same execution.

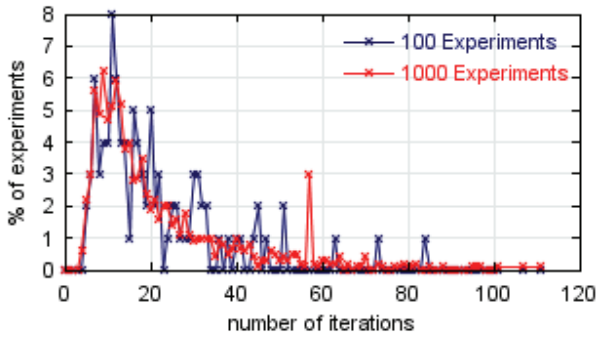
It is worth noting that, the experimental confirmation of the properties is not as conclusive as the theoretical proofs for reachability, unreachability, and stability. Only non-stability can be conclusively confirmed. However, we can gain a high confidence of the results from a large number of randomly generated initial settings and a large number of random executions of the system on each initial setting to a large number of iterations.

B. Distributions of convergence time

A question that rose in the experiments reported above is that how many iterations one would consider to be large enough. This question was answered by the second group of experiments that aims to discover how fast an agent community system converges to an emergent state. The number of iterations that a system needs to reach an emergent state represents the time that the system needs to reach the state. It is called the *convergence time* in the sequel.



(a) Maturity convergence time



(b) Optimality convergence time

Figure 3. Distributions of convergence time on the same initial setting with parameters k=50, m=50, c=20, s=30.

Figure 3 shows the distributions of convergence time for the mature and optimal emergent states in part (a) and (b), respectively. In the figure, the blue line represents the results obtained from 100 experiments, while the red line represents 1000 experiments. The x axis is the number of iterations the system needs to converge; the y axis is the percentage of the number of experiments in which the system converges to the emergent state using the number of iterations. As shown in these figures, there is no significant difference in the data obtained from 100 experiments and from 1000 experiments. This is confirmed in statistical data shown in Table 2.

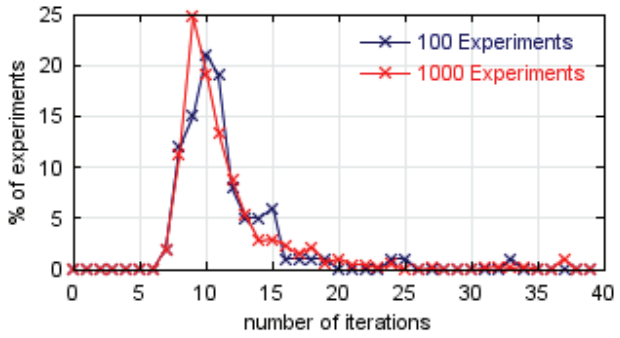
When the systems are run on different initial settings, the same distributions of the average convergence time were observed as shown in Figure 4, where the x axis represents the average iterations in which a system converges to the emergent state, and the y axis is the same as in Figure 3. The overall average convergence time for random initial settings with the same parameters of k=50, m=50, c=20, s=30 is presented in Table 3.

TABLE 2
AVERAGE CONVERGENCE TIME ON FIXED INITIAL SETTING

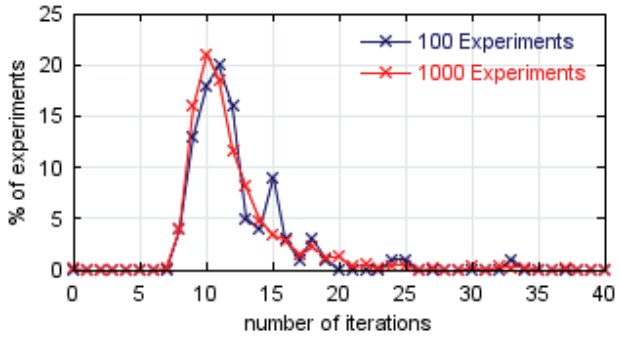
	Maturity		Optimality	
Experiments Times	Average Value	Standard Deviation	Average Value	Standard Deviation
100	9.99	18.515	20.820	213.280
1000	9.37	16.520	20.972	267.900
Relative Divergence	3.0%	5.7%	0.4%	11.4%

TABLE 3
AVERAGE CONVERGENCE TIME ON RANDOM INITIAL SETTINGS OF FIXED PARAMETERS

	Maturity		Optimality	
Experiments Times	Average Value	Standard Deviation	Average Value	Standard Deviation
100	10.848	13.852	11.681	13.693
1000	10.607	12.575	11.450	14.012
Relative Divergence	1.1%	4.8%	1.0%	2.1%



(a) Average maturity convergence time



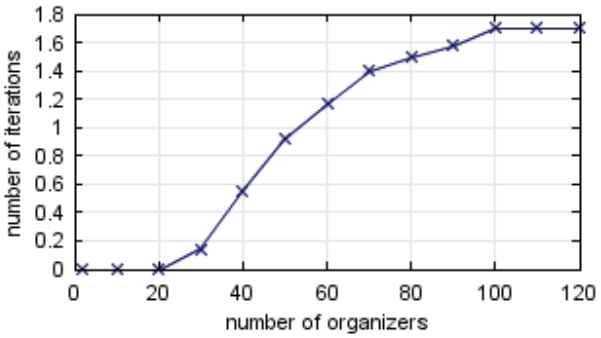
(b) Average optimality convergence time

Figure 4. Distribution of average convergence times on randomly generated different initial settings of the same parameters with k=50, m=50, c=20, s=30.

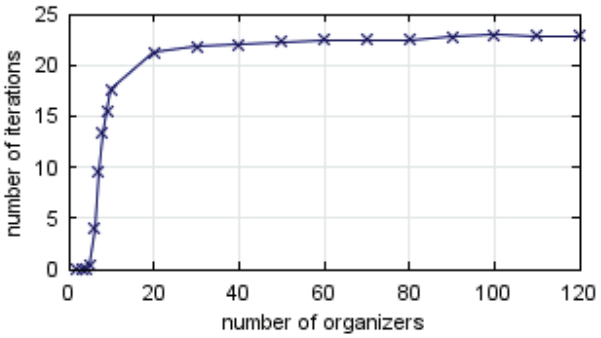
To explore the relationships between convergence time and various parameters of agent communities, we carried out further experiments with CAKM communities. The results are reported below.

C. The effect of number of organizers

In the experiments that aim at understanding the effect of the number of organizers on convergence speed, we fixed the parameters m (the number of members), c (the number of categories) and s (the size of each category). When k>m, there are organizers empty, which has no effect on the operation of the system. Thus the parameter k (number of organizers) varied from 2 to m. For each value of k, 100 initial settings were generated at random with the uniform distribution, and on each initial setting the agent community system was executed repeatedly for 100 times. The average convergence time was then calculated.



(a) Average maturity convergence time



(b) Average optimality convergence time

Figure 5. Convergence time for varying number k of organizers with fixed m=100, c=1 and s=30.

The particular parameters used in this group of experiments were: m=100 members, c=1 category and the size of each category s=30. The number of iterations in each execution was set as 500, which according to the results of above experiments is large enough for the agent community system to converge. The results are shown in Figure 5.

D. The effect of number of members

To study the effect of the number of members on convergence time, we fixed the parameters of k (number of organizers), c (the number of categories) and s (the size of each category) and let the parameter m (the number of members) vary.

The particular parameters used in the experiments were: k=100 organizers, c=1 category and the size of each category s=30. The number of iterations set in the experiments was 500. Similar to the experiment reported in sub-section C, 100 initial settings were generated at random and on each initial setting the CAKM agent community system was executed for 100 times and average convergence time were calculated.

It is worth noting that an organizer that is initially not assigned with any member has no effect on the operation of the system. In order to minimize the effect of empty initial organizers on statistical results, two strategies were used in the random generation of the initial settings according to the value of m. When m>k, there were at least one member initially assigned to each organizer. When m≤k, each organizer was initially assigned with at most one member.

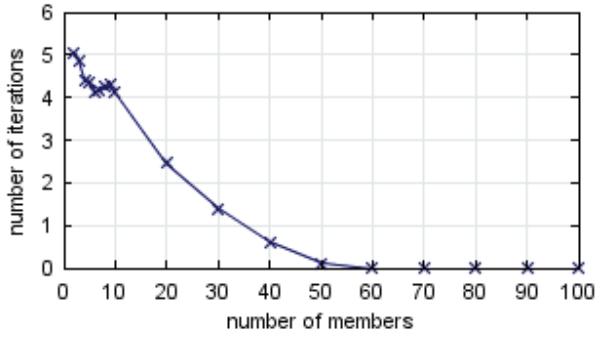


Figure 6. Average maturity convergence time for varying number m of member with fixed k=10, c=1 and s=30.

The experiments demonstrated that, see Figure 6, for the maturity emergence state, when m>k, the convergence time decreases as the number m of members increases. When m≤k, the number of non-empty organizers is equal to the number m of members. As shown in Figure 6, the convergence time first decreases, and then increases. This phenomenon was observed with surprise and difficult to explain. We believe it is the result of other factors, such as the interaction with the other parameters like category number and size.

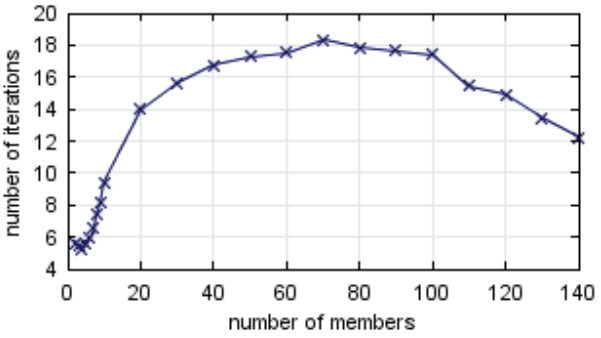


Figure 7. Average optimality convergence time for varying number m of members with fixed k=10, c=1, and s=30.

For the optimality emergence state, the experiments showed Figure 7 that when m>k, with the number m of members increasing, the convergence time first increases, and then decreases. When m≤k, with the number m of members increasing, the convergence time for the optimal state first decreases and then increases. This is in the same pattern of the convergence time for the mature state.

E. The effect of number of categories

In the experiments to investigate the effect of the number of categories on convergence speed, we varied the number c of categories while kept the other parameters fixed. The fixed parameters were: k=100 organizers, m=100 members and the size of each category s=30. When c≥m, there are categories empty, which has no effect on the operation of the system. Thus, the variable c varied in the range of 1<c<m in this set of experiments. The number of iterations was also 500. The results are shown in Figure 8 and Figure 9.

Figure 8 shows that for the maturity emergence state, with the number c of categories increasing, the convergence time first increases, and then decreases.

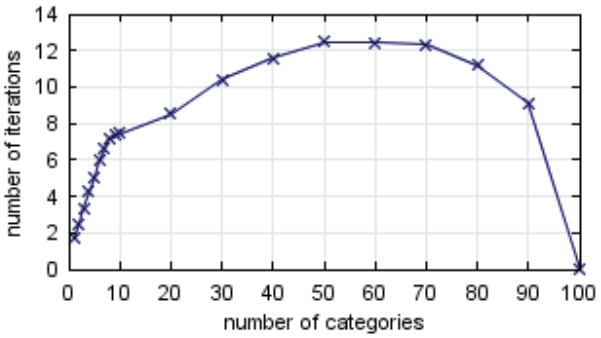


Figure 8. Average maturity convergence time for varying number c of categories with fixed k=100, m=100 and s=30.

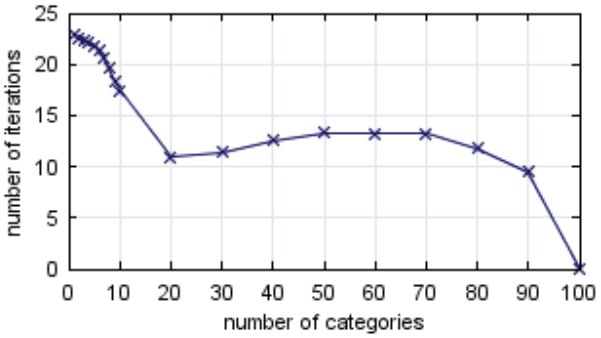


Figure 9. Average optimality convergence time for varying number c of categories with fixed k=100, m=100 and s=30.

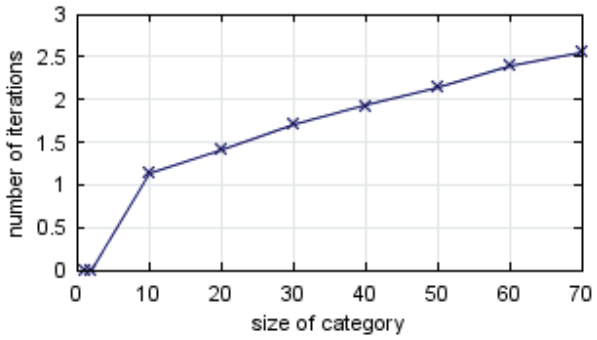
Figure 9 shows that with the number c of categories increasing, the convergence time for optimality emergence state first decreases sharply, then increases gently, and at last decreases sharply again.

F. The effect of the size of categories

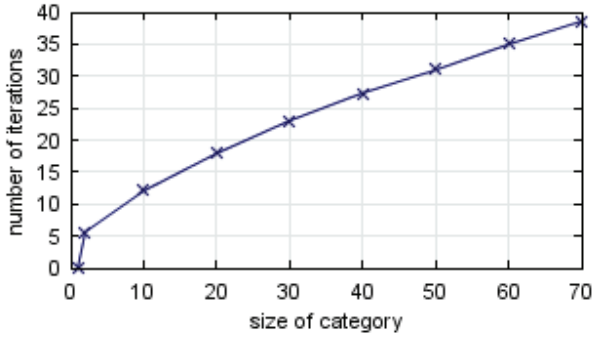
In the experiments about the size of categories, we varied the size of categories s while kept other parameters fixed, which were k=100 organizers, m=100 members, c=1 category. The number of iterations was 500 again.

The convergence time for both maturity and optimality emergence states increases with the size of the category increasing, as shown in Figure 10, although the speed is difference.

The above experiments were also repeated with smaller values of the parameters. We found that the curves of smaller parameters are not as smooth as the figures above. However, they are in the same pattern as those presented in this paper.



(a) Average maturity convergence time



(b) Average optimality convergence time

Figure 10. The average convergence time for varying values of category size s with fixed k=100, m=100 and c=1.

IV. CONCLUSION

In this paper, we presented an experimental study of the emergent behaviors of self-organized agent communities, which have been analyzed by using a formal theory called Scenario Calculus. The experiments demonstrated that experimental study can play a complimentary role to formal methods rather than to replace one by the other. First, although, theoretically speaking, the experiment results are not 100% conclusive that formal analysis can achieve, we can use the experiment method to validate the results of theoretical analysis with high confidence. This also tests the implementation of the simulation against the formal specification of the multi-agent emergence system. In particular, this paper confirmed the correctness of analysis of the properties

of the emergent behaviors of five variants of self-organized agent communities systems.

Second, experiment method can provide insight into the complicated dynamic emergent behaviors that are very hard to achieve through formal analysis. In this paper, we systematically studied the convergence speed in terms of the time an agent community needs to reach an emergent state. The distribution of convergence time and the effects of various parameters on the convergence time were systematically investigated. Table 4 summarizes the main findings of the experiments, where ↑ means increasing and ↓ means decreasing.

TABLE 4
RELATIONSHIPS BETWEEN SYSTEM PARAMETERS AND CONVERGENCE TIMES

Parameter	Condition	Convergence Time	
		Maturity	Optimality
Organizers # ↑		↑	↑
Members # ↑	m≤k	↓ ↑	↓ ↑
	m>k	↓	↑ ↓
Categories # ↑		↑ ↓	↓ ↑ ↓
Size of category ↑		↑	↑

For the future work, we will further investigate more complicated types of agent community systems as well as other emergent systems. In general, we are investigating how experiment method can be better combined with formal analysis method in the development of complex system and their emergent behaviors, for example, to discover the intrinsic laws of emergent behaviors and to facilitate the formal specification and reasoning about emergent behaviors. Another topic that worth further study is to prove the experimental results based on a probabilistic mathematical model of agent communities.

ACKNOWLEDGEMENT

The work reported in the paper is partially supported by the Basic Research Program (973 Program) of China Ministry of Science and Technology under grant 2005CB321802.

REFERENCES

[1] Holland, J. H.: *Emergence: From Chaos to Order*. Oxford University Press, 1998.

[2] Johnson, S.: *Emergence*. Penguin Books, 2001.

[3] Fromm, J.: *Types and Forms of Emergence*. Arxiv preprint nlin.AO/0506028, arxiv.org, 2005. Available online at URL: http://arxiv.org/ftp/nlin/papers/0506/0506028.pdf, accessed on 19 March 2007.

[4] Fromm, J.: *The Emergence of Complexity*. Kassel University Press, 2004.

[5] Dorigo, M., Stutzle, T.: *Ant Colony Optimization*. The MIT Press, 2004.

[6] Moukas, A.: Amalthaea: information discovery and filtering using a multi-agent evolving ecosystem. *Journal of Applied Artificial Intelligence,* 11(5), pp.437-457, 1997.

[7] Walsh, W.E., Wellman, M.P., Wurman, P.R., MacKie-Mason, J.K.: Some economics of market-based distributed scheduling. *Proc. of 18th Int'l Conf. on Distributed Computing Systems*, pp. 612-619, 1998.

[8] Wang, F., Sun, Y., Ghanea-Hercock, R.: Synaptic connection autonomic network. *Proc. of the EURESCOM Summit on Ubiquitous Services and Applications*, pp. 25-34, 2005.

[9] Fromm, J.: *On Engineering and Emergence*. Arxiv preprint nlin.AO/0601002, arxiv.org, 2006. Available online at URL: http://arxiv.org/abs/nlin.AO/0601002, accessed on 19 March 2007.

[10] De Wolf, T., Samaey, G. and Holvoet, T.: Engineering self-organising emergent systems with simulation-based scientific analysis. *Proc. of the Fourth Int'l Workshop on Engineering Self-Organising Applications*, Brueckner, S. and Di Marzo Serugendo, G. and Hales, D. and Zambonelli, F. (eds.), Universiteit Utrecht, 2005.

[11] De Wolf, T. Samaey, G. Holvoet, T. Roose, D.: Decentralised autonomic computing: analysing self-organising emergent behaviour using advanced numerical methods. *Proc. of 2nd Int'l Conf. on Autonomic Computing*, pp. 52-63, 2005.

[12] StarLogo, http://education.mit.edu/starlogo/, accessed on 30 May, 2007.

[13] Zhu, H.: Formal reasoning about emergent behaviors of MAS. *Proc. of 17th Int'l Conf. on Software Engineering and Knowledge Engineering*, pp. 280-285, 2005.

[14] Duda, R.O., Hart. P.E.: *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[15] Lu S.Y., Fu, K.S.: A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 8, pp.381-389, 1978.

[16] Iamnitchi, A., Ripeanu, M., Foster I.: Small-world file-sharing communities. *INFOCOM 2004*, Hong Kong, 2004.

[17] Babaoglu, O., Meling, H., Montresor, A.: Anthill: A framework for the development of agent-based peer-to-Peer systems. *Proc. of 22nd Int'l Conf. on Distributed Computing Systems*, pp. 15, 2003.

[18] Wang, F.: Self-organising communities formed by middle agents. *Proc. of 1st Int'l Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 1333-1339, 2002.

[19] Cid-Sueriro, J., Wang, F.: A scalability analysis of self-organizing agent communities. *Proc. of Int'l Conf. on Learning*, 2002.

[20] Zhu, H.: A formal specification language for agent-oriented software engineering. *Proc. of 2nd Int'l Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pp.1174-1175, 2003.

[21] Zhu, H.: SLABS: A formal specification language for agent-based systems. *Int'l Journal of Software Engineering and Knowledge Engineering*, 11(5), pp.529-558, 2001

[22] Zhu, H.: Formal specification of evolutionary software agents. *Proc. of Int'l Conf. on Formal Engineering Methods*, LNCS 2495, pp.249-261, 2002.

[23] Zhu, H.: The role of caste in formal specification of MAS. *Proc. of Pacific Rim Int'l Workshop on Multi-Agents*, LNCS 2132, pp.1-15, 2001.

[24] Zhu, H., Wang, F.: Formal analysis of emergent behaviors of autonomous agent communities in scenario calculus. *Submitted to Journal of AAMAS*.