On the Significance of Synchroneity in Emergent Systems

Adam Campbell
School of EECS
University of Central Florida
Orlando, FL 32816-2362
acampbel@cs.ucf.edu

Annie S. Wu
School of EECS
University of Central Florida
Orlando, FL 32816-2362
aswu@cs.ucf.edu

ABSTRACT

The goal of this paper is to explore the effects of synchronization on distributed decision making processes. In particular, we examine the dynamics of a spatially distributed multi-agent system where agents use local information to make role assignments. By investigating several role assignment procedures for this problem, we find that, in general, system stability increases as the number of agents that make a decision at any particular time decreases. This result is promising, because in a physical, distributed system the time at which agents make their decisions would most likely not be synchronized. Although the two decision making procedures examined in this paper are similar, their dynamics with respect to synchronization are very different. One shows a linear relationship between synchronization and system behavior, whereas a non-linear relationship is seen with the second method. We demonstrate the significance of synchroneity on the dynamics of these complex systems and argue that it should be taken into account when studying the behaviors of multi-agent systems that utilize emergent coordination.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.6.0 [**Artificial Intelligence**]: Simulation and Modeling—*General*

General Terms

Design, Experimentation

Keywords

Emergent coordination, dynamics, synchroneity, simulation

1. INTRODUCTION

In the multi-agent systems subfield of Distributed Artificial Intelligence, emergent coordination refers to control methods that utilize local interactions and little to no inter-agent communication [5]. Because the emergent coordination methods use little to no communication and the agents do not require global information, they scale well to teams of large sizes. However, their reliance on the emergent properties that result from the interactions of a large number, possibly thousands [7], of agents makes their behavior difficult to predict and formally model [5, 6]. Lerman *et al.* [5] discuss how mathematical models can be useful tools for studying multi-robot systems as they can be used to complement or even supplement experimental analyses. In some cases, the mathematical models may be the only practical way to study systems if the simulations are computationally expensive [5]. Here, a mathematical model provides evidence suggesting that a distributed system can become unstable when too many agents act simultaneously.

Several biological studies have shown that variation in agents' behaviors can increase system-wide stability [3, 4]. Honey bees regulate their nest temperature by fanning hot air out of their nest when temperatures get too high and cluster together when temperatures get too low [4]. To prevent the nest temperature from continuously fluctuating around the desired state, each bee has a different threshold for when it starts fanning or clustering. If all bees had the same threshold, the hive would never be able to converge to the desired state. As another example, studies show that the differences in ants' tolerances to pheromone concentrations may result in a division of labor within the colony [3]. In these and similar systems, variation amongst the individuals is necessary to achieve the desired system dynamics.

In this paper, we show that variation in the time at which agents make their decisions can greatly affect the dynamics and performance of distributed systems. Huberman and Glance [2] came to a similar conclusion when comparing sequential and parallel updating schemes in a spatial Prisoner's Dilemma problem. Here, we use both mathematical and algorithmic models of the distributed process found in Hecker *et al.* [1] to take a more in-depth look at how synchroneity can affect system dynamics. Hecker *et al.* give experimental results showing how sensors in a spatially distributed network can use local information to make decisions about what particular range of transmission frequencies to monitor. The goal is to have an equal number of sensors monitoring each one of the frequency ranges, and the problem is made difficult because each sensor only has limited knowledge about the team's composition. Because of its generality and simplicity, this problem is used to explore the effects of synchronization on the dynamics of distributed decision making processes.

A mathematical model of Hecker *et al.*'s system predicts that the system stabilizes when few agents are making decisions simultaneously, but there is a point at which the syn-

Cite as: On the Significance of Synchroneity in Emergent Systems, Adam Campbell, Annie S. Wu, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 449–456

chroneity becomes so high that the system constantly overshoots the desired role distribution and is not able to settle on a stable configuration. Using several observations about the original role updating procedure we present a modified version with higher stability across a wider range of synchronization values. Although the original and proposed decision making processes are similar, they are affected very differently by synchronization. With the former, there is a linear relationship between synchronization and system behaviors; whereas, with the latter we see a phase transition in behavior when synchronization becomes very high. The main contribution of this paper is to show the significant role that synchronization can play in the dynamics of systems that utilize emergent coordination.

2. THE PROBLEM

The problem consists of M agents arranged in a toroidal grid, and each agent is in one of N different roles at any given time. The goal of the team is to reach and maintain an equal distribution of roles, i.e., the attendance of each role should be as close to $\frac{M}{N}$ as possible.

Using the information gathered from their local neighborhood, agents attempt to determine what role to switch to in order to move the system closer to the equidistribution state. To simulate synchroneity, each agent makes a role assignment decision each time step with probability ϵ. Thus, on average, $M * \epsilon$ agents are making a decision each time step. Below we describe three ways the agents can use the information from their local neighborhood to make their role assignment decisions.

2.1 Method 1

Method 1 is the role updating procedure found in Hecker *et al.* [1]. To update its role, an agent begins by recording the roles of each of its neighbors. An agent takes into account its own role when gathering these counts. Next, the agent switches to the role that has the lowest count. If multiple roles have the lowest count, then the agent switches to one of these roles with uniform probability.

An agent in $role_i$ could find that there are less $role_i$ agents in its neighborhood than any other role and choose to stay in $role_i$. Also, an agent in $role_i$ could see the same number of $role_i$ and $role_j$ agents and choose to switch to $role_j$. There is no reason for the agent to perform this action since it does nothing to change the role distribution of the agent's neighborhood. Method 2 does not allow this unnecessary role changing to occur.

2.2 Method 2

With low synchroneity, Method 1 was shown to hover around the equidistribution of roles but never settled in on a stable configuration [1]. This behavior is undesirable, especially if it is costly to update the role of an agent. This problem can be eliminated by making two simple changes to the original procedure: First, agents do not change their role if their own role is one of those with the fewest count. This change reduces the amount of unnecessary role changes that agents make. Second, agents do not take into account their own role when obtaining the role counts.

Let us look at an example to understand why the second change leads to more stable behavior. Let $\langle a_1, a_2, ..., a_N \rangle$ be the role count of an agent when it takes its own role into account, i.e., a_i is the number of neighbors, including itself, of type $role_i$ for some arbitrary agent. Without loss of generality, assume the agent to be in $role_1$. For each of the following scenarios, the action and resulting role counts are given as:

1. $a_1 \leq a_i$, $2 \leq i \leq N$; stay in $role_1$; $\langle a_1, a_2, ..., a_N \rangle$,
2. $a_1 = a_i + 1$, for some $2 \leq i \leq N$; switch to $role_i$; $\langle a_1 - 1, ..., a_i + 1, ..., a_N \rangle$, and
3. $a_1 > a_i + 1$, $2 \leq i \leq N$; switch to some $role_i$; $\langle a_1 - 1, ..., a_i + 1, ..., a_N \rangle$.

In Scenario 1, there are no more $role_1$ agents than any other role, so the agent remains in $role_1$. In Scenario 3, there are at least two more $role_1$ agents than any other role, so the agent will switch to one of the roles with the lowest count. In both scenarios, the agent does its best to maintain an equidistribution of roles amongst it and its neighbors. In Scenario 2, the agent switches roles, but the switch does not cause the distribution to get closer to the equidistribution, i.e., because a_1 equals $a_i + 1$, an agent switching from $role_1$ to $role_i$ will only cause the values of a_1 and a_i to swap, and thus, get no closer to the equidistribution.

An agent's role count becomes $\langle a_1 - 1, a_2, ..., a_N \rangle$ when it does not take its own role into consideration. Substituting $a_1 - 1$ for a_1 and combining Scenarios 2 and 3 from above, the following scenarios are possible:

1. $a_1 \leq a_i + 1$, $2 \leq i \leq N$; stay in $role_1$; $\langle a_1, a_2, ..., a_N \rangle$, and
2. $a_1 \geq a_i + 2$, $2 \leq i \leq N$; switch to some $role_i$; $\langle a_1 - 1, ..., a_i + 1, ..., a_N \rangle$.

Now, the agent only switches roles when there are at least two more $role_1$ agents than some other role. Thus, the agent does not make the mistake of switching roles when the switch does not change the overall role distribution in its local neighborhood.

2.3 Method 3

One way to achieve the goal of an equidistribution of roles is to simply have each agent randomly choose its role with equal probability. This method requires no communication between agents but does result in a large amount of role switching. Method 3 is used as a baseline comparison for the other two methods.

2.4 Mathematical model

A mathematical model of Method 1 provides evidence for the significant effects that synchronization can have on a distributed multi-agent system. To begin the analysis, we define the following variables:

- M: number of agents
- N: number of roles
- t: current time step
- $role_i$, $1 \leq i \leq N$: used to refer to a specific role in the system
- $m_i(t)$, $1 \leq i \leq N$: number of $role_i$ agents at time t, $\sum m_i(t) = M$

- $p_i(t) = \frac{m_i(t)}{M}$, $1 \leq i \leq N$: proportion of $role_i$ agents at time t
- μ: number of neighbors each agent has
- ϵ: probability that an agent makes a decision each time step

Equation 1 gives the expected proportion of $role_i$ agents at time $t+1$ based on the distribution of roles at time t:

$$q_i(t+1) = (1-\epsilon)p_i(t) + \epsilon z_i(t+1). \quad (1)$$

The first part of Equation 1 gives the proportion of $role_i$ agents that do not update their role, and the second part gives the number of agents that change to $role_i$. Note, the second part includes agents that are in $role_i$ that choose to "switch to" $role_i$.

The number of agents that choose to switch to $role_i$ at time $t+1$ is proportional to the number of agents that see less $role_i$ agents in their neighborhood at time t than any other role. Equation 2 returns the probability that an agent sees no more $role_i$ neighbors than any other type and is defined as

$$z_i(t+1) = \sum_{a_i=0}^{min(\mu,m_i(t))} \sum_{\forall j \neq i, a_j = a_i}^{min(\mu,m_j(t))} \Omega, \quad (2)$$

where

$$\Omega = \delta\left(\mu - \sum_{k=1}^{N} a_k\right) \frac{\psi(a_1, a_2, ..., a_N, t)}{\gamma(a_i, a_i, a_{j.1}, ..., a_{j.N-1})}.$$

The summations in Equation 2 loop through all possible combinations of values where the number of $role_i$ agents is less than or equal to all of the other role counts. In Equation 2, δ is Kronecker's delta function and is defined as

$$\delta(x) = \begin{cases} 1, \text{ if } x = 0 \\ 0, \text{ if } x \neq 0 \end{cases}.$$

It is used to ensure that the sum of all a_i values in Equation 2 is equal to the number of neighbors, μ.

Equation 2 enumerates all possible combinations of N-tuples that sum to μ and have the first value less than or equal to all other values in the tuple, i.e., it enumerates all possible role counts that could be obtained where there are no more $role_i$ agents than any other role. Equation 3 calculates the probability of that combination of values occurring given the current distribution of roles:

$$\psi(a_1, a_2, ..., a_N, t) = \prod_{i=1}^{N} \rho(a_i, m_i(t), \sum_{j=1}^{i-1} a_i), \quad (3)$$

where

$$\rho(a, m, s) = \binom{\mu - s}{a} \prod_{i=0}^{a-1} \frac{m-i}{M-s-i}$$

gives the total probability of seeing a agents of a given role if there are a total of m of them in the remaining population. We say remaining population because if we were to take out 3 agents of $role_1$ from a population of 100 then there are only 97 agents left to choose from when calculating the probability of seeing $role_2$ agents. An example is given below to clarify these procedures.

Equation 4 returns the number of a_i values that are equal to the target value τ and is defined as

$$\gamma(\tau, a_1, a_2, ..., a_N) = \sum_{i=1}^{N} \delta(\tau - a_i). \quad (4)$$

Equation 4 is used to prevent over counting when calculating $z_i(t+1)$. When computing $z_i(t+1)$, there will be combinations of neighbors where the number of $role_i$ agents is equal to the number of $role_j$ agents, and so the agent would have equal probability of switching to $role_i$ or $role_j$. Therefore, when the probability of that configuration is calculated, it needs to be divided by 2 to prevent over counting.

Finally, we calculate the new distribution of roles:

$$m_i(t+1) = \lfloor M * q_i(t+1) \rfloor. \quad (5)$$

Equation 5 does not guarantee $\sum m_i(t) = M$, and so, the roles of the remaining $M - \sum m_i(t)$ agents are chosen with probability directly proportional to $q_i(t+1)$.

2.4.1 An example

An example is given to clarify the analysis above. For the example, $M = 100$, $N = 4$, $\mu = 4$, the current number of agents in each role are $m(t) = (20, 30, 15, 35)$, and the goal is to determine the value of $z_1(t+1)$.

First, we must determine all valid neighborhood configurations where there are no more $role_1$ agents than any other role. To refer to a neighborhood configuration, a string of four numbers will be used, e.g., $\langle 0, 0, 1, 3\rangle$ refers to a neighborhood configuration where there are zero $role_1$ and $role_2$ agents, one $role_3$ agent, and three $role_4$ agents. The configuration $\langle 1, 0, 1, 2\rangle$ would not be used when calculating $z_1(t+1)$ because there are less $role_2$ agents than $role_1$ agents. Also, the configuration $\langle 0, 1, 2, 2\rangle$ would not be valid because the four numbers do not sum to $\mu = 4$.

The configurations used when calculating $z_1(t+1)$ are $\langle 0, 0, 0, 4\rangle$, $\langle 0, 0, 4, 0\rangle$, $\langle 0, 4, 0, 0\rangle$, $\langle 0, 0, 2, 2\rangle$, $\langle 0, 2, 0, 2\rangle$, $\langle 0, 2, 2, 0\rangle$, $\langle 0, 1, 1, 2\rangle$, $\langle 0, 1, 2, 1\rangle$, $\langle 0, 2, 1, 1\rangle$, $\langle 0, 0, 1, 3\rangle$, $\langle 0, 1, 0, 3\rangle$, $\langle 0, 1, 3, 0\rangle$, $\langle 0, 0, 3, 1\rangle$, $\langle 0, 3, 0, 1\rangle$, $\langle 0, 3, 1, 0\rangle$, $\langle 1, 1, 1, 1\rangle$. For each configuration, we calculate the probability of that configuration occurring, i.e., we need to calculate ψ from Equation 3 for each of the sixteen configurations previously listed.

For brevity, only the details of calculating the probability of seeing a $\langle 0, 2, 0, 2\rangle$ neighborhood configuration are presented in detail. The probability of seeing exactly zero $role_1$ and $role_3$ agents, and two $role_2$ and $role_4$ agents is $\frac{30}{100} * \frac{29}{99} * \frac{35}{98} * \frac{34}{97}$ multiplied by the number of ways that configuration could have been achieved. Let us refer to an agent's neighborhood as including n_{up}, n_{down}, n_{left}, and n_{self}. One valid configuration the four neighbors could be in to give a count of $\langle 0, 2, 0, 2\rangle$ is $n_{up} = role_2$, $n_{down} = role_4$, $n_{left} = role_4$ and $n_{self} = role_2$. Table 1 shows the six valid states that the neighbors could be in to make a $\langle 0, 2, 0, 2\rangle$ neighborhood configuration. The reason for the six configurations is because there are $\binom{4}{0} * \binom{4}{2} * \binom{2}{0} * \binom{2}{2} = 1 * 6 * 1 * 1$ combinations. As another example, there are $\binom{4}{0} * \binom{4}{1} * \binom{3}{1} * \binom{2}{2} = 1 * 4 * 3 * 1 = 12$ combinations for configuration $\langle 0, 1, 1, 2\rangle$. Note that zero appears twice in the configuration $\langle 0, 2, 0, 2\rangle$ (for $role_1$ and $role_3$), and so, the $\langle 0, 2, 0, 2\rangle$ configuration would also be taken into account when calculating $z_3(t+1)$. To prevent over counting the $\langle 0, 2, 0, 2\rangle$ probability, $\psi(0, 2, 0, 2, t)$ is divided by 2 (Equation 4). Thus, the final total for the

n_{up}	n_{down}	n_{left}	n_{self}
$role_2$	$role_2$	$role_4$	$role_4$
$role_2$	$role_4$	$role_2$	$role_4$
$role_2$	$role_4$	$role_4$	$role_2$
$role_4$	$role_2$	$role_2$	$role_4$
$role_4$	$role_2$	$role_4$	$role_2$
$role_4$	$role_4$	$role_2$	$role_2$

Table 1: The six possible states that the neighbors could be in to make up a neighborhood configuration of $\langle 0, 2, 0, 2\rangle$.

$\langle 0, 2, 0, 2\rangle$ configuration is: $\frac{30}{100} * \frac{15}{99} * \frac{35}{98} * \frac{34}{97} * 12 * \frac{1}{2}$. To compute $z_1(t+1)$, this value is added to the values obtained from the other fifteen configurations.

3. EXPERIMENTS AND RESULTS

We next perform an empirical exploration of the effects of synchroneity on the three role updating procedures. For each experiment, there are 100 agents arranged in a ten-by-ten toroidal grid, all of the agents are initially in $role_1$, and they can communicate with the four neighbors in their immediate neighborhood. Empirically, we found that using different initial role distributions does not change the overall results of the system, and so, for brevity, only one initial role distribution is used throughout the remainder of the paper.

Experiment 1 looks at the overall behavior of the three procedures over a wide range of ϵ values. Experiment 2 examines the average behavior of the role updating methods, whereas Experiment 3 explores the details of individual runs. Experiment 4 performs an in depth analysis of the relationship between stability and synchroneity, and Experiment 5 looks at the relationship between convergence time and synchroneity. Taken together, these experiments demonstrate the significant effects that synchroneity can have on distributed decision making processes.

3.1 Experiment 1

The first experiment examines the overall effect of synchroneity on system dynamics. We hypothesize that large fluctuations from the equidistribution will occur when synchroneity is high because agents will overcompensate for those roles with lower attendance. Conversely, when synchroneity is low, few agents will switch roles, and so, they will slowly converge on the correct distribution. By scanning through the full range of ϵ values at very small increments, we can get an idea of the range of values that produce oscillatory behaviors. For each value of ϵ between zero and one taken at $\frac{5}{1000}$ increments, one-hundred runs of the mathematical model and three methods are conducted for 1000 time steps. The attendance values for each one of the roles is recorded each time step after $t = 900$. Once these values are recorded, they are each plotted as a point along the y-axis for the respective value of ϵ.

3.1.1 Mathematical model

Figure 1 shows the plots obtained when $N = 4$ and $N = 6$ roles for the mathematical model. The left-most plots of Figure 1 (similarly Figures 2, 3, and 4) show the attendance values for $role_1$, and the right-most plots show the attendance values for $role_2$. Because the initial attendances for the roles other than $role_1$ are zero, the values obtained from them will be similar, and so, the remaining role attendances are not shown.

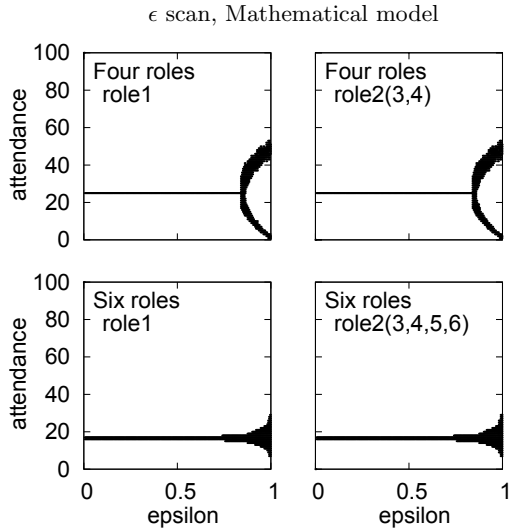

Figure 1: Attendance values obtained at the end of one-hundred runs of the mathematical model.

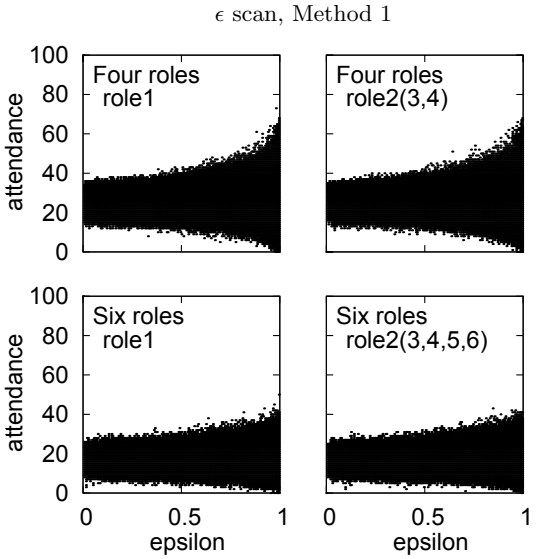

Figure 2: Attendance values obtained at the end of one-hundred runs when using Method 1.

For both four and six roles, the mathematical model clearly shows a point at which the system is unable to settle on the desired role distribution. With high synchroneity, the agents overcompensate for deviations from the equidistribution and constantly fluctuate around the desired state. As synchroneity increases, so, too, do these fluctuations.

3.1.2 Methods 1 and 2

Figures 2 and 3 show the same experiments for Method 1 and Method 2, respectively. The range of attendance values increases as synchroneity increases when Method 1 is used (Figure 2). With Method 1, the mean attendance value appears to be close to the desired value, but the devi-

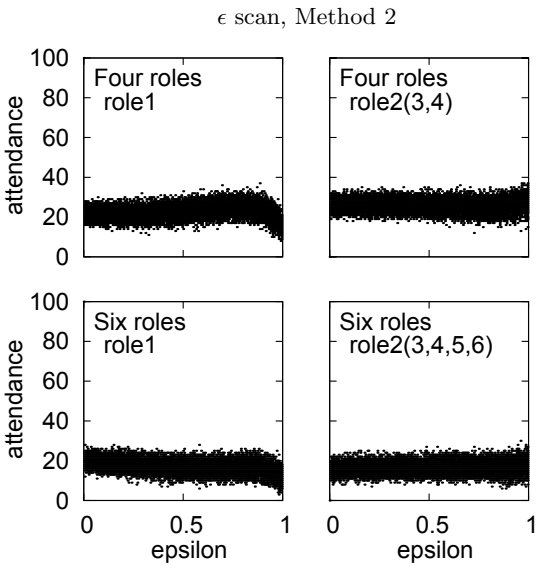

Figure 3: Attendance values obtained at the end of one-hundred runs when using Method 2.

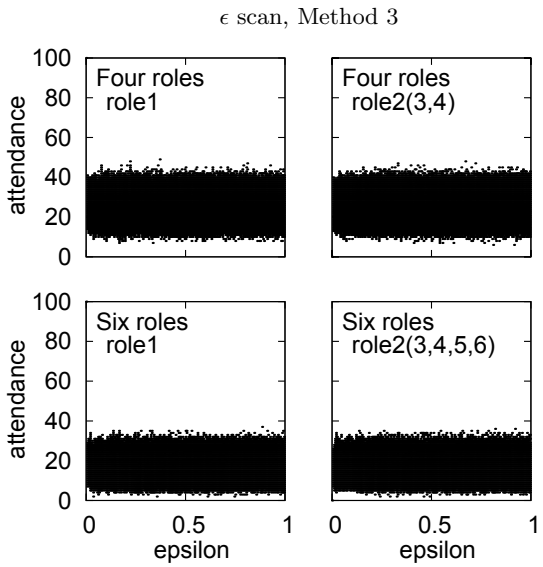

Figure 4: Attendance values obtained at the end of one-hundred runs when using Method 3.

ation increases as synchronization increases. Although the mathematical model is based on Method 1, Figures 1 and 2 show a clear difference in behavior between the two procedures. In the mathematical model, the spatial relationships are essentially ignored since the calculations determine the expected, average behavior of the agents. Spatial relationships can have a significant affect on the dynamics of a simulated system [8].

With Method 2, the mean decreases for $role_1$ when synchroneity is high, but the deviation from the mean does not seem to be affected by the level of synchroneity in the system (Figure 3). As synchroneity increases, the average attendance value for $role_1$ decreases. When synchroneity is very high, all (or most) of the agents will switch out of $role_1$ in the first time step. Because of the stability of Method 2, some of the agents will not switch from their role after the first time step, leaving a number of individuals in a role other than $role_1$ for the remainder of the run. Thus, on average, there will be less $role_1$ agents than other roles when synchroneity is at or near 1.0.

3.1.3 Method 3

Can the randomness in the choices the agents make be the sole reason for the behavior seen in Figures 2 and 3? If agents ignore the roles of their neighbors and choose a role at random, would we see the same dynamics? Figure 4 shows the experiments for Method 3, and we see that the dynamics are not the same as in Figures 2 and 3.

The way the agents use the information from their local neighborhood certainly has an effect on the dynamics of the system. When synchroneity is low, the range of values for Method 1 is lower than that of Method 3; however, as synchroneity increases, the range of attendance values is actually larger than the range produced by Method 3. With Method 2, the information is used more appropriately, and the range of attendance values is not significantly different for any value of ϵ.

3.2 Experiment 2

The previous experiments only show the attendance values that were obtained over the last 100 time steps of the runs; however, they do not show how the system behaves over time. Figures 5 and 6 show the behavior of the first 1000 time steps averaged over 1000 runs for $\epsilon = 0.01$, 0.5, and 1.0 for Method 1 and Method 2, respectively. The standard deviation is shown every 75 time steps.

With Methods 1 and 2, the agents slowly reach an equidistribution of roles when $\epsilon = 0.01$. On average, only one agent is updating its role each time step, and so, large fluctuations from the equidistribution are avoided. When $\epsilon = 0.5$, the agents quickly reach the desired distribution and do not deviate far from the appropriate attendances. There does not seem to be significant difference in the *average* behaviors between Methods 1 and 2 when $\epsilon = 0.01$ and 0.5.

When $epsilon = 1.0$ a clear difference in behavior is seen between the two methods. With both methods, oscillations in attendance occur, but with Method 1 these oscillations dampen and the attendance values begin to average the desired state. With Method 2, the system continuously switches between two configurations and is never able to get out of this cycle. Although the average behavior shows convergence to the desired state with Method 1, the standard deviations are so high, that for any single run, the system may be far from the equidistribution.

For Method 1, the average standard deviation throughout the runs is 3.010 when $\epsilon = 0.01$, 3.943 when $\epsilon = 0.5$, and 18.396 when $\epsilon = 1.0$. For Method 2, the average standard deviation is 2.525 when $\epsilon = 0.01$, 2.734 when $\epsilon = 0.5$, and 3.854 when $\epsilon = 1.0$. In general, the variance between runs increases as ϵ increases. This trend is more noticeable for Method 1 than for Method 2.

Plots for Method 3 are not shown here because there is no significant difference between the behaviors of the runs for the three values of ϵ. For small values of ϵ, the system takes longer to reach the equidistribution of roles, but after it has been reached, the average and standard deviations for

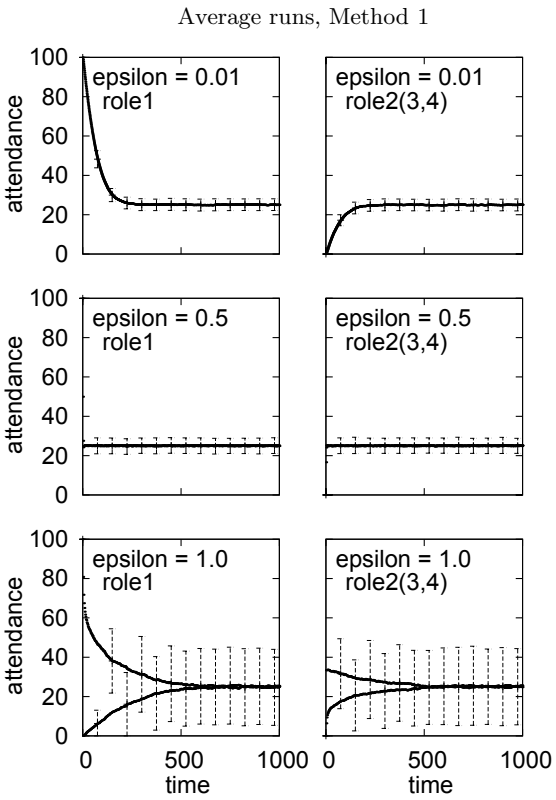

Figure 5: Attendance values averaged over 1000 runs for Method 1.

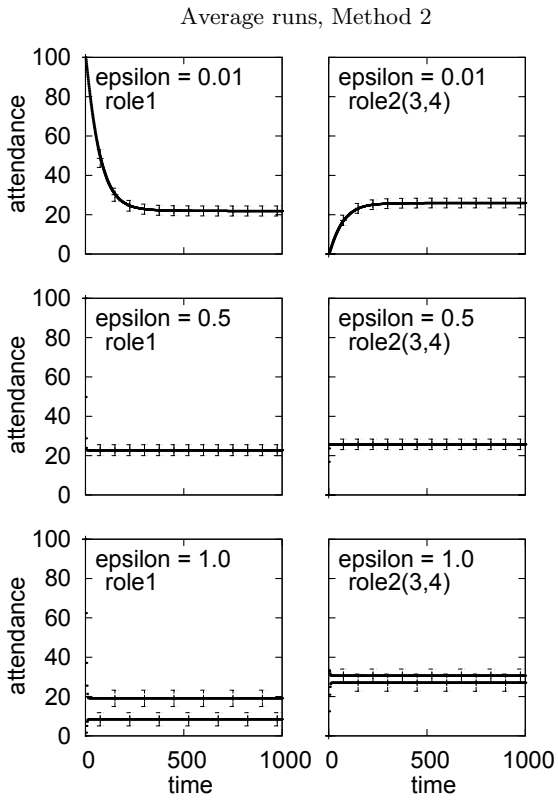

Figure 6: Attendance values averaged over 1000 runs for Method 2.

the attendance values are similar for the three values of ϵ. Because the agents do not use the information from their local neighborhood with Method 3, synchronization plays little to no role in the long-term dynamics of the system.

3.3 Experiment 3

Lost in the previous two sets of experiments are the detailed dynamics that can only be seen by viewing individual runs. For the original and proposed role allocation procedures, a run is conducted for $\epsilon = 0.01$, 0.05, and 1.0. Figures 7 and 8 show a single run for each value of ϵ for Method 1 and Method 2, respectively. A large amount of role switching takes place when Method 1 is used. Even with very low synchroneity, the agents are not able to settle in on a stable configuration. Method 2 allows the agents to settle in on a single stable configuration, except for when $\epsilon = 1.0$.

To determine why the agents oscillate between two attendance counts with Method 2 and $\epsilon = 1.0$, we went through several runs, and noticed the following: Occasionally, there are pairs of neighboring agents that are in the same role and have the same neighborhood configuration. These $role_i$ agents both see that there are no $role_j$ agents in their neighborhood and switch to $role_j$. In the next time step, they both see no $role_i$ neighbors, so they switch to $role_i$. The only way for them to break out of this cycle is for one of them to not act, and this event becomes less likely as ϵ approaches 1.0.

3.4 Experiment 4

If agents have to expend energy to switch roles, then one of the goals of the system designer would be to minimize the amount of times agents change their role. Experiment 4 is used to determine the average amount of role switching that occurs during each time step for Methods 1, 2, and 3. For this experiment, 100 runs are conducted for each ϵ at $\frac{5}{1000}$ intervals. The number of agents that switch roles during each time step between steps 900 and 1000 of these 100 runs is recorded and averaged.

Figure 9 shows the results for the three methods. With Method 3, approximately $\epsilon M(1 - \frac{1}{N})$ agents switch their role each time step. On average, ϵM agents will decide to change their role, and because $\frac{1}{N}$ of those agents will randomly choose their own role, only $\epsilon M(1 - \frac{1}{N})$ agents actually change roles. Thus, even when $\epsilon = 1$, the random procedure will not switch the role of every agent.

Surprisingly, the same is not true for Method 1. The number of agents that switch their role is directly proportional to ϵ. In the worst case, when $\epsilon = 1.0$, every agent is switching its role every time step. Clearly, this behavior is undesirable as agents would expend large amounts of energy switching roles. Although we can control the amount of role switching by lowering ϵ, the system is never able to reach a stable configuration where agents will not choose to change their role when given the opportunity.

When using Method 2, agents are able to settle into their roles. Only when synchroneity is very high ($\epsilon > 0.995$) are the agents not able to settle in on a stable configuration. With six roles, the system always reached a stable configuration, regardless of ϵ. As the number of roles increases, the number of stable neighborhood configurations also increas-

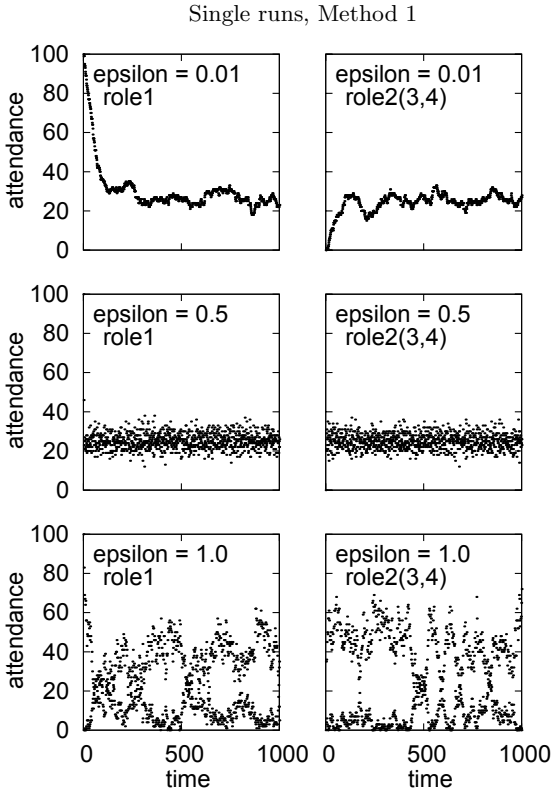

Figure 7: Attendance values for a single run for Method 1.

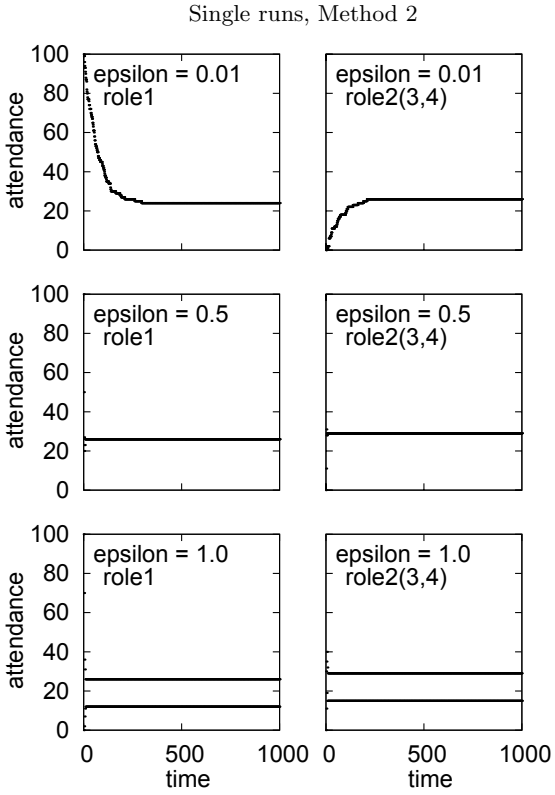

Figure 8: Attendance values for a single run for Method 2.

es. Thus, agents are less likely to switch roles as the number of roles increases, and the system will be more stable.

A linear relationship between ϵ and the number of role switches is seen with Method 1, but the same is not true with Method 2. Only in the most extreme cases does synchronization have an effect on Method 2. Figure 10 shows the values of ϵ between 0.995 and 1.0 at $\frac{1}{100000}$ increments with Method 2 and four roles. A sharp transition in system dynamics occurs at the highest levels of synchronization. For a majority of the values of ϵ, the agents are able to settle in on a single, stable configuration, but when all, or most, of the agents are working simultaneously they cannot reach a stable configuration. Although the two updating procedures are similar, they exhibit very different behaviors under different levels of synchroneity.

3.5 Experiment 5

Experiment 5 examines the average amount of time required for the system to converge with Method 2. For each value of ϵ at $\frac{5}{1000}$ increments, 1000 runs are conducted for both four and six roles. We say that the system has converged if every agent determines that it would not switch its role if given a chance to. The time at which the networks converge is averaged over 1000 runs for each value of ϵ. If any of the 1000 runs do not converge within 10000 time steps for some particular value of ϵ, the data at that point is not plotted. This only occurred with four roles and $\epsilon = 1.0$.

Figure 11 shows the average amount of time required for the systems to converge for four and six roles. When ϵ is small, the system takes a long time to converge because a small number of agents make a decision each time step. As ϵ increases, the system is able to converge faster. However, there is a point at which the time to converge starts to increase again because of the neighboring-agent problem described in Section 3.3. The increase occurs for both four and six roles, but because there are more stable configurations when six roles are used, the increase is not as significant.

4. CONCLUSIONS

Emergent coordination techniques for controlling the behaviors of distributed multi-agent systems are attractive to scientists and engineers because of their simplicity, scalability, and reliability. Agents communicate with only the agents in their local neighborhood (or not at all) and do not require global information when making decisions. Also, because the systems are distributed, there is no single point of failure, and agents can self adjust if members of their team can no longer perform their tasks. However, formally modeling and predicting the behaviors of emergent systems can be difficult because of the non-linear system dynamics that result from the interactions of hundreds or even thousands of agents.

The goal of this paper is to show the significant role synchronization can play in distributed, multi-agent decision making systems. Empirical results show that the level of synchronization has a large effect on the system dynamics, and thus, team's performance. High synchronization tends to cause difficulty in system convergence because the system becomes too reactive to fluctuations from the desired state. Low synchronization allows for a more reliable convergence to stable configurations, although time to converge increas-

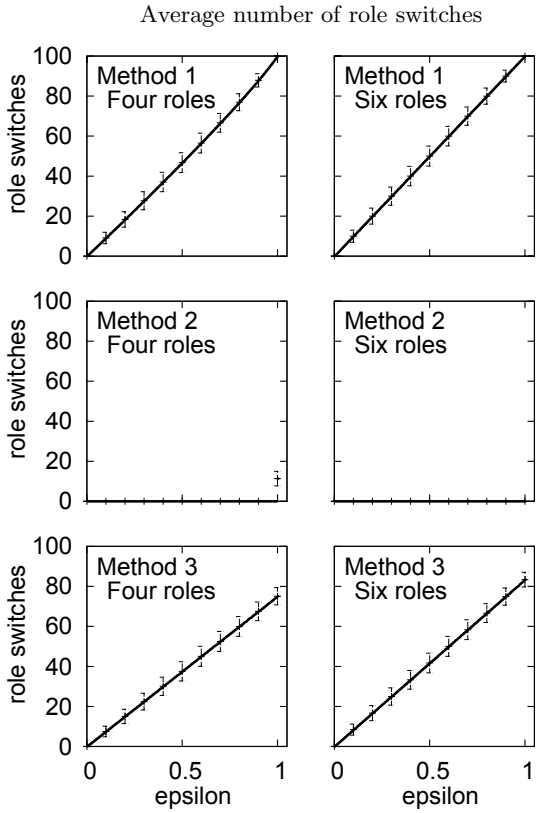

Figure 9: Average number of role switches for each of the three methods.

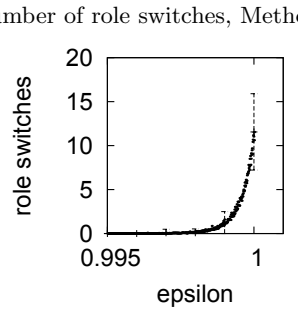

Figure 10: Average number of role switches for the highest values of ϵ with Method 2 and four roles.

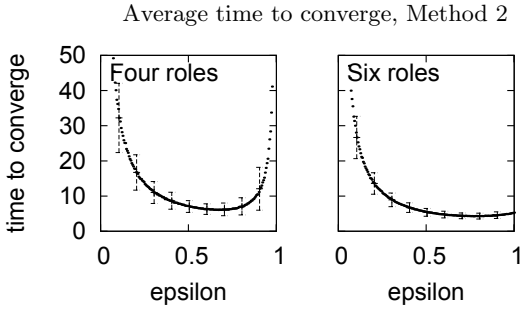

Figure 11: Average time required for the system to converge using Method 2 over all values of ϵ.

es as synchroneity decreases. With moderate amounts of synchroneity the system can quickly converge on, and not overshoot, the desired state.

The results also show that the effects of synchronization can be very different from one system to the next. We observe a linear relationship between the system dynamics and the level of synchronization when one role allocation procedure is used, whereas a sharp transition from stable to unstable behavior is observed when using a slightly different procedure. Designers of such distributed systems should be aware of the effects that various degrees of synchronization could have on their systems' dynamics and performance.

This paper opens up the possibility of future work to further explore the relationship between system dynamics and synchronization by investigating questions such as: How will a distributed system's dynamics be affected by the connectivity of the agents combined with the amount of synchronization in the system? The two examples shown in this paper illustrated both linear and non-linear relationships between system dynamics and synchronization, but are there more? Are there distributed systems that benefit from high levels of synchronization? This paper is a first step towards showing how important these and other questions are when studying or designing distributed decision making processes that utilize emergent coordination.

5. REFERENCES

[1] J. P. Hecker, A. S. Wu, J. A. Herweg, and J. John C. Sciortino. Team-based resource allocation using a decentralized social decision-making paradigm. In *Proc. of the SPIE, Evolutionary and Bio-inspired Computation: Theory and Applications*, 2008.

[2] B. A. Huberman and N. S. Glance. Evolutionary games and computer simulations. *Proceedings of the National Academy of Science*, 90(16):7716–7718, 1993.

[3] D. E. Jackson, S. J. Martin, F. L. W. Ratnieks, and M. Holcombe. Spatial and temporal variation in pheromone composition of ant foraging trails. *Behavioral Ecology*, 18(2):444–450, 2007.

[4] J. C. Jones, M. R. Myerscough, S. Graham, and B. P. Oldroyd. Honey bee nest thermoregulation: Diversity promotes stability. *Science*, 305:402–404, 2004.

[5] K. Lerman, C. Jones, A. Galstyan, and M. J. Mataríc. Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, 25(3):225–241, 2006.

[6] C. Rouff, A. Vanderbilt, M. Hinchey, W. Truszkowski, and J. Rash. Properties of a formal method for prediction of emergent behaviors in swarm-based systems. *Software Engineering and Formal Methods, 2004. SEFM 2004. Proceedings of the Second International Conference on*, pages 24–33, 2004.

[7] J. Seyfried, M. Szymanski, N. Bender, R. Estaña, M. Thiel, and H. Wörn. The I-SWARM project: Intelligent small world autonomous robots for micro-manipulation. *Swarm Robotics*, 3342:70–83, 2005.

[8] N. M. Shnerb, Y. Louzoun, E. Bettelheim, and S. Solomon. The importance of being discrete: Life always wins on the surface. *Proceedings of the National Academy of Science*, 97(19):10322–10324, 2000.