Social Networks for Effective Teams

Paul Scerri and Katia Sycara

Summary. In fields as diverse as sociology and physics researchers have been investigating the rich networks that exist in nature. More recently, a small number of multi-agent researchers have shown that the performance of a group can be significantly impacted by the nature of the network that connects them. In this chapter, we build on these initial efforts, performing systematic experiments in an attempt to understand how and why social networks affect group performance. Our key conclusion is that performance of a team can sometimes be improved by imposing a social network with relatively few connections even if it were feasible to connect the agents with a complete network.

1 Introduction

In a variety of important domains hundreds or thousands of heterogeneous agents are required to work together to achieve very complex goals. For example, for a large scale disaster response police, firefighters, paramedics and many others need to work together, albeit loosely, to mitigate the effects of the disaster. With current and future high speed network infrastructure any two members of the team could potentially communicate directly with one another. However, some recent work by Gaston and des Jardinss [5] has hinted that fully connected networks may not necessarily facilitate the most effective coordinated behavior. In this chapter, we explore this question in detail and attempt to quantify and explain the benefits of *not* having complete connectivity between all members of a large team.

In disciplines outside of artificial intelligence, including physics, economics, computer science and sociology, networks between people, *social networks*, have been extensively studied. For example, Milgram showed that people were often related into networks with a rich structural property called the *small worlds property* [15] and recent work has discovered that such structures exist not only between people but in a range of natural and man-made artifacts, including the Internet and electrical grids [7]. In a typical multiagent system, an agent can maintain connections with all other agents at negligible or no cost. This leads to most multiagent systems having completely connected communications graphs, although there are a small number of exceptions [5, 12].

However, the implicit assumption that complete networks are best for facilitating coordination, an assumption that may not be correct.

Coordination involves solving several intertwined problems in a distributed fashion. In this chapter, we first abstracted and isolated four key things that coordination must do and then compared a team's performance at those tasks using seven different types of networks, including a complete network. The four coordination problems were: (1) sharing information; (2) fusing information; (3) allocating tasks and; (4) developing an aggregate picture of the team state. We contend that these problems abstractly capture many of the things that a team must do. The seven types of network were: (1) complete; (2) lattice; (3) loop; (4) hierarchy; (5) random; (6) small worlds and; (7) scale free. The key result was that for only *one* of the problems did the complete network lead to the best performance. This result suggests that imposing a logical social network with relatively few links on a team of agents, even when a complete network is feasible, can improve performance in a variety of areas. Moreover, no particular network type was best for each of the problems, suggesting that networks might have to be carefully chosen for the situation.

Coordination requires that a team must address several problems in parallel. Our initial results showed that different networks had different performance for the different sub-problems. However, since no particular network was always best, choosing which network to use for overall coordination is not straight forward. In fact it is reasonable to hypothesize that the "best" network for a particular coordination application will depend on the relative frequency of the different coordination sub-problems and the relative importance of different performance metrics. In an abstract coordination simulator we tested networks with problems requiring relatively different emphasis on each sub-problem. In many cases complete networks turned out to be best although they were not best for sub-problems. However, this was largely due to an additional coordination not described above and very suited to complete networks. However when the coordination problem required relatively more information fusion, complete was not best.

In a cooperative team, there is no concern paid to whether one team member does more work than another. However, in practice, equitably sharing the workload can be important, perhaps for the morale or energy requirements of team members, for load balancing reasons or other reasons. While conducting experiments to understand the impact of networks on performance, a correlation was observed between the performance of a particular network and the disparity between the loads on individual agents. More particularly, the scale-free networks were often performing very well, but a small number of nodes with high degree, i.e., nodes with many links were performing a high proportion of the work required to coordinate. The high degree nodes were essentially allowing the team to centralize problem solving, which was leading to good performance even with algorithms designed to be distributed. If an equitable distribution of coordination load is important then the best "performing" network might not be the best choice.

2 Networks

A network N is a pair (A, E), where A is a set of agents, and E is a set of edges between the agents, $E = \{\{u, v\} | u, v \in A\}$. Many useful measurements of networks have been developed [3], but only two are important here: network *width* and the *degree distribution* of the network. The width of the network is the average maximum distance from any agent in the network to any other agent, where distance is the minimum number of edges that need to be traversed to get from one agent to the other. The width gives an indication of the maximum separation of the agents from one another. The number of edges that agent a is involved with is the *degree* of a, which we write $degree(a)$. The degree distribution of the network is frequency distribution of the number of edges involving each agent.

To understand the basic impact of a social network on coordination, we conducted experiments with seven different networks. The networks were chosen to be be representative of those used in the literature, but diverse enough to uncover interesting properties. Table 1 shows the networks used in the initial experiments and their width and degree distribution. The *Complete* network is most typically used in multiagent systems. The *Lattice* and *Loop* networks might be used when wireless or other communications limitations prevent more highly connected networks from being used. The *Hierarchy* is rarely used within the multiagent community but is a standard communications framework for human organizations. The *Random* network provides a sort of baseline low degree network. Finally, the *Small Worlds* and *Scale Free* networks have been shown to be very common types of networks in nature [16]. Figure 1 shows small examples of each of the network types.

Notice that we are making no assumption about the underlying physical network, which will in many cases be a complete network. In cases where an agent needs to direct by communication to another specific agent whose idenitity it knows, that communication does not need to go via the logical network described here.

3 Key Coordination Problems

In the following, we formally describe the coordination problem that must be solved by the team and that is the basis for this work.

Agents, $A(t) = \{a_1, \ldots, a_k\}$, are cooperating on a joint goal. Information, $I = \{i_1, \ldots, i_n\}$, are discrete pieces of information. Some, $I_o \subseteq I$, are able to be sensed, at various times, in the environment. The predicate $Observable(i, t)$ returns *true* if it is possible for some agent to observe information i at time t and returns false otherwise. Some information, e.g., location of a fixed resource, will be always observable, while others, e.g., events, will be observable for short periods. There is no implication that something will be sensed, even if it can be. Other information can only be

Name	Description	Degree Dist.	Width
Complete	Each agent is connected to each other agent	All = 1	1
Lattice	Agents are arranged into 2D grid	All = 4, except edges	$\sqrt{\|A\|}$
Loop	Agents arranged in a circle	All = 2	$\frac{\|A\|}{2}$
Hierarchy	Traditional tree, branching factor = 3	All = 4	$\log_n \|A\|$
Random	Each new agent makes 2 connections to other agents	Average is 4	$\log_n \|A\|$
Small World	Loop plus each agent makes one random connection	Average is 4	$\log_n \|A\|$
Scale Free	New agents connect to 3 existing with probability proportional to number of connections a node already has	Exponential	$\log_n \|A\|$

Table 1. The seven different types of network investigated in this chapter.

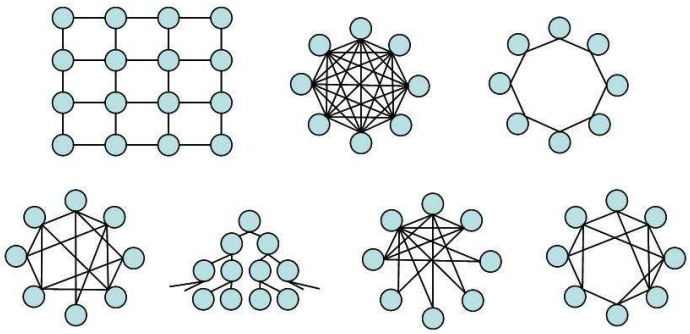

Fig. 1. Examples of the seven network types. Clockwise from top left: Lattice, Complete, Loop, Small World, Scale Free, Hierarchy, Random.

inferred by fusing observable information. i_f can only be inferred from $I_F \subseteq I$ and, hence, must be inferred by some agent $Knows(a,t) \subset I_F$. Some subset of the information is $Knowable(t) \subseteq I$ but an agent might only know some of the information, $Knows(a,t) \subseteq Knowable(t)$. $i_k \in Knowable(t)$ if $\exists t` \leq t, Observable(i_k, t') \vee \forall j \in I_K \exists t` \leq t, Observable(j, t')$

The team $A(t)$ is attempting to achieve a high level goal G, which is broken into discrete sub-tasks $\alpha_1, \ldots, \alpha_n$, typically performed by individuals. A subtask, $alpha_i$ is applicable when the predicate $Applicable(I_{\alpha_i}), I_{\alpha_i}) \subseteq I$ is *true*.

For some tasks, agents require sharable resources. These resources, $R(t) = \{r_1, \ldots, r_m\}$, are a dynamically changing set of available resources. We assume

that the resources are discrete and non-consumable. Agent a has exclusive access to $Holds(a) \subseteq R$. $\forall a, b \in A, a \neq b, Holds(a) \cap Holds(b) = \emptyset$.

Agents must perform the individual tasks, when they are applicable for the team to receive reward. The reward received by the team when an agent performs a task is a function of the agent and task as well as what the agent knows and the resources it has. Specifically:

$$Reward(a, \alpha, Knows(a), Holds(a), t) \rightarrow \mathcal{R}$$

Notice that unless $I_{\alpha_i} \subseteq Knowable(t)$, $Reward(\cdot) = 0$, i.e., unless the task is currently applicable the team gets no reward for performing it.

We specifically distinguish between *necessary* and *useful* resources. Define $IR_i \subseteq R$ as a set of substitutable resources. Necessary resources are those where IR_i^* if $Holds(a) \cap IR_i^* = \emptyset$ then $Cap(a, \alpha, Knows(a), Holds(a)) = 0$. Useful resources are those where IR_i^+ if $Cap(a, \alpha, Knows(a), Holds(a)) > Cap(a, \alpha, Knows(a), Holds'(a)) if Holds(a) \cap IR_i^+ \neq \emptyset \cap Holds(a) \cap IR_i^+ = \emptyset$.

The coordination problem is to maximize the reward to the team, while minimizing the *costs of coordination*. The overall reward is simply:

$$\sum_{i=0}^{n} Reward(a, \alpha_i, Knows(a), Holds(a), t)$$

The costs of coordination can be very general and in some cases difficult to define. Here we are specifically concerned with only two elements: the volume of communication and the equity with which the efforts of the team were spread over its team members.

This basic coordination problem imposes a number of coordination requirements on a team. These can be viewed as four abstract problems: (1) sharing information; (2) fusing information; (3) allocating tasks and; (4) gaining a joint perspective. Many of the things that a team must do to coordinate can be mapped to one of these basic activities.

3.1 Information Sharing

The agent that senses an event or situation in the environment will not necessarily be the same agent that needs that information. Moreover, unless it has complete knowledge of a teams activities, it may not know which, if any, agent needs the information. Most solutions to this problem use some sort of information broker or other centralized approach, but Xu [19] presented a distributed approach which we follow here.

To model how information sharing occurs in a social network, we adapt the common technique of using Markov chains [2]. We begin by assuming a randomly distributed selection of source agent, a_s and target agent, a_t. a_s does not know the identity of a_t but it knows the properties of an agent that would be interested in the information and it may know something about its

neighbors in the network (see below). The information is passed from agent to agent along links in the network encapsulated in a .

The state, s_i, is defined to be the situation where the mimimum length path to a_t from the current location of the token is of length i. The special state s_0 corresponds to the token, i.e., information, arriving at the target.

If the token moves randomly from agent to agent, then the structure of the network will govern the behavior of the random walk through the network. Specifically, the structure of the network will define the transition function for the Markov chain. We can write $P(s_i, s_j)$ as the probability of transitioning from s_i to s_j, where only $P(s_i, s_{i-1})$, $P(s_i, s_{i+1})$ and $P(s_i, s_i)$ are possible. We determined values for $P(s_i, s_j)$ empirically, for all the networks considered here. Notice that we average $P(s_i, s_j)$ over each node at distance i, though this will vary from node to node.

Once P is known, the expected time to transition from s_i to s_0, t_i can be calculated in a straightforward way:

$$t_i = \sum_{n=1}^{\infty} n(1 - P(s_i, s_i))P(s_i, s_i)^{(n-1)}(P(s_i, s_{i-1})t_{i-1} \tag{1}$$

$$+P(s_i, s_{i+1})t_{i+1}) \tag{2}$$

$$t_i = \frac{1 + P(s_i, s_{i-1})t_{i-1} + P(s_i, s_{i+1})t_{i+1}}{1 - P(s_i, s_i)}$$

Intuitively, in Equation 1, the first term after the sum captures the amount of time the token is expected to stay this distance away from the target and the second captures how long it will take after leaving this distance (and going either closer or further.)

Figures 2 and 3 show the relative rates of $P(s_i, s_{i-1})$, marked "Closer", $P(s_i, s_{i+1})$, marked "Further" and $P(s_i, s_i)$, marked "Same" for Scale Free and Random networks. The x-axis shows the distance of a node to the target node, i.e., the subscript i. Notice that the closer to the target the more likely random movement is to lead further from the target and conversely, the further from the target the more likely random movement will lead the token closer. The figures show that the closer a token is to the target, the easier it is to move away. Moreover, since the figures show different distributions, their information sharing characteristics are likely to be different.

However, in teams, information does not just move randomly from agent to agent. Often the agents will know something (perhaps a lot) about the characteristics of their network neighbors and even the neighbors of their neighbors. Several sociologists have shown how information delivery can be very efficient in human teams with simple models of acquaintances [15, 17] and Xu [19] has effectively illustrated this for multi-agent teams.

To model the fact that the movement is not completely random, but is in fact biased towards the target location, we use a parameter β to make $P(s_i, s_{i-1})$ larger and $P(s_i, s_{i+1})$ smaller. However, this bias should

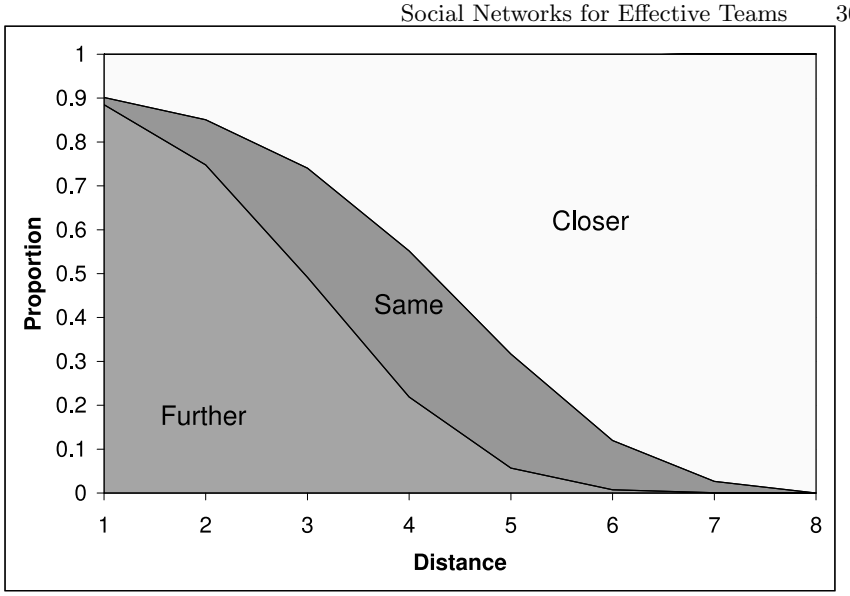

Fig. 2. The relative proportions of links in a Scale Free network that lead closer to, keep the same distance from or move further from some target node, as the distance to the target is varied.

be stronger nearer to the target location, since it is more likely that agents need the target information know what is required to intelligently route the

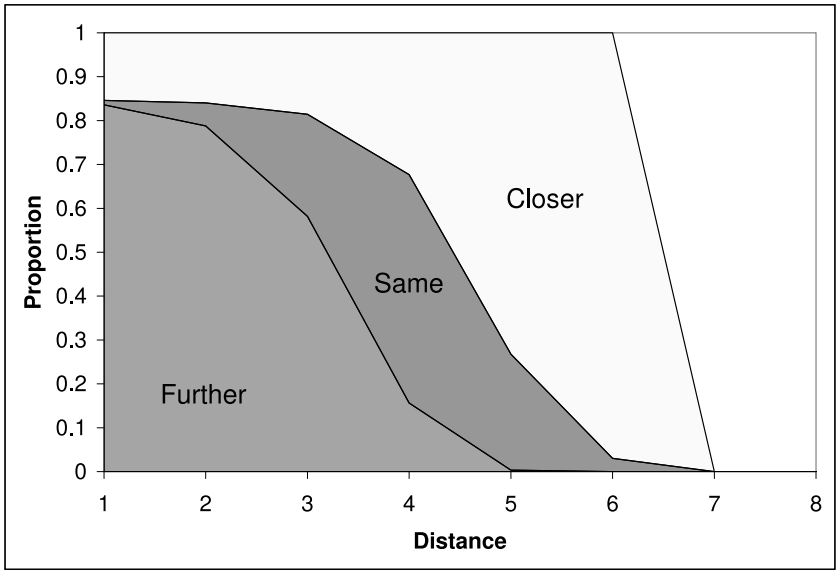

Fig. 3. The relative proportions of links in a Random network that lead closer to, keep the same distance from or move further from some target node, as the distance to the target is varied.

information. We can model this by using $\beta(i) = \frac{1}{e^{\alpha i}}$. Informally, one can think of β as the total learning learning of the team about the team and α as how much more agents "near" an agent know about it than do agents "far" from it. Using α and β the Markov chain state transitions can be rewritten as:

$$\breve{P}(s_i, s_{i-1}) = P(s_i, s_{i-1}) + (1 - e^{\beta(i)})P(s_i, s_i) + (1 - e^{2\beta(i)})P(s_i, s_{i+1})$$

$$\breve{P}(s_i, s_i) = P(s_i, s_i) - (1 - e^{\beta(i)})P(s_i, s_i)$$

$$\breve{P}(s_i, s_{i+1}) = P(s_i, s_{i+1}) - (1 - e^{2\beta(i)})P(s_i, s_{i+1})$$

Figure 4 shows the effect on the scale free distribution from Figure 2. Especially close to the target, i.e., the left of the graph, tokens are much more likely to get closer to the target. Clearly, the result will be more efficient delivery of information when there is a bias as described above.

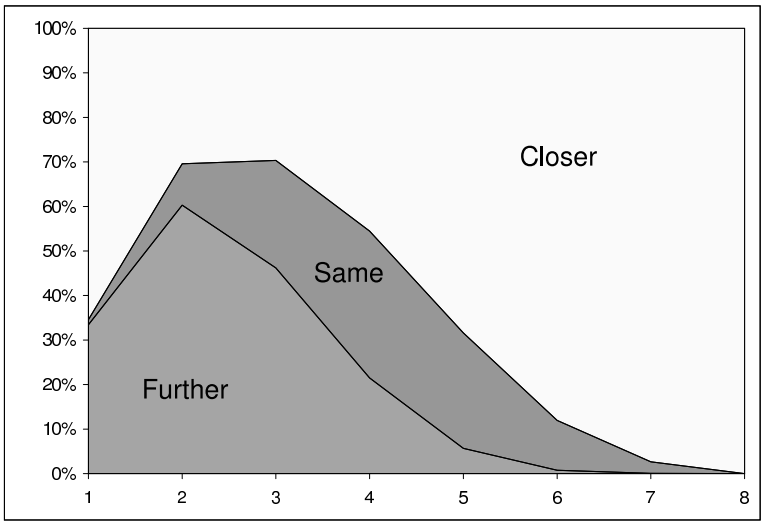

Fig. 4. The relative proportions of links that lead closer to, keep the same distance from or move further from some target node, as the distance to the target is varied. This plus: an α value of 1.5 is used to bias the links towards moving to the target node.

3.2 Information Fusion

In a distributed team, different members of the team may take sensor readings that must be *fused* together to allow action to occur. This basic phenomena can occur for a number of different reasons, including fusing of low confidence sensor readings to get high confidence in the occurrence of some event, for detecting conflicts or synergies between different activities or to know that

multiple preconditions for some course of action have all been met. In a cooperative environment, it is typically only necessary for one member of the team to know the sensor readings to be fused and, in the following, we assume there is no preference ordering across agents for who fuses the information.

The probability that an agent a senses the information itself is, *sense*, which is assumed to be uniform across the team. If each piece of sensed information is randomly passed from neighbor to neighbor then the probability that a gets it in a particular step is $\gamma_a = degree(a)/2|E|$. Thus, if the event occurs at $t = 0$ the probability that a knows about it at t is $\Gamma_a(t) = 1 - ((1 - sense)(1 - \gamma)^t)$.

If n of m sensor readings must be "fused" for action to occur, the probability that a can do the fusion at t is:

$$Fuse(a,t) = \sum_{n'=n}^{n'=m} mCn' \times \Gamma_a(t)^{n'} \times (1 - \Gamma_a(t)^{m-n'}) \tag{3}$$

Informally, this says that the agent can get any combination of sensors readings with equal probability.

$$Fuse(A,t) = 1 - \prod_{a \in A} (1 - Fuse(a,t)) \tag{4}$$

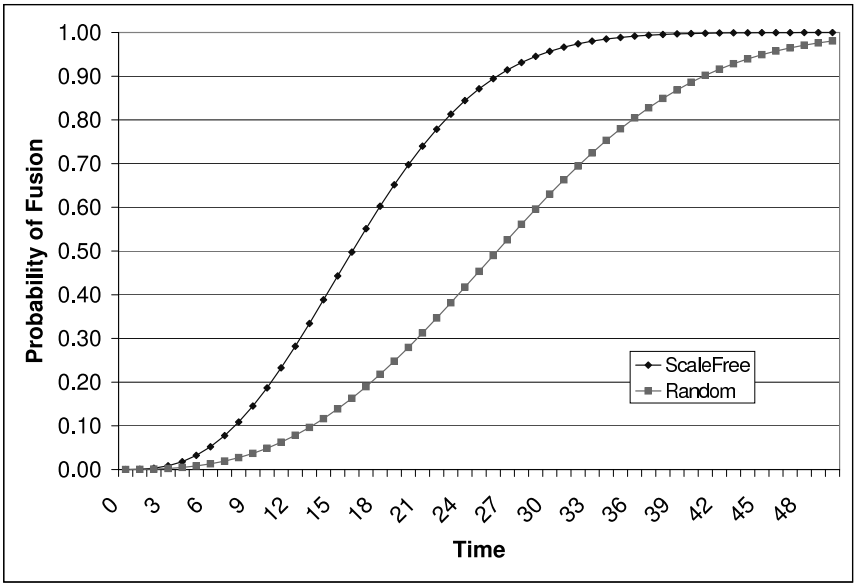

Fig. 5. The probability of fusion over time for two types of network. At each time step five pieces of information move randomly from one agent to another, three must meet for fusion to occur.

Figure 5 shows the probability of fusion for scale free and random networks. The probability of fusion for the scale free network is clearly higher. Examination of Equations 3 and 4, suggests that the advantage of the scale free network is primarily due to the small number of nodes with very many links, since this leads to a high value for γ and consequently Γ. Empirical results bear out this observation (see Section 4).

3.3 Task Allocation

When tasks dynamically arise, capable and available team members need to be found to execute those tasks. There are a large number of different task allocation algorithms that might be used to allocate the tasks. In this chapter, we consider the impact of network structure on LA-DCOP [13] an algorithm that encapsulates a task in a token and moves it around the team until some agent is available and has capability above a *threshold* recorded on the token.

The same formal model developed for the information sharing problem can be applied to understanding the impact of network structure on LA-DCOP. The key is to observe that there are n agents in the team that will accept the task if it reaches them. Each of these n agents can be thought of as a target for the token and we can perform the same analysis as in Equation 2 to determine how far the token will be from an agent that would accept it. Figure 6 shows the relative proportion of different distances to any of n target nodes (on the x-axis) for a random network with 500 agents. The lowest area shows the proportion that are capable of the task themselves, the next area up shows the proportion that is adjacent to an adjacent to an agent capable of the task, the next shows agents 2 links from a capable agent and so on. The figure clearly shows that even if only about 2% of agents are capable, every agent in the network is close to some capable agent.

Task allocation is a specialized coordination task for which it is intuitively important precisely which agent is next to which other. Specifically, it seems intuitive that if agents with different capabilities are close to each other in a network, task allocation can function more effectively, since if the agent is not capable of a task, it is more likely to be able to find someone that is. To evaluate this hypothesis, we configured random networks in two ways: one where random links are created with a strong preference to agents with similar capabilities and one where random links are created with a strong preference for linking to agents with different capabilities. Figure 7 shows an experiment with 500 agents and 2500 tasks. Tasks take some duration to execute and an agent may perform only one task at a time, hence the availability of agents to perform tasks will change over time. As expected the network where agents were connected to others with different capabilities did lead to better results, but the difference was not unexpectedly small.

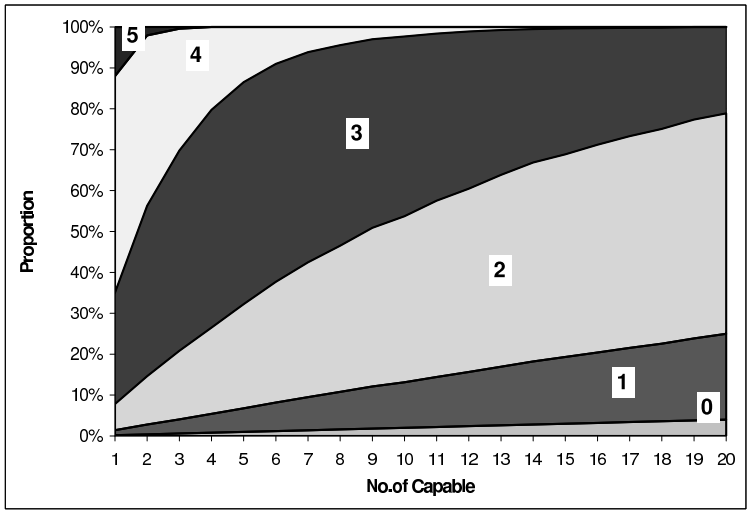

Fig. 6. Percentage of nodes that are of distance 0-5 from an agent capable of performing some task as the number of capable agents is increased. There are 500 agents in all, arranged in a random network.

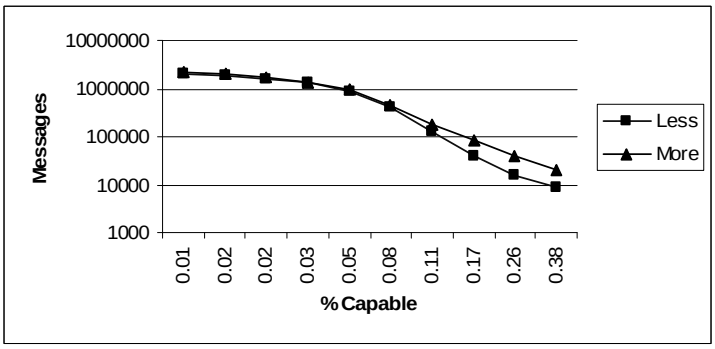

Fig. 7. The number of messages required to allocate 2500 tasks in two random 500 agent networks. One random network arranges agents so those with similar capabilities are near one another (More) and the other arranges agents so that agents with similar capabilities are far from one another (Less).

3.4 Perspective Aggregation

Some coordination algorithms or activities require that each member of the team builds an accurate view of the team's state. For example, the way that an individual agent uses shared resources such as communication bandwidth or fuel should depend on the team's overall need for such resources.

Technically, we can think of every member of the team having a local value for some variable, e.g., their local need for some resource, and needing to know the average value of that variable across the whole team, e.g., the average need

for some resource. Formally, each agent has some variable v. The perspective aggregation problem is for each agent to know $\bar{v} = \frac{\sum_{a \in A} v}{|A|}$.

One way of building up a perspective across the team is to have a small number of *propagators* move from agent to agent, taking the current perspective from one agent and adding it to the current perspective at the next agent. If there are multiple propagators simultaneously moving around the team, perspectives build up very quickly. Notice that it is typically infeasible for a propagator to record precisely which agents it has collected values for, since it would need to record all the agent IDs as it moved from agent to agent. In a large team this imposes an unreasonable communication load. However, because the propagator does not know precisely which agents it has visited, some will be visited repeatedly and their values counted repeatedly, distorting the average results.

A simple model of how quickly these perspectives build up can be straightforwardly created by considering how many other values each agent knows about. Before any propagators move, each agent knows only their value. Thus, the average number of values known by each agent $Avg(v, 0)$ is 1. When a propagator moves, one of the agents gets to know 1 new value, hence $Avg(v, 1) = 1 + \frac{1}{|A|}$. In general, the average number of values known by an agent after move t of the propagators is $Avg(v, t) = Avg(v, t-1) + \frac{Avg(v,t-l)}{|A|}$. Because propagators collect information as they move, $Avg(v, t)$ rapidly grows with t. Figure 8, shows $Avg(V, t)$ for a team with 500 members. The x-axis shows the number of propagator moves divided by the number of agents and the y-axis shows $Avg(V, t)$ on a logarithmic scale. The figure suggests that perspective aggregation is not a communication intensive task for a team, even one with relatively few edges.

The average value does not capture two key aspects of the perspective aggregation problem. First, nodes with higher degree will be visited more often by a randomly moving propagator than nodes with lower degree. This effect can be modelled by changing the denominator in $\frac{Avg(v,t-l)}{|A|}$ to be $\rho|A|$, i.e., agents with higher than average degree will have $\rho < 1.0$ and those with lower than average degree will have $\rho > 1.0$. Thus, networks with many nodes with low degree are likely to perform poorly on this task. Second, clearly many of the values an agent gets to know will be repeated. The distortion caused to the agent's perspective by the repeats will be proportional to the relative rates at which repeats occur, i.e., if some values are repeated many times and others are not, the agent's perspective will get very distorted. An agent will likely get to know about another agent's value more often if that agent is close to it in the network than if it is far from it. Thus, networks with higher width are likely to perform poorly on this task.

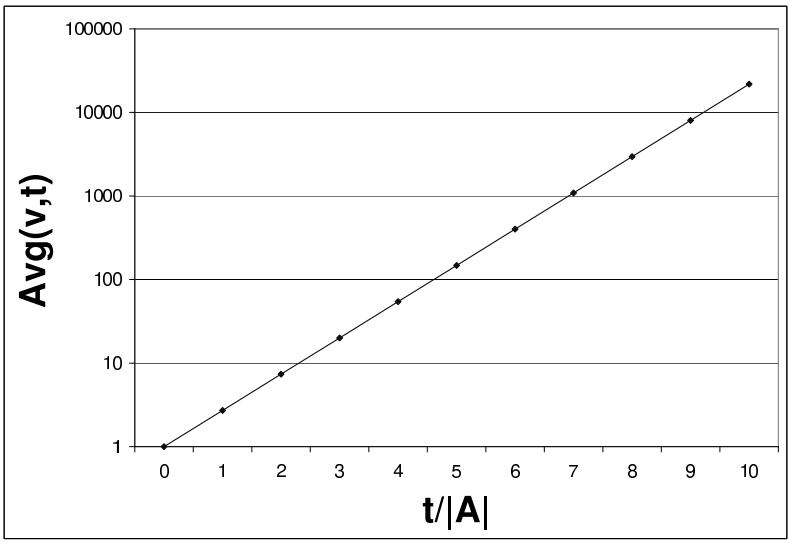

Fig. 8. The average number of samples each agent has (y-axis) after a propagator has moved a fixed number of steps (x-axis). The y-axis has a logarithmic scale.

4 Experiments

To empirically evaluate all the different networks, we developed simple Java programs to simulate each of the algorithms and networks. For each subproblem we created 100 networks of 500 agents of each type and measured two things: the performance of the algorithm on the network and the standard deviation of the contribution of each agent to the observed performance. For example, for the information fusion problem, we randomly allowed five agents to "sense" five pieces of information and then propagated that information around the network until some agent knew of three of the pieces of information. For this information fusion, we measured the time taken to fuse and the number of times each agent performed the fusion. A summary of the results is shown in Figures 9 (algorithm performance) and 10 (distribution of effort). The results have been normalized for clarity. In Figure 10 the distribution of effort is computed as the magnitude of the standard deviation of effort, based on the idea that higher standard deviation means more variability in effort. The experiments indicate that no network outperforms all others on all tasks and there is often an inverse relationship between performance and distribution of effort.

Finally, to perform a more complete test on the effects of networks on coordination, we used an abstract coordination simulator called CoordSim. This simulator is capable of simulating the major aspects of coordination including sensor fusion, plan management, information sharing, task assignment and resource allocation. CoordSim abstracts away the environment, instead just

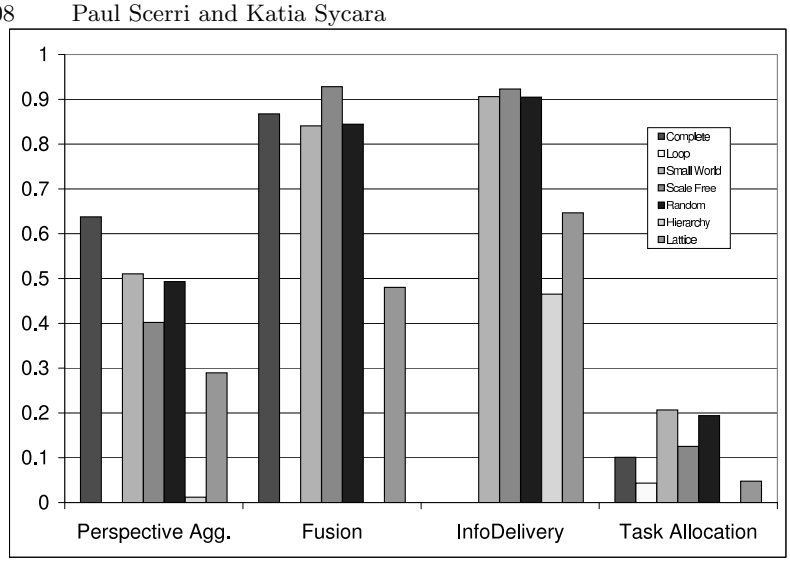

Fig. 9. Relative performance of each network on each abstract problem. Values are normalized and inverted where necessary so that bigger is better.

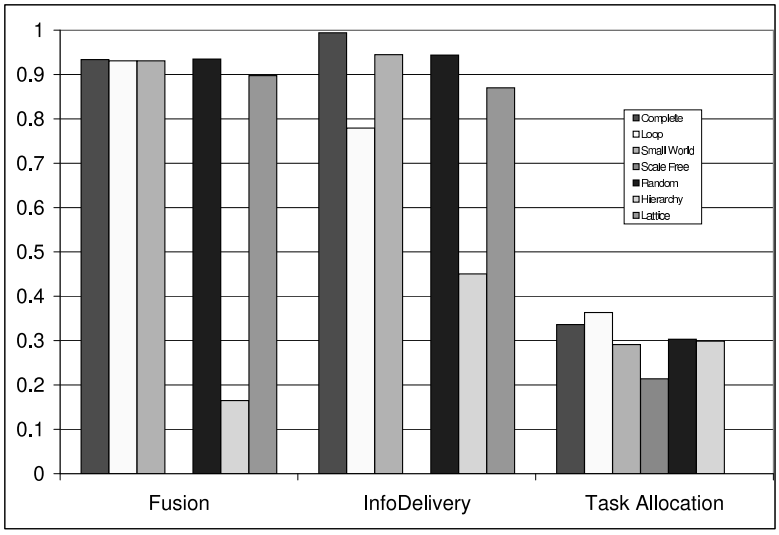

Fig. 10. Relative equitableness of algorithms on different problems. Values are normalized and inverted where necessary so that bigger is better.

simulating its effects on the team. Uncertain sensor readings are received randomly by some agent or agents in the team at a parameterizable rate. Agents cannot "know" anything they do not sense or is not communicated to them from a teammate. Time is simulated and all agents are allowed to "think"

and "act" at each step, although the effects of their "actions" are abstractly simulated. Communication is implemented via object passing, making it very fast. Reward is simulated as being received by the team when the agent is allocated the task, the agent's simulated location is at the task location and it has exclusive access to required resources. Reward is received while the agent is simulating the task, which takes one time step.

One key element simulated by CoordSim that is not addressed in the subproblems above is that of deconflicting plan instantiations. In CoordSim, when any agent comes to know that the preconditions for some joint activity are known, it initiates that joint activity and informs *its neighbors in the network*. As tasks are allocated for the joint activity, agents assigned roles in the joint activity also inform their neighbors *in the network* of the joint activity. If any agent gets to know of two activities achieving the same goal, they initiate a process to stop one of the activities. If two activities aimed at achieving the same goal are executed, the team only gets reward for one.

Figures 11-14 show the results from CoordSim for four different coordination instances. In each case, there are 200 agents and 50 plans. Unless otherwise noted: (i) plans are initiated when two preconditions are true and some agent gets to know both preconditions and starts the plan; (ii) there are between one and four roles per plan; (iii) there are one to five pieces of information available that can improve execution of a task, if the agent performing the task knows of that information; (iv) there are 20 different types of task and each agent can perform between one and five of them and; (v) there are 150 pieces of information that can be known, each must be fused from at least three of five sensor readings randomly distributed to the team. Figure 11 shows a baseline configuration. Figure 12 shows a case where fusion is more important, specifically four preconditions must be known to instantiate a plan and four of five sensor readings are required to fuse a piece of information. Figure 13 shows a case where task allocation is more important, with between five and ten roles per plan, fifty different types of task and each agent only capable of one thing. Finally, Figure 14 shows a case where information sharing is important, because five to ten pieces of information can improve the reward of the agent performing the task.

In all but one case, Figure 12, the complete network performed best. A primary reason for this was that is was never executing conflicting plans, because whenever one agent initiated execution of a plan, it would inform all others and any duplicate instantiations would be immediately removed. The low number of role allocation messages is testemant to this efficiency. This ability to remove conflicting plans, overwhelmed any other advantages the other networks had. However, in the configuration in Figure 12 instantiating plans relies on very effective information fusion, both to get the pieces of information from the sensor readings and then to have one agent know four readings and instantiate the plan. Since fusion was so difficult, relatively few duplicate plans were created and hence deconfliction became less important. Instead, the networks more suited to sensor fusion performed relatively better.

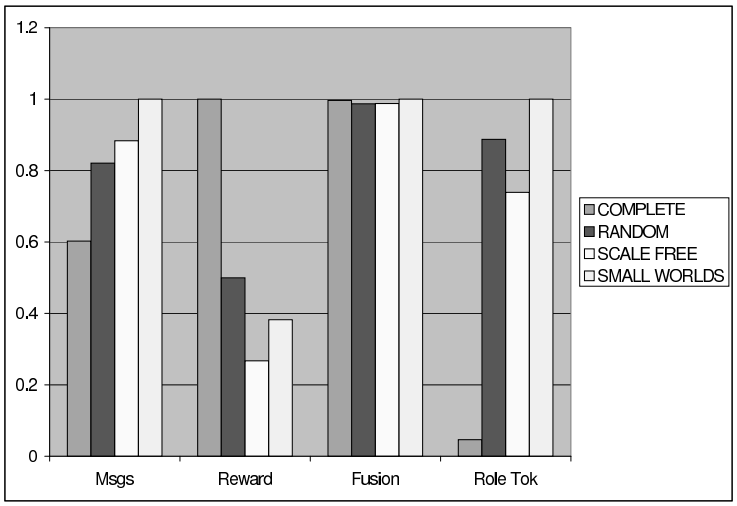

Fig. 11. Coordination experiment baseline case

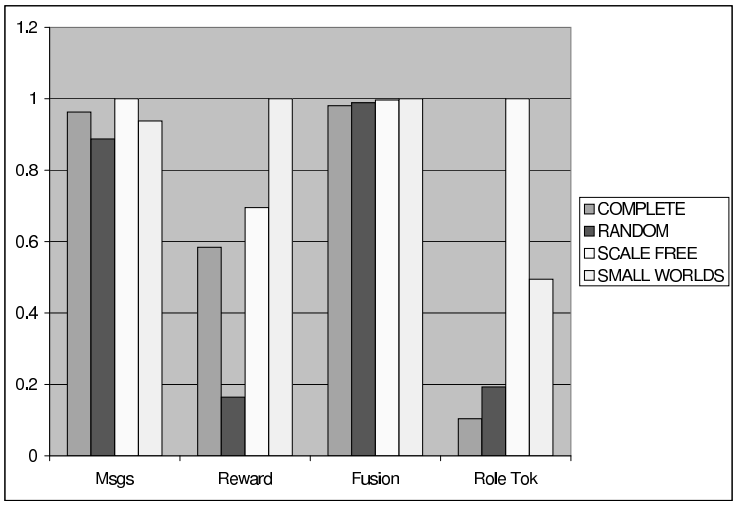

Fig. 12. Coordination experiment with more emphasis on fusion sub-problem

5 Related Work

Social networks have been an active area of interest since Milgram observed the small worlds effect nearly 40 years ago [15]. Recent interest in such networks was inspired by models by Barabasi [1] and Watts [16] who observed that similar networks occurred in nature. Over time an amazing array of fields of research have contributed to our understanding of networks [11], from biologists [18], to mathematicians [14] to physicists [10]. Some economists explained such networks by balancing the cost of maintaining an acquaintance against the value of that acquaintance [6]. Sociologists, including Carley, showed the

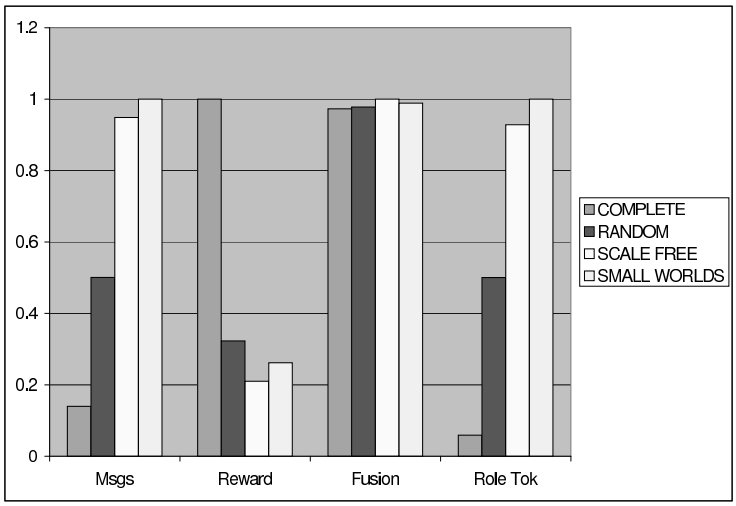

Fig. 13. Coordination experiment with more emphasis on task allocation subproblem

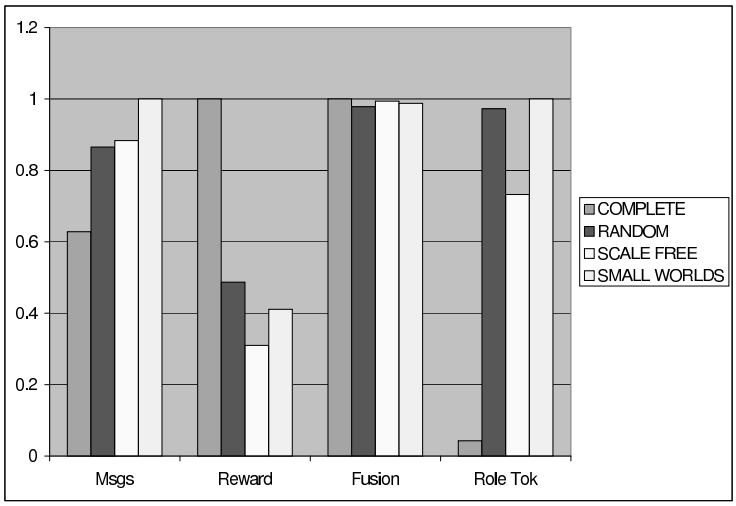

Fig. 14. Coordination experiment with more emphasis on information sharing subproblem

rich network structure underlying some organizations and how that network facilitates effective organizational behavior [9].

Computer scientists and agents researchers have also been interested in these network structures. For example, Kleinberg [8] shows some of the information requirements on an agent to replicate Milgram's results in multiagent systems. Gaston recently published two key papers showing how network structure has an impact on multiagent systems and proposing an algorithm for designing networks for multiagent systems [5, 4].

6 Conclusions

This chapter presented a quantitative investigation of the impact of social networks on multi-agent coordination. Our results support previous work which indicated the importance [5, 4] and utility [20] of social networks. For abstracted coordination tasks, networks with relatively low degree were shown to often significantly outperform completely connected networks. Some of the advantage low degree networks were shown to have was due to a small number of nodes performing a relative large proportion of the work, essentially partially centralizing the coordination. However, we showed that the advantage of social networks disappeared when all coordination tasks were taken into account.

While this chapter advances our understanding of the impact of social networks on coordination, key work is required to utilize this new understanding. Most urgently, we showed that none of the networks we used were best for all coordination problems. A key question is whether there is a particular network that is best in *all* situations. It is possible, or even likely, that changing structure over time is better than any fixed structure. Finally, although initial experiments that looked more carefully at which agent should be adjacent to which other showed little effect, it is likely there are some more significant effects for other adjacencies. Our future work will investigate these questions and apply these social networks to practical multiagent systems.

Another possibility is to have different logical networks for each task in the same team, with networks chosen to be specifically good for the sub-problems they are used for.

References

1. A.-L. Barabasi and E. Bonabeau, Scale free networks, *Scientific American*, pp. 60–69, May 2003.
2. V. Buskens and K. Yamaguchi, A new model for information diffusion in heterogeneous social networks, *Sociological methodology*, Vol. 29(1), 1999.
3. L. Freeman, Centrality in social networks: Conceptual clarification, *Social Networks*, Vol. 1(3), pp. 215–239, 1979.
4. M. Gaston and M. desJardinss, Agent-organized networks for dynamic team formation, In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Utrecht, Netherlands, 2005.
5. M. Gaston and M. desJardinss, Agent-organized networks for multi-agent production and exchange, In *Proceedings of the Twentieth National Conference on Artificial Intelligenc*, Pittsburgh, PA, 2005.
6. J. Geunes and P. Pardalos, Network optimization in supply chain management and financial engineering: An annotated bibliography, *Networks*, Vol. 42(2), 2003.
7. P. Harrison and W. Knottenbelt, Networks, dynamics, and the small-world phenomenon, *The American Journal of Sociology*, Vol. 105, No. 2:493–527, 1999.

8. J. Kleinberg, The small world phenomenon: An algorithmic perspective, In *Proceedings of Symposium on Theory of Computing*, 200.
9. Z. Lin and K. Carley, Dycorp: A computational framework for examining organizational performance under dynamic conditions, *Journal of Mathematical Sociology*, Vol. 20, 1995.
10. R.M. May and A.L. Lloyd, Infection dynamics on scale-free networks, *Physical Review E.*, 2001.
11. M. Newman, The structure and function of complex networks, *SIAM Review*, Vol. 45(2), 2003.
12. P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara, Scaling teamwork to very large teams, in *Proceedings of AAMAS'04*, 2004.
13. P. Scerri, A. Farinelli, S. Okamoto and M. Tambe, Allocating tasks in extreme teams, In *AAMAS'05*, 2005.
14. C. Topper and K. Carley, A structural perspective on the emergence of network organizations, *Journal of Mathematical Sociology*, Vol. 24(1), 1999.
15. J. Travers and S. Milgram, An experimental study of the small world problem, *Sociometry*, Vol. 32, pp. 425–443, 1969.
16. D. Watts and S. Strogatz, Collective dynamics of small world networks, *Nature*, Vol. 393, pp. 440–442, 1998.
17. D.J. Watts, P.S. Dodds and M. E. J. Newman, Identity and search in social networks, *Science*, Vol. 296(5571), pp. 1302–1305, 2002.
18. Y.I. Wolf, G. Karevand, E.V. Koonin, Scale-free networks in biology: new insights into the fundamentals of evolution? *BioEssays*, 2002.
19. Y. Xu, M. Lewis, K. Sycara and P. Scerri, Information sharing in very large teams, in *In AAMAS'04 Workshop on Challenges in Coordination of Large Scale MultiAgent Systems*, 2004.
20. Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis and K. Sycara, An integrated token-based algorithm for scalable coordination, in *AAMAS'05*, 2005.