Techniques and Directions for Building Very Large Agent Teams

(Invited Paper Number 1049)

Paul Scerri, Joseph A. Giampapa, Katia P. Sycara
The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, Pennsylvania
{pscerri,garof,katia}+@cs.cmu.edu
http://www.cs.cmu.edu/~softagents

Abstract— **We have developed probabilistic algorithms that leverage the associates network for distributed plan instantiation, role allocation, information sharing and adjustable autonomy with a team. By developing such new algorithms, we have been able to build teams of hundreds of cooperating agents, and test specific behaviors among tens of thousands of agents. In this paper, we describe the algorithms that we have developed, the tests that we subjected them to, and sketch some of the key challenges that remain to be addressed.**

I. Introduction

Teamwork allows autonomous, heterogeneous agents to cooperate to achieve complex goals in complex environments. Very large teams are needed for domains such as the military, space operations [1], disaster recovery and commerce, where there are many interrelated, complex activities to be performed by many agents. Typically, the problems for which such teams are most applicable do not require optimal behavior, but require reliably "good" behavior despite an uncertain, complex and potentially hostile environment.

The key technical difference between teamwork and other forms of coordinated behavior is the use of an explicit model of teamwork and an explicit representation of the current team status maintained by each team member. Each team member uses their model of teamwork and the team status to reason about the actions, both in the environment and communication acts, that will lead to the best expected outcome for the team. The reasoning an agent performs with its models is derived from early work on the logic of teamwork [2], [3], although practical implementations [4], [5], [6] relax some of the requirements of the logical formulation. The key to building large teams is to find ways for the maintenance of agents' models of team state, and reasoning with those models, to be scalable.

The abstraction used in our teamwork models is a *team oriented plan* (TOP). A TOP describes the individual activities, *roles*, that must be performed to achieve team objectives and any constraints between those individual activities. The team starts with a library of multiple plan templates that are dynamically selected and instantiated when preconditions on the plan are matched by environmental conditions and/or individual team member intentions. For example, a plan template for rescuing an injured civilian in a disaster response domain may have two roles: one for bringing the civilian to safety; and another for administering first aid. A constraint would require that the latter role be performed after the former. Typically, a large team will be simultaneously executing many plans at once. Using the TOP abstraction, the task of coordination via teamwork is to determine what actions a team member should take given the current plans, and what each team member should communicate about the plans. In practice, scalable teamwork requires that each team member does not need to reason and communicate about each plan, but only some subset of plans that concern them or are within their capabilities.

We achieve the scalability of team coordination among the members of a large team (e.g. in the range from hundreds to tens of thousands of members) by organizing all members into an *associates network* [7]. An associates network is a static, logical organization of an agent team that gives the team the properties of a *small-world* network [8]. The organization of the team into an associates network is performed once, at the initialization of the entire large-scale team, and remains immutable for the duration of the team's existence. In imprecise intuitive terms, it is like organizing the individual team members into a static hierarchy of departments and subdepartments. Unlike a static hierarchy, an associates network does not have any root node or imply any relationships among neighbors — there are no parent/child or sibling relationships — and topological proximity and organization of the nodes does not imply relationships among them. This lack of explicit and implicit relationships among the nodes provides generality of application context to the associates network by keeping it neutral of any task domain, and the lack of hierarchical vertices provides robustness by limiting the existence of critical nodes that, should they fail, would break the connectivity of the network. Models of networks with small-world properties exhibit enhanced signal-propagation speed and synchronizability. The properties that enable such characteristics are a high degree of clustering within the network and small characteristic path lengths[1]. The team uses the associates network to limit the amount of communication required for cohesive activity to just sharing information probabilistically and finding individual

[1]The number of nodes through which a message must pass to get from any one agent to any other.

members for a dynamic subteam. For example, team members can leverage the network to search for agents with special skills who can participate in a plan. When an agent wishes to find another agent with particular characteristics, it pushes a search request to its neighbor in the associates network that is *most likely* to be able to respond to the search characteristics, or if not, is most likely to have another associate neighbor that does. This allows team members to act despite having very limited models of the status of the rest of the team, hence reducing the need to saturate team communications with the propagation of state information. As a consequence, however, the logical guarantees provided by previous teamwork algorithms are replaced by algorithms that have high probability of working correctly.

Our approach to teamwork has been implemented in domain independent software proxies, semi-autonomous coordination modules for each team member [9], called *Machinetta* [10]. Each proxy encapsulates the team work algorithms and works closely with an individual team member and with other proxies to effect teamwork. We have used Machinetta proxies in several domains, such as robotic rescue and the coordination of large groups of unmanned aerial vehicles.

II. The Description of a Team

At a high level, team behavior can be understood as follows. All members of a team are initialized with access to the same plan library. An individual team member, i.e. an agent, detects events in the environment and respond by selecting one plan that best describes the best response to the event that can also achieve the team's top level goal, or at least satisfy a portion of the overall team's top goal. This agent, which we call a *plan emitting agent*, then communicates a description of the subplan that it would like to execute to its neighbors in the associates network, according to the likelihood of those neighbors, or in turn, their neighbors, committing the subplan. The description is propagated through the network until agents are found that can commit to the individual roles of the subplan. Once all of the roles of a subplan have been committed to by agents, one can now acknowledge the existence of a dynamically-formed subteam, which remains focused on achieving the subteam's goal until the goal is achieved or it has been determined that the team cannot achieve it [2]. Any possible conflicts or synergies with other subteams are highly likely to be detected by virtue of the degree of agent subteam overlap due to agents being members of more than one team. Finally, agents share locally sensed information on the associates network to allow the whole team to leverage the local sensing abilities of each team member.

A. Associates Network Organization

A team A consists of a large number of agents (e.g. in the range from hundreds to tens of thousands of members), $A = \{a_1, a_2, \ldots, a_n\}$ that have been assembled to pursue an overall team goal. The *associates network* arranges the whole team into a small worlds network [8] defined by $N(t) = \bigcup_{a \in A} n(a)$, where $n(a)$ are the *neighbors* of agent a

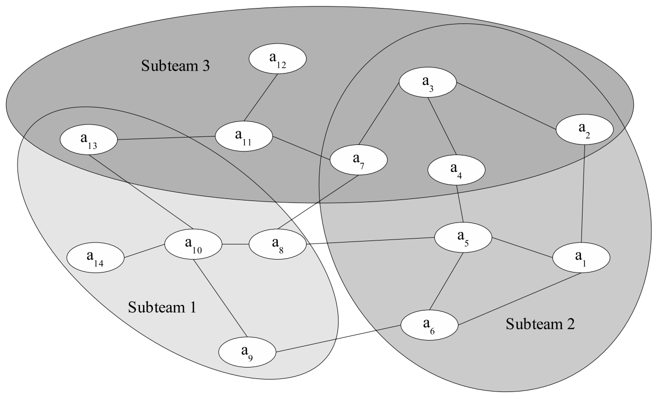


Fig. 1. Relationship between subteams and the associates network

in the network. The minimum number of *agents* a message must pass through to get from one agent to another via the associates network is the $distance$ between those agents. For example, as shown in Figure 1, agents a_1 and a_3 are not neighbors but share a neighbor, hence $distance(a_1, a_3) = 1$. We require that the network be a small worlds network, which imposes two constraints. First, $\forall a \in A, |n(a)| < K$, where K is a small integer, typically less than 10. Second, $\forall a_i, a_j \in A, distance(a_i, a_j) < D$ where D is a small integer, typically less than 10.

B. Plans and Subteams

The team A has a top level goal, G to which the team *commits*, with the semantics of STEAM [9]. Achieving G requires achieving sub-goals, g_i, that are not known in advance but are a function of the environment. For example, sub-goals of a high level goal to respond to a disaster are to extinguish a fire and provide medical attention to injured civilians. Each sub-goal is addressed with a plan, $plan_i =< g_i, recipe_i, roles_i, d_i >$. The overall team thus has plans $Plans(t) = \{plan_1, \ldots, plan_n\}$, though individual team members will not necessarily know all plans. To maximize the responsiveness of the team to changes in the environment, we allow any team member to commit the team to executing a plan, when it detects that g_i is relevant. $recipe_i$ is a description of the way the sub-goal will be achieved [6] and $roles_i = \{r_1, r_2, r_3, \ldots, r_r\}$ are the individual activities that must be performed in order to execute that $recipe_i$. d_i is the domain specific information pertinent to the plan. For convenience, we write $perform(r, a)$ to signify that agent a is working on role r. We capture the identities of those agents involved in role allocation with $allocate(plan_i)$.

C. Allocating Roles to Team Members

Role allocation can be approached as the problem of assigning roles to agents so as to maximize overall team utility [11], [12]. Large-scale teams emphasize key additional requirements in role allocation:

1) rapid role allocation as domain dynamics may cause tasks to disappear;
2) agents may perform one or more roles, but within resource limits;

3) many agents can fulfill the same role; and
4) inter-role constraints may be present.

This role allocation challenge in such teams will be referred to as *E-GAP*, as it subsumes the generalized assignment problem (GAP), which is NP-complete [13].

We have adapted ideas from Distributed Constraint Optimization (DCOP) [14], [15] for role allocation, as DCOP offers the key advantages of distributedness and a rich representational language which can consider the costs and utilities of tasks. Despite these advantages, DCOP approaches to role allocation suffer from three weaknesses. First, complete DCOP algorithms [14] have exponential run-time complexity and, thus, fail to meet the response requirements of extreme teams. One reason for this is that the purely local view of the team that each agent has forces the search to explore many potential solutions that are clearly sub-optimal. Teams of agents will often have reasonably accurate estimates of both the situation and the state of the team, however, which can be used to accurately estimate likely solution characteristics. For example, when a team of fire fighters responds to a disaster, it is reasonable to assume that they know the number of fires and number of available fire trucks to within an order of magnitude of the correct figure, even though they may have very little specific knowledge of individual fires or trucks. While relying on such estimates prevents guarantees of optimality, they can dramatically reduce the search space. Second, similar agent functionality within very large teams results in dense constraint graphs increasing communication within a DCOP algorithm. Third, DCOP algorithms do not address the additional complications of constraints *between* roles.

For large scale teams, we have developed a novel DCOP algorithm called LA-DCOP (Low communication Approximate DCOP) [16]. LA-DCOP uses a representation where agents are variables that can take on values from a common pool, i.e., the pool of roles to be assigned. The mechanism for allocating values to variables encapsulates three novel ideas. First, LA-DCOP improves efficiency by not focusing on an exact optimal reward; instead by exploiting the likely characteristics of optimal allocations, given the available probabilistic information, it focuses on maximizing the team's expected total reward. In particular, the agents *compute* a minimum threshold on the expected capability of the agent that would maximize expected team performance. If the agent's capability to perform a role is less than the threshold capability, it does not consider taking on the role, but channels the role towards more capable agents. Second, to reduce the significant communication overheads due to constraint graph denseness, *tokens* are used to regulate access to values. Only the agent currently holding the token for a particular value can consider assigning that value to its variable. The use of tokens removes the possibility of several agents taking on the same role, thus dramatically reducing the need to communicate about and repair conflicts. Third, to deal with groups of tightly constrained roles, we introduce the idea of allowing values to be represented by *potential tokens*. When groups of roles must be simultaneously performed, instead of committing to a role by assigning the value represented by a token, a team member accepts a potential token. This indicates that it will accept the role only when all simultaneous roles can be assigned. While team members are being found to fill the other simultaneous roles, a team member with a potential token can perform other roles. Only when team members have been found for all roles will the holders of the potential tokens actually take on the roles. This technique frees team members up for other roles when not all roles in a constrained set can be filled. More detail can be found in [17].

D. Mutual Beliefs and Subteams

Agents working on the plan *and* their neighbors in the associates network, make up the *subteam* for the plan (we write the subteam for $plan_i$ as $subteam_i$). Since allocation of team members to roles may change due to failures or changing circumstances, the members of a subteam also change. All subteam members must be kept informed of the state of the plan, e.g., they must be informed if the plan becomes irrelevant. This maximizes cohesion and minimizes wasted effort. Typically $|subteam_i| < 20$, although it may vary with plan complexity. Typically, $subteam_i \cap subteam_j \neq \emptyset$. These subteams are the basis for our coordination framework and leads to scalability in teams.

We distinguish between two sets of agents within the subteam: those that are assigned to roles, $roles_i$, in the plan and those that are not. The subteam members actually assigned to roles in a plan $plan_i$, called the *role executing agents*, $REA(p_i) = \{a | a \in A, \exists r \in roles_i, perform(r, a)\}$. The non-role executing agents are called *weakly goal related agents* $WGRA(p_i) = \{a | a \in A, a \in allocate(p_i) \wedge associate(allocate(p_i)) \wedge associate(REA)\}$.

A key to scaling teamwork is the efficient sharing of information pertaining to the activities of the team members. Using the definitions of subteams, we can provide relaxed requirements on mutual beliefs, making it feasible to build much larger teams. Specifically, agents in REA_i must maintain mutual beliefs over all pieces of information in $plan_i$, while agents only in $WGRA_i$ must maintain mutual beliefs over only g_i and $recipe_i$. Maintaining these mutual beliefs within the subteam requires relatively little communication, and scales very well as more subteams are added.

E. Detecting Conflicts and Synergies

Detecting conflicts or synergies between two known plans is a challenging task [18], [19], but in the context of a large team there is the critical additional problem of ensuring that some team member knows of both recipes. Here we focus on this additional challenge. When we allow an individual agent to commit the team to a goal, there is the possibility that the team may be executing conflicting plans or plans which might be combined into a single, more efficient plan. Once a conflict is detected plan termination or merging is possible due to the fact that the agents form a subteam and thus maintain mutual belief. Since it is infeasible to require that every team member to know all plans, we use a distributed approach, leveraging the associates network. This approach leads to a high probability of detecting conflicts and synergies, with very low overheads.

If two plans $plan_i$ and $plan_j$ have some conflict or potential synergy, then we require $subteam_i \cap subteam_j \neq \emptyset$ to detect it. A simple probability calculation reveals that the probability of overlap between subteams is:

$$Pr(overlap) = 1 - \frac{{}_{(n-k)}C_m}{{}_nC_m}$$

where where n = number of agents, k = size of subteam A, m = size of subteam B and ${}_aC_b$ denotes a combination.

For example, if $|subteam_i| = |subteam_j| = 20$ and $|A| = 200$, then $P(overlap) = 0.88$, despite each subteam involving only 10% of the overall team. Since, the constituents of a subteam change over time, this is actually a lower bound on the probability a conflict is detected because over time more agents are actually involved. In Section III we experimentally show that this technique leads to a high probability of detecting conflicts.

F. Sharing Information in Large Teams

In Section II-D, we showed how requiring mutual beliefs only within subteams acting on specific goals can dramatically reduce the communication required in a big team. However, individual team members will sometimes get domain level information, via local sensors, that is relevant to members of another subteam. Due to the fact that team members do not know what each other subteam is doing, they will sometimes have locally sensed information that they do not know who requires. In this section, we present an approach to sharing such information, leveraging the small worlds properties of the associates network. The basic idea is to forward information to whichever acquaintance in the associates network is most likely to either need the information or have a neighbor who does.

Agents send information around the team in *messages*. A message consists of four parts, $M = < sender, i, E, count >$. The first two elements, $sender$ and i, denote the agent that sent the message and the piece of information being communicated. With this algorithm, we are only interested in delivering domain level information (as opposed to coordination information). Hence, $I = \{i_1, i_2, \ldots, i_n\}$, defines all the information that could be sent, here i is defined according to d_i in Section II-B. The last two elements of a message, E and $count$, are used for improving the team's information flow and determine when to stop forwarding a message, respectively.

For the purposes of information sharing, the internal state of the team member a is represented by $S_a = < H_a, K_a, P_a >$. H_a is the (possibly truncated) history of messages received by the a. $K_a \subseteq I$ is the local knowledge of the agent. If $i \in K_a$, we say $knows(a,i) = 1$, otherwise, $knows(a,i) = 0$. Typically, individual team members will know only a small fraction of all the team knows, i.e., $|K_a| << |I|$. Our algorithms are primarily aimed at routing information in $I - K_a$, since it is this information that needs to be shared. Thus, the agents are reasoning in advance about how they would route information. For example, a fire fighter might build a model of who might be interested in particular street blockages.

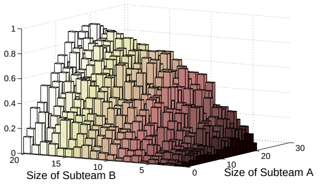
Since the reason for sharing information between teammates is to improve performance of a team, quantifying the importance of a piece of information i to an agent a at time t is needed. Specifically, we use the function $U : I \times A \rightarrow \mathcal{R}$. The importance of the information i is calculated by determining the expected increase in utility for the agent with the information versus without it. That is, $U(a,i) = EU(a, K_a \cup i) - EU(a, K_a)$, where $EU(a, K_a)$ is the expected utility of the agent a with knowledge K_a. When $U(a,i) > 0$, knowledge of i is useful to a, and the larger the value of $U(a,i)$ the more useful i is to a. Formally, the reward for the team is $reward(i) = \frac{\sum_{a \in A} U(a,i) \times knows(a,i)}{\sum_{a \in A} knows(a,i)}$. Since this calculation is based on knowing the use of a piece of information to each agent, agents cannot compute this locally. Thus, it is simply a metric to be used to measure algorithm performance.

The heart of our algorithm is a model of the relative probabilities that sending a piece of information to a neighbor will lead to an increase in the reward as defined by our objective function. This is P_a in the agent state. P_a is a matrix where $P_a[i,b] \rightarrow [0,1], b \in N(a), i \in I$ represents the probability that neighbor b is the best neighbor to send information i to. For example, if $P_a[i,b] = 0.7$, then a will usually forward i to agent b as b is very likely the best of its neighbors to receive it. To obey the rules of probability, we require $\forall i \in I, \sum_{b \in N(a)} P^t_a[i,b] = 1$. Although space does not allow a full explanation of these models, the reader is referred to [7], [20], [21] for more detailed descriptions of the different models that we have examined.

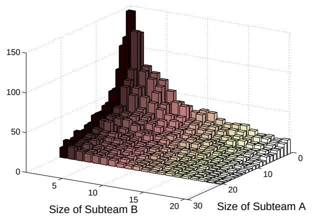
III. Experimental Results

Our model of teamwork described above has been implemented as 200 Machinetta proxies, with the same proxies applied to both the rescue domain and a simulated UAV search and destroy domain [7]. Although experiments with large teams demonstrate the feasability of the approach, often they are not effective at isolating specific factors affecting performance. Hence, to better understand the key algorithms, we first ran a series of experiments in *MatLab*, and then extended our previous Machinetta experiments to *TeamSim*, a highly configurable Machinetta simulator with extensive logging capabilities.

First, we tested our information sharing algorithms on very large teams using two different types of networks: a small worlds network and a network with random links. We arranged $32,000$ agents into a network and randomly picked one agent as the source of an element of information, i, and another as a sink. The sink agent sent out 30 messages with information of strong interest to i with $MAX\_STEPS = 300$. Then, the source agent sent out i and we measured the time that was required for the message to reach the sink agent. Routing via the small world associates network, on average, required less than half of the number of hops than what was required when

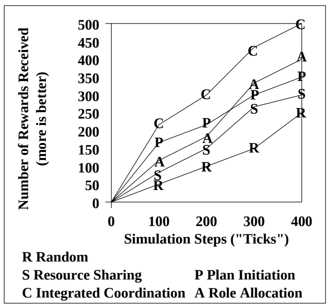


(a) Pr(overlap) vs team size

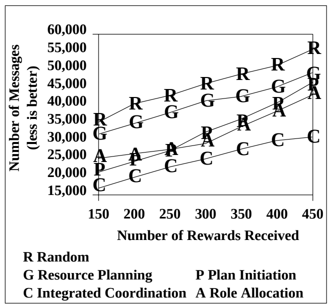


(b) Avg. replacement times to overlap

Fig. 2. Fig. 2(a): The probability of having at least one common agent vs. subteam size. Fig. 2(b): The average number of times that agent need to be replaced to have at least one common agent.



(a) Team Rewards per Tick



(b) Avg. Message per Agent

Fig. 3. Fig. 3(a) shows relative strengths of different coordination algorithms as a function of time, e.g. simulation "ticks". Fig. 3(b) shows the number of messages that are exchanged by a team in order to achieve a greater team reward.

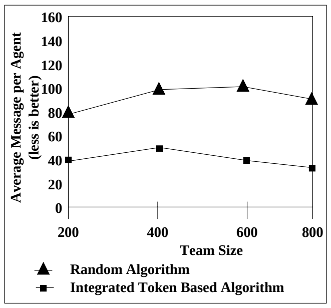
routing i randomly. Using a similar setup, we then measured the variation in the length of time it takes to get a piece of information to the sink. We consider a frequency distribution of the time taken for a network with $8,000$ agents and $MAX\_STEP = 150$. While a large percentage of messages arrive efficiently to the sink, a small percentage get "lost" on the network, illustrating the problem with a probabilistic approach. However, despite some messages taking a long time to arrive, they all eventually did, and faster than if moved randomly.

Figure 2 shows the results of simulations that were run to see how the probability of overlap — and thus the ability of agent subteams to detect conflicts and synergies — varied according to the sizes of two dynamically changing subteams. Two subteams, each composed of from $1-20$ members, were formed from a group of 200. For each subteam size, members were chosen at random and then checked against the other subteam for any common team members. Figure 2(a) shows the calculated percentage of team member overlap when the subteams are initially formed during the simulation. This graph matches closely with the calculated probability $Pr(overlap) = 1 - \frac{_{(n-k)}C_m}{_nC_m}$. In the event that two mutually exclusive subteams are formed, that is, neither subteam has a member in common, we wanted to test the likelihood of membership turnover in each team such that the new member would be part of both teams. Figure 2(b) shows the average number of times that team members needed to be replaced before a common team member was found.

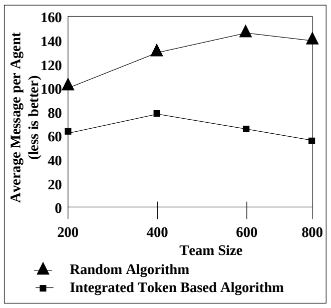
For the TeamSim experiments, a group of 400 distributed UAVs were configured to search in a hostile area. Each UAV had, on average, four "associates", and the network topology was that of a small world network. Simulated data, in the form of 200 simulated automatic target recognition (ATR) events was randomly sensed by UAVs and passed around the team of 400 UAVs as information. Fifty plans, each with four independent information tokens, were allowed in each trial. After a plan was initiated, the tokens which represented its preconditions were retained by the initiating agent and tokens for the three additional roles needed to realize the plan were circulated through the acquaintance network. To accept a role, an agent must be close to the location that the role specifies and have access to tokens for free airspace in order to fulfill the role. Airspace over the hostile area was divided into fifty regions. Each of these regions was represented by three duplicate resource tokens. This restriction limits the risk of collisions by allowing only three UAVs to use a region of airspace at the same time. Each UAV needed to obtain the resource for the region which the role specified before it could perform its task. After all four roles of a plan were successfully implemented, a team reward of 10 units was credited. A maximun reward of 500 units (10 units $\times 50$ plans) was possible. Results for each experiment are based on one hundred trials.

The first experiment investigated how the associates network improves information sharing in the network of team members, as evidenced by the total amount of team reward achieved through an information sharing heuristic. In the simulation, reward was recorded for each tick of the simulation. Five algorithms were compared. In the evaluation of the first algorithm, agents passed tokens randomly. For the next three algorithm evaluations, only one of possibly three token types, *information*, *resource*, and *role*, was preferentially routed throughout the associates network while the other two token types were routed randomly through the TeamSim network. The fifth algorithm evaluated was an *integrated coordination* heuristic, a technique explained in [21] in which tokens of each type are used to improve the routing of the others by opportunistically conveying information of neighbors' preferences and capabilities. The results, shown in Figure 3(a) show how all four associates network based heuristics performed better than random, and that the integrated coordination algorithm performd the best of all. The reader is referred to [21] for more details on the four algorithms.

The second experiment investigated the effect of the coordination algorithms on communication costs. This experiment compared the number of message exchanges required by a team to achieve a particular level of reward. A message was credited to each transfer of a token from an agent to its preferred acquaintance. The same five algorithms — random, three with coordination for single token-types, and the integrated technique — were employed. As shown in Figure



(a) Team Reward = 200



(b) Team Reward = 400

Fig. 4. A measure of efficiency in terms of the average number of messages per agent. This metric is in terms of team reward (200 units in Fig. 4(a) and 400 units in Fig. 4(b)), team size, and coordination algorithm used.

3(b), algorithms using token coordination performed better, in terms of fewer communications to attain the same level of reward, than random. Of the coordinated algorithms, the integrated token routing technique performed best.

The last experiment examined scalability of large teams in terms of the communication load on the individual agent. Teams of 200 to 800 agents were run under conditions that were otherwise identical to the first two experiments. Only two algorithms were used in this experiment: the random algorithm, in which agents chose the associate to which to forward a token, randomly, and the integrated algorithm which utilized relevance between all types of tokens to improve token routing. Performance was measured in terms of *messages per agent*, the number of messages passed by the team divided by the number of agents in the team. As shown in Figure 4, integrated token-based routing produced fewer messages per agent. More significantly it increased at a much lower rate as team size increased from 200 to 800 agents. For both the 200 (Fig. 4(a)) and 400 (Fig. 4(b)) unit award levels, there were fewer messages per agent for teams of 800 agents using the integrated algorithm than for 400 member teams using the random algorithm.

IV. Conclusion

In this paper, we have presented an approach to building large teams that has allowed us to build teams an order of magnitude bigger than previously published. To achieve this aim, fundamentally new ideas were developed and new, more scalable algorithms implemented. Specifically, we presented an approach to organizing the team based on dynamically evolving subteams. Potentially inefficient interactions between subteams were detected by sharing information across a network independent of any subteam relationships. We leveraged the small worlds properties of these networks to very efficiently share domain knowledge across the team. While much work remains to be done to fully understand and be able to build large teams, this work represents a significant step forward.

Acknowledgment

This research has been sponsored in part by AFRL/MNK Grant F08630-03-1-0005 and AFOSR Grant F49620-01-1-0542.

References

[1] D. Goldberg, V. Cicirello, M. B. Dias, R. Simmons, S. Smith, and A. Stentz, “Market-based multi-robot planning in a distributed layered architecture,” in *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, vol. 2. Kluwer Academic Publishers, 2003, pp. 27–38.

[2] P. R. Cohen and H. J. Levesque, “Teamwork,” *Nous*, vol. 25, no. 4, pp. 487–512, 1991.

[3] B. Grosz and S. Kraus, “Collaborative plans for complex group actions,” *Artificial Intelligence*, vol. 86, pp. 269–358, 1996”.

[4] J. A. Giampapa and K. Sycara, “Team-oriented agent coordination in the retsina multi-agent system,” Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-02-34, December 2002, presented at AAMAS 2002 Workshop on Teamwork and Coalition Formation.

[5] M. Tambe, “Agent architectures for flexible, practical teamwork,” *National Conference on AI (AAAI97)*, pp. 22–28, 1997.

[6] N. R. Jennings, “Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving,” *Intl. Journal of Intelligent and Cooperative Information Systems*, vol. 2, no. 3, pp. 289–318, 1993.

[7] P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara, “Scaling teamwork to very large teams,” in *Proceedings of AAMAS’04*, 2004.

[8] D. Watts and S. Strogatz, “Collective dynamics of small world networks,” *Nature*, vol. 393, pp. 440–442, 1998.

[9] M. Tambe, W.-M. Shen, M. Mataric, D. Pynadath, D. Goldberg, P. J. Modi, Z. Qiu, and B. Salemi, “Teamwork in cyberspace: using TEAMCORE to make agents team-ready,” in *AAAI Spring Symposium on agents in cyberspace*, 1999.

[10] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si, and M. Tambe, “A prototype infrastructure for distributed robot-agent-person teams,” in *The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[11] G. Tidhar, A. Rao, and E. Sonenberg, “Guided team selection,” in *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.

[12] B. B. Werger and M. J. Mataric, “Broadcast of local eligibility for multi-target observation,” in *Proc. of 5th Int. Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2000.

[13] D. Shmoys and E. Tardos, “An approximation algorithm for the generalized assignment problem,” *Mathematical Programming*, vol. 62, pp. 461–474, 1993.

[14] P. J. Modi, W. Shen, and M. Tambe, “Distributed constraint optimization and its application,” University of Southern California/Information Sciences Institute, Tech. Rep. ISI-TR-509, 2002.

[15] S. Fitzpatrick and L. Meertens, *Stochastic Algorithms: Foundations and Applications, Proceedings SAGA 2001*. Springer-Verlag, 2001, vol. LNCS 2264, ch. An Experimental Assessment of a Stochastic, Anytime, Decentralized, Soft Colourer for Sparse Graphs, pp. 49–64.

[16] A. Farinelli, P. Scerri, and M. Tambe, “Building large-scale robot systems: Distributed role assignment in dynamic, uncertain domains,” in *Proceedings of Workshop on Representations and Approaches for Time-Critical Decentralized Resource, Role and Task Allocation*, 2003.

[17] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe, “Allocating roles in extreme teams,” in *Proceedings of AAMAS’04, Poster Presentation*, 2004.

[18] B. Clement and E. Durfee, “Scheduling high level tasks among cooperative agents,” in *Proceedings of the 1998 International Conference on Multi-Agent Systems (ICMAS’98)*, Paris, July 1998, pp. 96–103.

[19] M. Paolucci, O. Shehory, and K. Sycara, “Interleaving planning and execution in a multiagent team planning environment,” *Journal of Electronic Transactions of Artificial Intelligence*, May 2001.

[20] 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.

[21] Y. Xu, M. Lewis, K. Sycara, and P. Scerri, “An integrated token-based algorithm for sclable coordintion,” in *Proceedings of AAMAS’05 (submitted)*, 2005.