Team Formation Strategies in a Dynamic Large-Scale Environment

Chris L.D. Jones and K. Suzanne Barber

The University of Texas at Austin
Laboratory for Intelligent Processes and Systems
1 University Station C5000, Austin, TX, 78712-0240
{coldjones,barber}@lips.utexas.edu

Abstract. In open markets and within business and government organizations, fully autonomous agents may form teams to work on large, multifaceted problems. Factors such as uncertain information, bounded rationality and environmental dynamicism can lead to sudden, unforeseen changes in both solution requirements and team participation. Accordingly, this paper proposes and examines strategies for team formation strategies in a large-scale, dynamic environment. Strategies control how agents select problems to work on and partners to work with. The paper includes an experimental evaluation of the relative utility of each strategy in an increasingly dynamic environment, and concludes that a strategy which combines greedy job selection with adaptive team selection performs best in highly dynamic environments. Alternatively, greedy job selection combined with selecting smaller teams performs best in environments with little to no dynamicism.

Keywords: Coalition formation, Task selection, Partner selection, Fault-tolerance, Dynamic environments, Large-scale environments, Request for Proposal.

1 Introduction

Individual agents and multi-agent systems increasingly represent mature technologies, capable of competently solving problems in their specific domains and operating with a high degree of autonomy. Therefore, as computing becomes more ubiquitous and agents consequently become more pervasive, there is an increasingly high probability that an agent exists to help solve any given problem.

These trends, combined with the basic Request For Proposal domain described in [1] suggests the potential for a huge electronic marketplace, where a constant stream of questions or problems is handled by thousands or millions of humans and self-interested agents, each taking on different roles such as general contractor or service provider, each buying or selling skills, expertise, and other services as needed. The potential for such markets arises within government institutions, the military, and business corporations as well.

While many job requests submitted to such a market might be trivial, many are likely to be complex, requiring multiple skills from multiple providers. Furthermore,

N. Jamali et al. (Eds.): CCMMS 2007, MMAS 2006, LSMAS 2006, LNAI 5043, pp. 92–106, 2008.

many of these problems are likely to be novel, undertaken with incomplete information about the problem and a limited understanding of the solution requirements, both of which will almost certainly change as the problem is worked on. Real world problems also occur in dynamic environments, where unexpected changes occur to both the problem and the personnel involved in solving it.

In this type of market environment, many issues arise. Even given a way to accurately determine which agents possess what skill set, how can an agent seeking to maximize its own utility select the best jobs to work on, and the best agents to partner with? How do the partners available to work on different possible jobs influence which job an agent should pursue? How should a team of agents be structured to handle difficulties such as sudden changes to a problem, or defections from the team? And how can these varying requirements be balanced against each other?

One initial attempt at addressing this combination of factors may be found in [2], which draws on previous work in the field in areas such as task selection [3, 4, 5], coalition formation in dynamic environments [6, 7], bottom-up team formation between autonomous agents [8], and team formation in environments with imperfect information [9, 1, 10]. This paper therefore continues the work started in [2] and creates an experimental framework that simulates a decentralized problem-solving marketplace within a very large organization. More particularly, the paper introduces a set of strategies for team formation between autonomous agents in a large-scale, dynamic, decentralized environment, and seeks to determine the relative performance of different classes of agents utilizing these strategies as the rate of change ("dynamicism") of jobs in the environment increases.

This paper is organized as follows. In section two, it discusses related work in the multi-agent systems community. In section three, it describes several job and team selection heuristics, and how these heuristics combine to form team formation strategies in a dynamic environment. In section four, it describes the setup and parameters of a simulation to test the relative utility of the proposed strategies. In section five it discloses the results of the simulation, and analyzes those results. Finally, section six suggests several ideas to expand upon these strategies, and investigate their utility in different settings.

2 Related Work

Coalition formation has been studied both inside and outside of the multi-agent systems community for some time. Some research has focused on the formation of optimal coalitions by a centralized authority, [11] while other research has focused on the formation of coalitions to solve jobs by a hierarchical structure of agents [3]. Still further research has been focused coalition formation between selfless agents in a dynamic [7] or open environment, [5] or between agents willing to delegate their autonomy to a centralized controller or consensus decisions among groups of agents [12].

However, such research has limited applicability to decentralized selfish agents, which may be unwilling or unable to take direction from a centralized authority. Other work has therefore examined selfish agents operating in various environments. Research has focused on building coalitions of agents who lack a common view of a

coalition's value, [19] as well as coalitions developed between rationally-bounded agents, [10] or agents who lack full knowledge about the abilities of potential partner agents [11]. Others have examined the interaction between agents in market environments [13]. Such research frequently focuses on relatively small groups of agents, although still other research has focused on the use of congregations, [14] adaptive social networks, [15] and even physics-motivated mechanisms to allow large groups of agents to form large, mutually beneficial coalitions [16]. It should be noted, however, that such work is frequently focused on only one possible task at a time, or does not occur in dynamic, unpredictable environments.

In contrast, Klusch and Gerber focus on the formation of coalitions of agents to work on multiple possible tasks in dynamic environments, by utilizing a simulation-based scheme to determine the utility of various potential coalitions in a given state of a dynamic environment [6]. This work allows for complex negotiations between the different potential partners of a coalition, and takes risk vs. reward considerations into account when considering different potential coalitions. However, it differs substantially from the work below in that the coalitions formed are not adaptive once formed, nor are jobs selected based on the potential teams available to solve a given job.

Tambe's work is likewise relevant, wherein selfish agents are collected into a team by an initial authority, often a human programmer. The agents may then be delegated by software algorithms into roles which pursue various sub-goals critical to the overall mission [17]. As will be shown in further detail below, the strategies described below build on this research by allowing agents to form adaptive teams without the need for an initial organizing authority.

In addition, Soh and Tsatsoulis have focused on the possibility of hastily-formed coalitions in response to new problems or events [18]. This research forms the basis for one of our heuristics for job selection, as will be described in further detail below.

3 Team Formation Strategies

This paper introduces a set of strategies for team formation between fully autonomous agents in a large-scale, dynamic, unpredictable environment, which is described with a simple, but widely applicable model similar to that used in [15]. Consider a set of general tasks $T = \{T_i\}$, where $1 \leq i \leq \alpha$. Each general task T_i represents a type of job that an agent might carry out: if T is limited to tasks involved in building construction, for example, T_1 might be building a driveway, while T_2 might be constructing a roof, and so on. Each general task T_i is therefore a set of task instances $\{T_{ij}\}$, where each T_{ij} is a specific instance of general task T_i associated with a job J_j, and where each job J_j is part of a set of jobs $J = \{J_j\}$, where $1 \leq j \leq \beta$. For example, if T represents the set of all tasks associated with building a building, and if J is the set of all buildings under construction, then T_{11} might be building a driveway at a first building under construction, T_{12} might be building a driveway at a second building under construction, T_{21} is constructing a roof at the first building, and so on.

Each job J_j in set J contains a potential task instance T_{ij} of every possible task T_i in T, but only a subset of these tasks need to be completed to finish the job. Again, returning to the building example, every building in existence could conceivably have

a swimming pool, or a loading dock, or a conference room, but in practice factories and offices rarely have swimming pools, and houses rarely have loading docks. Accordingly, within each job J_j, task instances T_{ij} are separated into a set of active task instances $ActiveTasks_j$, all of which must be completed for the job to be finished, and a set of inactive task instances $InactiveTasks_j$, which are irrelevant to the job's completion status. For any job J_j, $ActiveTasks_j \cup InactiveTasks_j = \{T_{ij}\}$ for all i, and $ActiveTasks_i \cap InactiveTasks_i = \emptyset$.

Continuing on, a set of skills $S = \{S_i\}$ and a set self-interested of agents $A = \{A_k\}$ are introduced, where once again $1 \leq i \leq \alpha$ and $1 \leq k \leq \chi$. Each skill S_i is associated with a general task T_i, and may be used to work on and eventually complete any task instance T_{ij} in T_i. Furthermore, each agent A_k has an associated set of skills $AgentSkills_k$ that A_k is capable of doing, where $AgentSkills_k$ is a subset of S. Each agent A_k has the same number of skills, and each skill in S equally common among agents in A.

Each task instance T_{ij} has an associated $TaskLength_{ij}$, where $1 \leq TaskLength_{ij} \leq \gamma$. To complete task instance T_{ij}, an agent A_k must use an appropriate skill S_i on the task instance for $TaskLength_{ij}$ timesteps. Accordingly, function $C(T_{ij})$ is defined as a value ranging from 0 to $TaskLength_{ij}$, and represents the amount of time that one or more agents have worked on T_{ij}. Different tasks are worth the same amount of credit, but agents earn rewards proportional to the $TaskLength_{ij}$ of any task instance T_{ij} they have finished. For example, an agent that completes a task over five timesteps earns five credit points, while an agent that completes a task over eight timesteps earns eight credit points.

To simulate the end results of uncertain information, bounded agent rationality and dynamic, unpredictable environments, jobs in J are dynamic and unpredictable. More particularly, task instances T_{ij} in J_j are randomly moved between $ActiveTasks_i$ and $InactiveTasks_i$ on a periodic basis. This may be best understood as a sudden change to a job's solution requirements. For example, despite the best efforts of project management and requirements engineering, software development projects frequently change their required functionality in the middle of development, making some already-completed portions of the project obsolete and requiring new modules to be built from scratch. Similarly, task instances T_{ij} which are moved from $InactiveTasks_j$ to $ActiveTasks_j$ must be done from scratch, while only active task instances which have been fully completed are immune from being moved to $InactiveTasks_j$. (Admittedly, it is not unheard-of for fully-completed portions of many different types of projects to be discarded, but it is also reasonable to argue that work which has been fully completed is often used in some way, somehow, whereas partially completed work is often abandoned entirely.) Note that the number and types of tasks that must be completed for an individual job to be completed is therefore continually changing.

3.1 Agent Strategies

Accordingly, a set of strategies for bottom up team formation between agents in dynamic environments is defined as follows. A strategy is a combination of a job selection heuristic that orders potential jobs in J, and a team selection heuristic that ranks a set of potential teams capable of completing a specific job. More particularly, self-selected foreman agents from A each utilize a job selection heuristic to select one

or more top-ranked jobs from *J*, depending on if one or more jobs are tied for the top ranking. This creates a set of jobs $P = \{J_w\}$.

For each J_w in *P*, each foreman agent then generates ε agent teams capable of solving J_w. More specifically, these agent teams each include the foreman agent which generated the team, and one or more other agents A_k, such that the combined skills of all the agents in the team are capable of completing the task instances in $ActiveTasks_w$ of associated job J_w. These teams thus form a set of teams $Q = \{Team_x\}$. Furthermore, each agent A_k in a $Team_x$ is associated with a set $AssignedInstances_{kw}$, which, after a team has been formed, represents the set of task instances T_{iw} that each agent A_k in the team is assigned to complete in job J_w. Teams are currently assembled via a semi-random approach that seeks to satisfy the various solution requirements one at a time, but nearly any constraint satisfaction solver could also be used. Once *Q* is generated, each foreman agent uses a team selection heuristic to select the top-ranked team from *Q*, which it then attempts to form through protocols described in further detail below.

The first job selection heuristic is a Greedy heuristic (Eqn. 1) that maximizes the expected reward from a job J_j. While a naive heuristic would simply choose jobs that require the greatest amount of work (and thus the greatest amount of associated reward), the Greedy heuristic takes the dynamicism of the environment into account by giving double weight to portions of a task that have already been completed, thereby giving preferential treatment to large jobs that are less likely to undergo changes before the job is complete.

$$\textbf{Greedy heuristic: } \max_{J_j \in J} \sum_{T_{ij} \in AssignedInstances_{kj}} (TaskLength_{ij} + C(T_{ij})) \tag{1}$$

The second job selection heuristic is a Lean heuristic (Eqn. 2) that minimizes the amount of work needed to complete a job, thereby letting agents opportunistically form teams to quickly solve simpler problems, similar to [22].

$$\textbf{Lean heuristic: } \min_{J_j \in J} \sum_{T_{ij} \in AssignedInstances_{kj}} (TaskLength_{ij} - C(T_{ij})) \tag{2}$$

Note that these mechanisms stand in contrast to previous work in task selection under uncertain conditions, such as Hannah and Mouaddib [4], where a problem's uncertain elements are explicitly modeled probabilities. Instead, the heuristics described here operate under any level of uncertainty, from any source. However, future work is possible where the above heuristics are adaptive based on a known or suspected level of uncertainty in the environment, or in a specific problem.

The first team selection heuristic is a Null heuristic that does not rank the teams, but rather keeps teams ordered according to how the strategy's job selection heuristic ranked the jobs associated with each team. This effectively randomly selects one of the teams generated to handle the top-ranked job from *P*.

The second team selection heuristic is a Fast heuristic (Eqn. 3) that minimizes the maximum amount of work that any member of a $Team_x$ needs to complete. Alternatively, the Fast heuristic could be said to minimize the amount of time needed before the entire team has completed work on associated job J_w.

$$\textbf{Fast heuristic: } \min_x \left[\max_{A_k \in Team_x} \sum_{T_{iw} \in AssignedInstances_{kw}} (TaskLength_{iw} - C(T_{iw})) \right] \quad (3)$$

The third team selection heuristic is a Redundant heuristic (Eqn. 4) that seeks to maximize the number of redundant skills in $Team_x$. In other words, the Redundant heuristic prefers teams with multiple agents capable of working on active task instances, thereby increasing the ability of a team to deal with the defection of an agent.

$$\textbf{Redundant: } \max_x \left| \bigcup_{A_k \in Team_x} AgentSkills_k \cap ActiveTasks_w \right| \quad (4)$$

The fourth team selection heuristic is an Auxiliary heuristic that seeks to maximize the number of auxiliary skills in $Team_x$. In other words, the Auxiliary heuristic (Eqn. 5) tries to maximize the combined skills of a team that are not immediately applicable to task instances in $ActiveTasks_w$, thereby increasing the ability of the team to deal with newly added task instances.

$$\textbf{Auxiliary: } \max_x \left| \bigcup_{A_k \in Team_x} AgentSkills_k \cap InactiveTasks_w \right| \quad (5)$$

Note that, intuitively, the Fast, Redundant, and Auxiliary heuristics each prefer a greater number of partners in a team, since this increases the amount of work that can be done in parallel and the number of unused skills for each partner. Alternatively, the MinPartner heuristic (Eqn. 6) prefers teams with the smallest number of partners, thereby implicitly using a greater number of skills per partner and thus a greater amount of potential profit per partner.

$$\textbf{MinPartner heuristic: } \min_x \left[|Team_X - 1| \right] \quad (6)$$

Each of these five team selection heuristics is combined with each of the job selection heuristics for experimental comparisons, as described in further detail below.

4 Experimental Setup

To evaluate the relative utility of each of the team formation strategies described above, these strategies are tested in a simulation environment wherein agents compete to form teams and solve jobs according to the described strategies. More particularly, a set of agents is divided into ten different classes, each of which contains an identical number of agents, and each of which implements a different team formation strategy. By assigning agents credit for completing task instances, the relative utility of each strategy may be determined by comparing the average amount of credit earned by each class of agents. Furthermore, by varying the rate of change of the solution requirements for different jobs ("dynamicism"), the relative performance of these strategies in a dynamic environment can be determined.

Agents in set *A* operate in a simulation environment that is divided into discrete timesteps, or rounds. During each round, each agent may coordinate with other agents in *A* to form teams, or, if it is part of a team, may work on a task instance associated with a specific job in *J*. Each agent in *A* can belong to, at most, one team at a time, and each team works on only one job at a time. This is arguably a simplistic assumption, since real world providers of valuable skills or expertise frequently multitask between different projects at the same time. However, many problem solutions require complete focus from the workers involved, or security or other constraints may require exclusivity. Furthermore, requiring each agent to be part of only one team at a time allows us to clearly delineate where an agent is making a contribution. Determining to what degree an agent's partial efforts require task reassignments touches on complex multidimensional trust issues [19], and as such is too complex to be addressed here.

During each round, an agent A_k may work on a job J_j by utilizing a skill S_i found in $AgentSkills_k$ to work on a task instance T_{ij} found in set $ActiveTasks_j$. Each agent utilizes only one skill in any given round. After A_k has worked on T_{ij} for a given number of rounds, T_{ij} is completed.

Credit is distributed to agents when a job J_j is completed, which, in turn, occurs when all task instances in $ActiveTasks_j$ are completed. Upon job completion, credit points for each active, completed task are given to the agent which completed the task. As described above, these credit points are proportional to the length of the completed task (e.g. a completed task of length five would give five credit points). No credit is given for work on task instances that were moved to $InactiveTasks_j$ before completion, or to agents who worked on, but did not finish, a completed task instance. Accordingly, agents in the simulation may be said to work in a "pay-for-play" environment, where credit is distributed directly to those who have fully completed a given job.

Once a job J_i has been completed and paid out its credit, it is removed from *J* and a new J_j is created and inserted in *J*. Each new J_j starts with the same number of task instances randomly placed in $ActiveTasks_j$, and task instance in the new job must be completed from scratch.

As described above, the simulation incorporates unpredictability by shuffling task instances between $ActiveTasks_j$ and $InactiveTasks_j$. More particularly, each round a given percentage of jobs in *J* are randomly selected to be shuffled. This percentage is referred to as the dynamicism of the simulation. Each task instance in each selected job J_j has a random chance of being selected for shuffling between $ActiveTasks_j$ and $InactiveTasks_i$, such that, on average, one task instance per selected job is shuffled. However, as described above, task instances that have been fully completed cannot be moved from $ActiveTasks_j$.

4.1 Team Formation

As described above, teams are formed by a foreman agent. The opportunity to act as a foreman agent is randomly distributed among agents, such that any given round of the simulation a given percentage of agents will have the opportunity to form teams. Once the foreman agent has used its associated team formation strategy to select a potential team $Team_x$, the foreman sends proposal messages to potential partners in

$Team_x$ indicating the $AssignedInstances_{kw}$ that a potential partner would work on. Note that, to encourage agents to form teams, $|AgentSkills_k|$ is constant for all k, and the initial value of $|ActiveTasks_i| > |AgentSkills_k|$.

When these proposal messages are received, each agent ranks the $AssignedInstances_{kw}$ it is currently working on (if any) against one or more proposed $AssignedInstances_{kw}$ using the job selection heuristic associated with that class of agent. If the agent finds that its current assigned tasks are preferable to any of the proposals, it continues working on its current job, and the lack of a response is taken as a decline message by the foreman which sent the proposal. If the agent receives a proposal it finds more attractive than its current job assignment, the agent returns an accept message to the foreman which sent the proposal.

Accordingly, it is noted that agents may stop work on their current assignments at any time upon receiving a more attractive proposal (or, if they become a foreman agents, upon finding a more attractive job to work on). This obviously runs counter to a significant amount of work which his been done in contract negotiation and breaking contracts [20]. However, such work usually involves complete information, and/or occurs between a relatively small number of agents. In contrast, the scheme described here allows for agents which are better able to take advantage of new opportunities, and better simulates many environments where contracts are largely nonexistent or unenforceable (i.e. informal task forces and many Internet transactions). This lack of commitment between agents, combined with the dynamicism and unpredictability of jobs within the simulation, also makes it desirable for agents to assemble teams that can survive agent defections and changes in the task instances required to finish the job.

If the foreman does not receive accept messages back from all potential partners, the team formation process has failed and the foreman, as well as the agents which accepted the team proposal, must wait for new team proposals or for their next chance to be a foreman. If the foreman receives accept messages from all of its potential partners, the team is successfully formed and the foreman claims J_w, thereby making it off limits to other teams. Jobs may be claimed only by foremen who have successfully formed full teams. Note that during this process, other agents may be attempting to assemble a team to handle the same job, thereby simulating a realistic "churn" in which a degree of effort is unavoidably lost to competition. Jobs are claimed by means of a lock mechanism which prohibits "ties" between agents trying to claim a job.

Once agents have formed $Team_x$, they begin to work on the task instances associated with J_w. Non-foreman agents may work on J_w until they have completed all task instances in $AssignedInstances_{kw}$. In contrast, the foreman agent may stay with J_w until the job is complete, even if the foreman has completed its assigned tasks. While J_w is incomplete, if a non-foreman agent defects from the job, or a new task instance is moved into $ActiveTasks_w$ set, the foreman is responsible for finding an agent to work on the new or abandoned task instance. The foreman may therefore assign the new task instance to the $AssignedInstances_{kw}$ set of itself or a partner agent, in a manner similar to the team reformation strategies in [17]. If no team member has the skill required to handle the new task, the team has failed and dissolves with the job uncompleted. A new team which later tries to claim the job must begin the job from

scratch. It therefore follows that teams must handle defections and new task instances to be successful.

Furthermore, it should be noted that a balance must be achieved when forming a new team with regard to the number of agents recruited. Because only one decline message will prohibit the formation of a new team, or will break an existing team seeking to recruit an outside member, larger teams are more difficult to form. Furthermore, team members running the Greedy job selection heuristic are more likely to defect from a larger team, since, on average, they will be assigned less work and thus have less opportunity for profit. However, larger teams are better able to handle agent defections and new task instances within the team, making them more stable once formed.

Table 1. Experimental parameters

Parameter	Value
Number of classes	10
Agents per class	250
Per round chance of agent acting as foreman	1%
Jobs	1000
$\|T\|$	20
$\|AgentSkills\|$	5
Initial size of $\|ActiveTasks\|$	10
Range of TaskLength	1 to 10 rounds
Credit received per round of completed task instance	1
Number of potential teams examined per top-rank job	15
Dynamicism range	0% to 100%, 25% increment
Number of rounds per simulation	2500
Number of simulations per dynamicism step	20

Experiments were conducted using the basic parameters in Table 1, which were selected to broadly model a problem-solving market internal to an organization such as a large corporation or a moderately large number of freelance agents. The experiments tested all strategies against each other simultaneously to see which strategies were most successful in a field of heterogeneous agents. More particularly, a two-factor ANOVA analysis was carried out, and the Fisher LSD method was used to determine significant differences between different strategies at the same dynamicism level.

5 Results and Discussion

Figure 1 displays the average credit earned by each agent class at different levels of dynamicism, along with standard deviation bars for each value. A cursory examination of the data indicates that the credit earned by each class decreased as the level of dynamicism in the environment increased, and that while GreedyMinPartners was the most successful strategy for 0% and 25% dynamicism, GreedyAuxiliary was

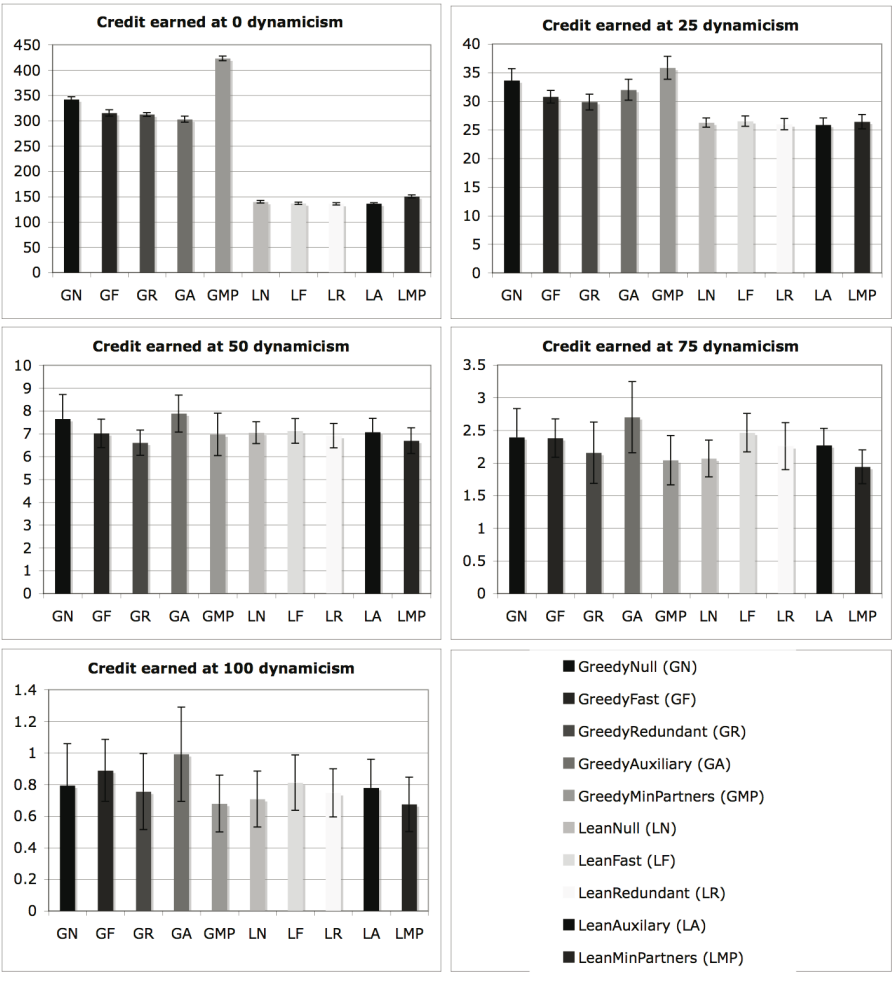

Fig. 1. Strategy performance as a function of dynamicism

the most successful strategy for all other dynamicism levels. ANOVA analysis (alpha = .05) indicates that the strategy selection causes statistically significant differences in the results.

Furthermore, Fisher LSD tests (alpha = .05 for all comparisons below) indicate that the top-performing strategy for each dynamicism level was statistically significant from all other strategies, with the exception of 50% dynamicism, where GreedyAuxiliary and GreedyFast were statistically significant from all other strategies but not from each other, and 100% dynamicism, where GreedyAuxiliary was significantly better than all other strategies, save GreedyFast.

Similar results can be seen when examining the aggregate performance of each team selection heuristic, regardless of the job selection heuristic it was paired with: at

0% dynamicism, the MinPartner performed significantly better than all other heuristics, and better than all but the Null heuristic at 25% dynamicism. At 50% dynamicism, the Auxiliary heuristic performed significantly better than all other strategies save the Null heuristic, while at 75%, and 100%, the Auxiliary heuristic performed significantly better than all heuristics but the Fast heuristic. Comparing the aggregate performance of the job selection heuristics, the Greedy job selection heuristic performed significantly better than the Fast heuristic at all dynamicism levels. Note that this differs from the results found in [2], where credit could sometimes be earned from unsuccessfully completed jobs.

To better understand these results, it is helpful to know how successful different classes are at forming teams, as shown in Figure 2 as the ratio between the number of team formation requests by a foreman and the number of teams successfully formed. Furthermore, Figure 3 displays the average success rate of formed teams at completing an assigned job as a function of both agent class and dynamicism level.

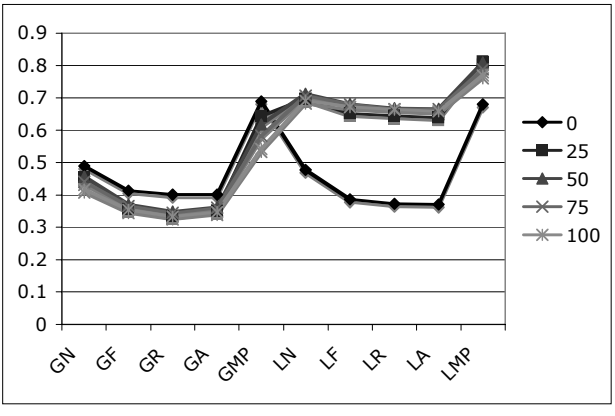

Fig. 2. Ratio of teams successfully formed by class and dynamicism level

As can be seen in Figure 2, certain patterns are immediately obvious. Agents using Lean job selection strategies are significantly more successful at forming teams than Greedy job selection strategies, except at 0% dynamicism, when job requirements do not change. This may be due to the fact that, in dynamic environments, a subset of jobs will have a lower than average number of tasks, which are more attractive to agents using the Lean job selection heuristic, thereby improving the probability that these agents will join a team.

Furthermore, agents using the MinPartner team selection heuristic are far more likely to successfully form a team. As discussed below, this is likely due in part to the fact that a smaller number of partners means a smaller number of potential rejections, any one of which can break team formation. In addition, a smaller number of agents means a higher number of tasks assigned to each agent, and accordingly, a greater amount of profit for each agent.

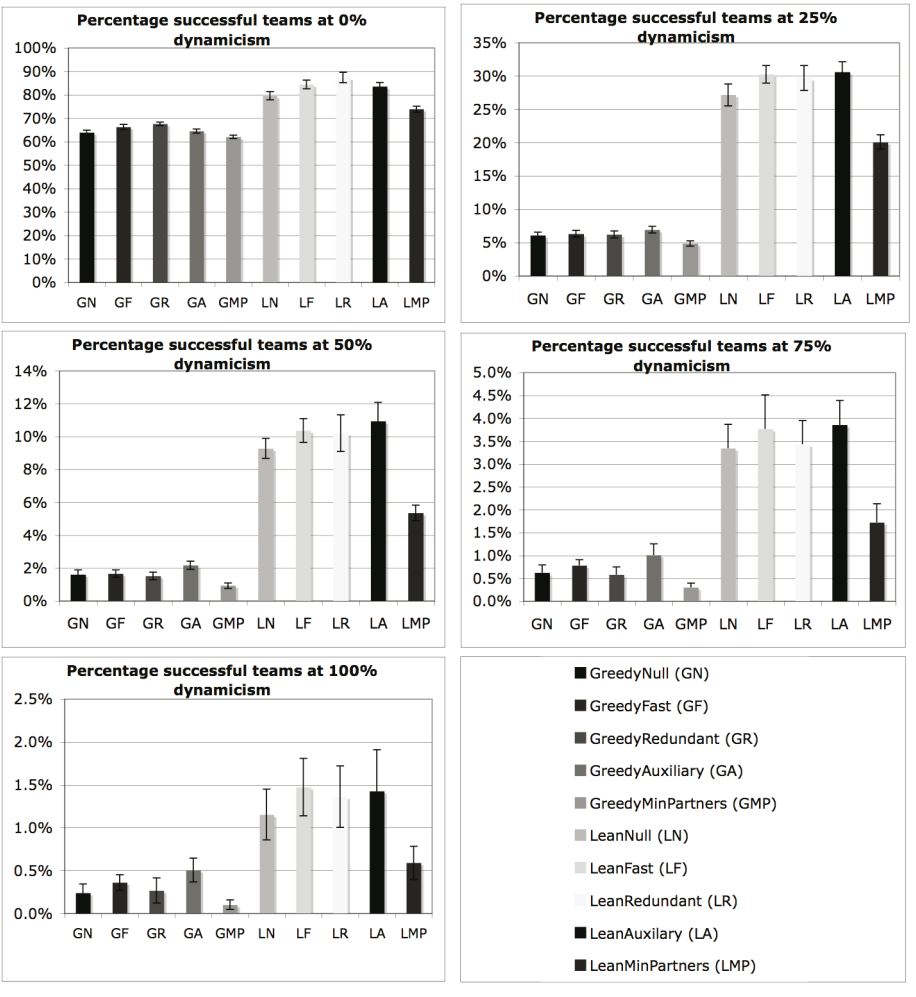

Fig. 3. Teams successful at completing jobs by class and dynamicism level

Conversely, Figure 3 shows that, although agents which utilize the MinPartners team selection heuristic are the most successful at forming teams, they are consistently the least successful agents at forming teams which successfully complete their assigned jobs. Alternatively, agents which utilize the Fast, Redundant, and Auxiliary heuristics are consistently the most successful at forming successful teams. In fact, ANOVA and LSD analysis (alpha = .05) indicates that, although agents utilizing these three heuristics do not have success rates that are always significantly different from each other, they are almost always significantly different from agents using the Null and MinPartner heuristics. Again, as discussed above, this is likely due to the fact that teams formed according to the Fast, Redundant and Auxiliary heuristics are more likely to have team members with unused skills that can be used in

the event that another team member leaves the team, or a new task is added to the list of *ActiveTasks.*

This result suggests that agents which can recognize a team with significant redundant or adaptable characteristics may be more successful than agents which cannot. In other words, agents which can recognize a proposed team that has a lower offered reward but a greater opportunity for success because of its redundant or auxiliary resources may be able to earn more credit.

It is also noted that, although agents using the Lean job selection heuristic are significantly more successful at both forming teams and successfully completing jobs, these agents are generally less successful than agents which use the Greedy job selection heuristic. This may be in part due to the fact that Lean agents generally select jobs with far less work involved, and thus earn far less credit per job. Accordingly, it may be possible to determine under what circumstances a Lean vs. Greedy job selection heuristic is preferred, or even to find an optimal heuristic which combines the two considerations, based on environmental characteristics.

6 Conclusions and Future Work

This paper presented a set of strategies for team formation between fully autonomous agents in a large-scale dynamic environment. Strategies prioritized which jobs and partners each agent selected, and were composed of a combination of one of two job selection heuristics and one of five team selection heuristics:

- Greedy job selection, which selects jobs according to profit potential
- Lean job selection, which selects jobs for minimal time to completion
- Null team selection, which defers to the associated job selection heuristic
- Fast team selection, which selects teams for minimal time to completion
- Redundant team selection, which selects teams to best handle partner failures
- Auxiliary team selection, which selects teams most capable of adapting to new tasks
- Minimum Partner team selection, which selects teams with the fewest number of partners

The paper then analyzed the relative performance of the strategies, as executed by different classes of agents, in an experimental test bed. Results indicated that a strategy combining the Greedy and Auxiliary heuristics performed the best in highly dynamic environments, while a strategy combining the Greedy and MinPartner heuristics performed better in environments with low or no dynamicism.

Furthermore, the paper showed that, although agents using the MinPartner heuristic were more successful at forming teams, these teams were significantly less successful at completing jobs than teams formed by agents using the Fast, Redundant, and Auxiliary team selection heuristics.

This paper therefore examined an unpredictable, dynamic domain of independent heterogeneous agents, and proved which of a given set of strategies for team

formation best maximized agent utility in such a domain. These strategies work with very large groups of agents (>2500) and may potentially be applied to request for proposal (RFP) environments where teams of autonomous agents continually seek to solve dynamic problems by forming multi-skilled teams. These market environments potentially include Internet-wide open markets, and markets internal to large organizations such as corporations and government agencies.

A number of possible avenues of further investigation suggest themselves. One possibility is to modify the mechanism by which agents decide between joining potential teams; agents may be more successful if they can balance the reward a team offers for membership against that team's ability to adapt to change. Other potential work involves determining in what circumstances a Lean job selection heuristic is preferable to a Greedy job selection heuristic, and potentially finding an optimal balance between the two.

Acknowledgments. This research is sponsored in part by the Defense Advanced Research Project Agency (DARPA) Taskable Agent Software Kit (TASK) program, F30602-00-2-0588. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

1. Kraus, S., Shehory, O., Taase, G.: Coalition Formation with Uncertain Heterogeneous Information. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1–8. ACM Press, New York (2003)
2. Jones, C., Barber, K.: Bottom-up Team Formation Strategies in a Dynamic Environment. In: AAMAS 2007 Workshop on Coordination and Control in Massively Multiagent Systems, pp. 60–72. ACM Press, New York (2007)
3. Abdallah, S., Lesser, V.: Organization-Based Cooperative Coalition Formation. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 162–168. ACM Press, New York (2004)
4. Hanna, H., Mouaddib, A.: Task Selection Problem Under Uncertainty as Decision-making. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1303–1308. ACM Press, New York (2002)
5. Shehory, O., Kraus, S.: Methods for Task Allocation via Agent Coalition Formation. Artificial Intelligence 101, 165–200 (1998)
6. Klusch, M., Gerber, A.: Dynamic Coalition Formation among Rational Agents. IEEE Intelligent Systems 17, 42–47 (2002)
7. Nair, R., Tambe, M., Marsella, S.: Team Formation for Reformation. In: Proceedings of the AAAI Spring Symposium on Intelligent Distributed and Embedded Systems (2002)
8. Rathod, P., desJardins, M.: Stable Team Formation Among Self-Interested Agents. In: Working Notes of the AAAI 2004 Workshop on Forming and Maintaining Coalitions in Adaptive Multiagent Systems. AAAI, Menlo Park (2004)
9. Ketchpel, S.: Forming Coalitions in the Face of Uncertain Rewards. In: Proceedings of the Twelfth National Conference on Artifical Intelligence, pp. 414–419. AAAI, Menlo Park (1994)

10. Sandholm, T., Lesser, V.: Coalitions among Computationally Bounded Agents. Artificial Intelligence 94, 99–137 (1997)
11. Sen, S., Dutta, P.: Searching for Optimal Coalition Structures. In: Proceedings of the Fourth International Conference on Multi Agent Systems, pp. 287–292. ACM, New York (2000)
12. Martin, C.: Adaptive Decision Making Frameworks for Multi-Agent Systems. Ph.D. Thesis, University of Texas, Austin, TX (2001)
13. Wellman, M., Wurman, P.: Market-Aware Agents for a Multiagent World. Robotics and Autonomous Systems 24, 115–125 (1998)
14. Brooks, C., Durfee, E., Armstrong, A.: An Introduction to Congregating in Multiagent Systems. In: Proceedings of the Fourth International Conference on Multi Agent Systems, pp. 79–86. MIT Press, Cambridge (2000)
15. Gaston, M., desJardins, M.: Agent-Organized Networks for Dynamic Team Formation. In: Proceedings of the fourth international joint conference on Autonomous Agents and Multiagent Systems, pp. 230–237. ACM Press, New York (2005)
16. Lerman, K., Shehory, O.: Coalition Formation for Large-Scale Electronic Markets. In: Proceedings of the Fourth International Conference on Multi Agent, pp. 167–174. MIT Press, Cambridge (2000)
17. Tambe, M., Pynadath, D., Chauvat, N.: Building Dynamic Agent Organizations in Cyberspace. IEEE Internet Computing 4, 65–73 (2000)
18. Soh, L., Tsatsoulis, C.: Satisficing Coalition Formation among Agents. In: Proceedings of the first international joint conference on Autonomous Agents and Multiagent Systems, pp. 1062–1063. ACM Press, New York (2002)
19. Griffiths, N.: Task Delegation using Experience-Based Multi-Dimensional Trust. In: Proceedings of the fourth international joint conference on Autonomous Agents and Multiagent Systems, pp. 489–496. ACM Press, New York (2005)
20. Faratin, P., Klein, M.: Automated Contract Negotiation and Execution as a System of Constraints. MIT, Cambridge (2001)