Building large-scale robot systems: Distributed role assignment in dynamic, uncertain domains

Alessandro Farinelli, Paul Scerri and Milind Tambe

Dipartimento di Informatica e Sistemistica
Univerista' di Roma "La Sapienza"
Via Salaria 113 CAP 00198 Rome, Italy
Computer Science Department
University of Southern California
941 W 37th Street, Los Angeles, CA 90089
farinell@dis.uniroma1.it, scerri@isi.edu, tambe@usc.edu

Abstract

For robot teams in large-scale, real-world domains, an effective approach for allocating roles to team members (role allocation) is critical. Unfortunately, role allocation is extremely challenging in such real-world domains since robots face significant uncertainties in their own capabilities and they may be faced with dynamic and continuous changes in their capabilities. Previous work in multi-robotic and multiagent systems has provided algorithms for role allocation, but these algorithms often fail to explicitly represent and reason with uncertainty in a robot's own capabilities, and often do not address dynamic continuous changes in such capabilities. This paper presents two key contributions to address these limitations in large-scale settings. First, it presents a novel role allocation and re-allocation algorithm that explicitly reasons with uncertainty and dynamic changes in robot's own capabilities. Second, it presents the application of a proxy-based architecture (previously applied only in agent-human settings) in large-scale robotic domains — demonstrating the application of a reusable infrastructure for multirobotic domains. Moreover this paper presents an experimental study based on simulations performed in order to evaluate the algorithm. These simulations pave the way to large real-world practical multi-robotic system involving 100s of robots.

1 Introduction

Large numbers of robots are capable of achieving complex goals in distributed environments, provided they act together effectively. *Teams* of robots can achieve goals that individual robots or uncoordinated groups of robots cannot achieve. However, achieving effective teamwork in a dynamic, uncertain environment with robots that do not always have accurate models of their current situation and capabilities is very challenging. Algorithms for coordinating the robots must deal with the inherent distribution of the team, unpredictable robot behavior uncertainty and dynamics in the environment.

A key requirement for distributed team coordination algorithm is an effective approach for distributing the responsibility for roles among the team members (*role allocation*). In this paper, we focus on real-world domains that provide a significant challenge for distributed role allocation due to three key properties. First, these domains are highly uncertain, and in particular, robots may face significant uncertainties in their own capabilities. This uncertainty goes beyond a low-level action (e.g., movement) uncertainty, and fundamentally relates to a significant uncertainty in how well a robot can fulfill a role. Second, these domains are highly dynamic, where external or internal events may cause significant dynamic changes to a robot's capabilities. These changes are not just binary (from a fully functioning robot to a completely failed robot); rather, there may be a continuous dynamic degradation in robot capability over time. Third, these domains may involve a very large number (100) of robots.

While there have been significant investigations of such role allocations in both the distributed robot and multiagent literature, these are largely inadequate in addressing a combination of the three challenges listed above. For instance, previous work in multiagent systems has provided algorithms for optimal initial role allocations based on capability analysis [14], combinatorial auction [6], distributed constraint optimization [8] and Belief Desire and Intension [12]. Role reallocation has also been considered: team reorganization based on intends.that in the Shared Plan approach [5] or critical role failures in STEAM [13]. Within the distributed robotic literature, authors have proposed methods using auctions[16, 4], others utilize behavior based architectures[9, 15]. While these algorithms perform well for small teams, as applications are scaled to larger and more dynamic and uncer-

tain environments, several limitations are apparent. First, clearly, given that robot's own capabilities may change dynamically over time, an optimal initial allocation is insufficient and inefficient. Second, given the large uncertainty in robot capability, it is critical to represent and reason with such uncertainty explicitly. Unfortunately, such explicit reasoning is often missing in the above approaches. Finally, while many role allocation approaches are explicitly designed to handle robot or agent failures, dynamic and continuous degradation in capabilities raises novel challenges in large scale robotic teams.

We provide two key contributions in this paper to address the above shortcomings. First, we provide an architecture, based on proxies[11] for coordination among multiple robots. While such proxy-based architectures have been previously demonstrated to allow effective coordination among teams of intelligent software agents and people[11, 10, 7], this paper takes the first step towards applying such architectures to robotic domains. Each robot is provided with a *proxy* that is in charge of enabling teamwork among the team members. The proxies encapsulate the coordination reasoning required to coordinate a heterogeneous team. The proxies have a variety of responsibilities including initiating and terminating team plans, monitoring the progress of team plans and communicating relevant information among team members.

A second and more important contribution of this paper is an approach for role allocation, that addresses the three role allocation challenges discussed above. Our approach relies on the algorithm described in [11] This algorithm is a distributed approach for role allocation that guarantees very low communication overhead and allows roles allocation and mission execution to occur in parallel. Moreover the algorithm focuses on quickly finding an allocation of capable robots to roles, rather than spending time finding optimal role allocations. Therefore this algorithm is particularly well suited for large scale team in real-world domains. However the previous algorithm is not able to deal with dynamic and continuous capabilities changing and uncertainty in the robot performance.

We extended the previous algorithm in order to deal with those issues, by exploiting two key ideas. First, the team considers reallocating roles whenever any robot notices a significant change in its estimate of its ability to complete its assigned roles and when it notices a significant change in the uncertainty in its estimates of performance. The uncertainty is measured exploiting domain level knowledge, e. g. the probability distribution of the robot position. Triggering a reallocation process at these points provides an opportunity for the team to avert failures before they occur. Second, we provide two mechanisms for changing roles. The two mechanisms have distinctly different properties and the robots choose between them based on the current environmental situation. The first mechanism lets the robot give up the role without any guarantee that the role will be taken by someone else in the team. The second mechanism requires that the robot does not relinquish the role until another robot has accepted responsibility. The second mechanism is less efficient at finding another robot to take on the role but ensures that at least one robot is always working on it.

	No Unc. model	Expl Unc. Model
Bin. Cap. Reall.	[4, 16]	
Cont. Cap. Reall.	[9, 1, 15]	OUR APPR.

Table 1: *Taxonomy for the Dynamic role assignment*

In order to provide a better view of the relationship among our work and the previous approaches we present in table 1 a possible taxonomy of the different approaches for role allocation. On the vertical axes we consider how the changes in the robot capability are taken into account for the reallocation process while in the horizontal axes we consider the modeling of uncertainty in the robot capabilities. In the upper left corner we put those works that do not address the issue of reallocation or that consider a possible reallocation only when the robot has failed in the role execution. this can be seen as a *Binary* policy for reallocation that consider only a full or an empty value of the robot capability for the role assignment. In the lower corner we put those work that consider the reallocation as the robot capability for the role gradually changes during the role execution. On the lower right corner we put our approach because it can both deal with role re-allocation as a significant change occur in the robot capability and explicitly model uncertainty for the robot performance.

While our algorithms could potentially be applicable in a variety of domains, one concrete application for our work is the *Software for distributed robotics* (SDR) project. The project aims at building a large-scale distributed robotic system (with 100 robots) that will be actually deployed in the real world later this year (2003). These 100 robots must guard a building for 24 hours and detect intruders. While the project is a collaborative effort involving several research groups (some of the other organizations involved include SAIC, University of Tennessee and Telecordia), our focus is on the use of proxies and in particular on role allocation and reallocation in critical situations. One such critical situation arises due to the need for a 24 hour operation: robots must be periodically recharged. In our work, intelligent "shepherd" robots must cooperate to bring simple sensor robots to a charging station and back to their location in the environment.

In order to conduct some experiments on this scenario we developed a simulation of the critical *battery re-*

charging scenario discussed above. The simulator models in some detail the uncertainty in action and perception faced by real robots. The experiments performed in the simulated environment show that while the presented approach seems to be promising, more effort should be conducted in order to have a better evaluation of our approach. The remainder of this paper is organized as follows. In section 2 we describe our application domain in greater detail, and give a more formal definition of our approach, in section 3 we present a sketch of the basic algorithm and describe in detail the novel extensions, discussing the new issues involved. In section 4 we present the results obtained in our simulation environment. In section 5 we compare our approach to existing techniques and finally in section 6 we draw the conclusions from our work.

2 Problem Definition

In this section we give a proper definition of the problem we are addressing, we describe in detail our application domain and give the basic motivation for the original role allocation extension. We consider a set of roles to be executed $R = \{r_1, ..., r_m\}$, and a set of robots $A = \{a_1, ..., a_n\}$. Each roles can be considered as a complex action that the robot should perform in order to reach a goal. A robot A_i could be assigned to more than one role R_j, we define the set $AS_i = \{R_j \text{ s.t. } R_j \text{ is assigned to } A_i\}$. In our framework we assume that all the robots are part of a team and that each of them is performing its assigned role(s) in order to achieve a common shared goal for the overall team. The problem is to assign each robots to a set of roles in the most effective way, facing the dynamic evolution of the environment, the uncertainty of the robots about their capabilities and the real time constraint of the application.

Dynamism implies that after a robot a_i has been assigned to a set of roles $R_a = r_1, r_2, .., r_k$, during the execution of one of the roles, the robot's ability to perform r_i may depletes. Thus there is a chance the robot may not be able to successfully execute all the roles in R_a. Moreover the robot has a degree of uncertainty on its capabilities in performing a given role r_i. This implies that the robot can not collect enough information in order to decide if it is capable of fulfilling a particular role r_i. The level of uncertainty that the robot has about its capabilities varies during the role execution, accordingly to the robot internal state and environment configuration.

Our domain of application is the SDR (software for Distributed Robots) project. This project involves several university and research institutes and aim at building a very large robot system that should be able to patrol a building and detect intrusion. The system is composed of heteregeneous robots; we have several different kinds of robots, with different capabilities that should coordinate among themselves in order to reach the common goal. One of the crucial aspect of the project is that the system should be able to perform its mission autonomously for a long period (24 hrs). Thus the robots must be able to re-charge themselves autonomously. For this particular aspect of the problem we consider two different kinds of robot: Comms sensor robots (CSR) and Comms helper robots (CHR). The CSR are very simple (e.g. do not have localization capability) and are used as sensors to patrol the environment. They are not able to navigate autonomously in the building thus in order to reach the charging station they need that more sophisticated robots (CHR) drive them all the way to the charging station and back. The CHR have the capability of autonomously navigate the building. While it takes only a single CHR to bring a CSR to the re-charge station, we have only a limited number of CHR, and thus we need to have an efficient way to allocate them to the sensors, in order to avoid that none of the CSR runs out of power.

In this domain we focus on the role assignment of CHR to CSR. We have a set of CHR $A = CH_1, ..., CH_n$ and a set of roles $R = CS_1, ..., CS_m$. A role CS_j for a given CH_i is to pick up a the j-th CSR, drive it to the re-charge station and bring it back to its original locations. A CSR CS_j can be picked up by only one CHR CH_i, but the CHR can have more than one assigned CSRs. The common goal for the team is to avoid that any of the CSR robot runs out of power.

As the environment is unpredictable and the robots does not have perfect and complete information about their state, each robot of the team should monitor and reconsider its role assignment, in order to face as best as they can any kind of unexpected situations. In particular let's say we have a particular assignment of robots to roles, that assign the CHR CH_i to the CSRs CS_j, CS_K

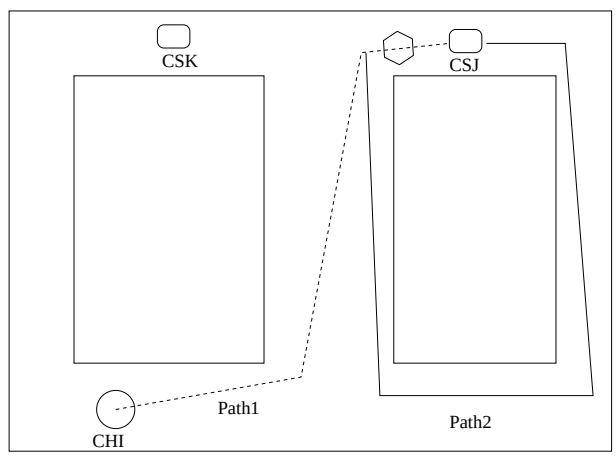

Figure 1: *Motivating example*

This role allocation entails that the CHR CH_i believes it

can reach both the CSRs and bring them to the re-charge station before their batteries run out of power. During the execution of the role however the environmental configuration may vary arbitrarily thus it may be the case that while trying to reach the CSR CS_j the robot discovers that an unexpected obstacle obstructs the shortest path to CS_j. The only option CH_i has is to take a longer way to CS_j. However this detour from the original planned path will increase the time CH_i needs to reach both CS_j and CS_k possibly resulting in one of the robots to run out of power. In such a case the original role allocation algorithm would not take into account a role re-allocation because the changing in the capabilities of the robot are not monitored until a failure occur. However in this domain a failure is unrecoverable because a CSR that runs out of power can not be restarted. So in this case what we really need is the CHR to reconsider its assignment and decide to reallocate to some one else in the team the role it cannot perform anymore before the failure occurs. Moreover the CHR could be uncertain on the results of its performance. For example a robot while executing its role could have a degradation of its localization capabilities. Consequently the robot will not be sure anymore if it can still perform successfully the assigned role(s). The role performance degradation could possibly lead to have one of the CSR running out of battery so we may need to reconsider the role assignment. On the other hand in such a case we need the robot to give up the present role only when it is sure that some one else in the team could perform the role better. In order to take this decision the role allocation algorithm should take into account some measure of the uncertainty the robot has about its position, and thus having the robots to exchange more detailed information about their status.

In order to test our approach we set up a simulator of the SDR scenario. The simulator gives us the possibility to introduce in the domain the uncertainty of the robots about their position, and test the performance of the different re-allocation mechanisms. We modeled in the simulator the robot uncertainty about their position by implementing a Bayesian method for the robot localization, and considering a simple model for the robot motion and observation uncertainty.

3 Algorithm for the role allocation approach

In this section we first describe in detail the basic role allocation algorithm developed in previous work[11] and then two extensions presented in this paper. We model each robot as being in a space of possible states, S_i. The states of interest in this case are those dynamic features of the robot that influence its capability to perform roles. For example, a CHR robot i's local state space, S_i, could consist of all possible combinations of CHR positions and the CSRs it is currently responsible for. We define the set of roles, R, to be the set of all possible instances of role (e.g. picking up a CSR) that can arise during the mission execution. Each such role, $r \in R$, has its own set of possible states, S_r. For example, a role associated with picking up a particular CSR will have state information representing the battery level of the CSR.

To match robots to roles, we require a representation of each robot's ability to successfully execute each role. As a starting point, we assume that the architecture has a quantitative representation of each robot's dynamic capabilities. More precisely, we represent the capabilities of a robot i as a function, $c_i : S_i \times R \rightarrow [0, 1]$, that maps the robot's current state and a possible role into a quantitative estimate of that robot's ability to succeed at the given role. For example for a CHR the capability of re-charging a CSR could be a function of the distance that the CHR should cover in order to re-charge the CSR, representing the fact that increasing this distance diminishes the CHR capability of fulfilling the role.

In evaluating the assignment of a robot to a role, we need to measure not only the capability of that robot to perform the role but also the particular relevance for that role to the overall mission execution. For example, the CSR battery levels evolves in different way as the mission is being executed, and we need to consider this differentiation among the CSR. In order to have a more effective role allocation algorithm we model the distinctions among roles by associating a *priority* with each of them. The priority of a role, r, maps its state (e.g., battery level of a CSR) into a total ordering over roles: $p_r : S_r \rightarrow [0, 1]$

The basic role allocation algorithm, assigns roles to robot by passing the responsibility for a role around the robots, until a robot accepts the role or all robots reject the role. Each robot can consider whatever factors it likes to when deciding whether to accept a role, e.g., it could see how capable it is at performing the role or the importance of the role. In the experiments presented below, a CHR will accept a role to collect a CSR if it calculates that it can most likely collect the CSR before the CSRs battery runs out, given that it may also have responsibility for collecting other CSRs. If it chooses to accept the role, other robots are not offered the role and the robot attempts to execute the role, using the resources at its disposal. If a robot accepts a role, it will continue to work on the role until one of three things happens: (a) it succeeds in completing the role; (b) it fails to perform the role; or (c) it accepts a higher priority role that requires it pass the role onto other robots. If all robots reject the role, the team puts the role aside for some time before passing it around again.

While this basic algorithm has been shown to work effectively for highly heterogeneous teams in very dynamic environments [11], it has two clear limitations when applied to teams of robots. Firstly, once a robot has accepted a role, if it comes to believe that it is unlikely to fulfill the role or it becomes unclear whether it will be

able to fulfill the role, it has no mechanism for attempting to find another robot to perform the role. Mobile robots embedded in a dynamic environment may come to know that they are unlikely to be able to fulfill their current roles or, due to problems localizing be unsure whether they have the capability to perform their roles, yet they have no mechanism to find another robot to perform the role. Secondly, the mechanism of passing on roles implies that a robot will not have another opportunity to accept the role until all robots have rejected it and some period of time has passed. Mobile robots, at times facing significant uncertainty with respect to their capabilities, have a difficult choice when deciding what to do when they realize their ability to perform a role is uncertain. If they decide to give up a role, it may be the case that there is no other robot capable of performing the role and the role is left unfilled. However, if it continues to take responsibility for the role another robot, very likely to be able to fulfill the role will not get the opportunity.

To overcome these limitations for the special case of a team of robots we have developed two extensions to the basic algorithm. First, we have given the robots the option of transferring roles when their estimate of their capability to perform a role changes, either in their estimate of how they will perform the role or their uncertainty about that estimate. Second, we have developed a auxiliary mechanism for transferring roles. If a robot is uncertain, it can negotiate with other robots to find one which would have higher expected utility for performing the role. Such a mechanism is slower and more communication intensive than simply passing on roles, but has the advantage of ensuring that the role will not be left unfilled if the robot cannot find another robot with higher capability or lower uncertainty.

3.1 Handling Changes in Capability Estimation

When a robot accepts a role it does so based on an estimate of whether it can perform that role. Given the dynamic and uncertain environment in which the roles must be performed, these estimates can turn out to be quite poor. For example, in the SDR domain it could be that a CHR estimates that it could collect three CSRs before their batteries fail, but faster than expected battery usage by the CSRs and more obstacles in the path of the CHR may quickly show the initial estimate to be wrong. Rather than waiting until a robot completely fails, we can use the revised capability expectations to trigger a reallocation of responsibilities. Periodically, a robot will reconsider its estimate of whether it can complete its assigned roles. If it estimates that it will not be able to do so, it attempts a reallocation of the role via the role allocation mechanism of the basic algorithm. That is, if the robot no longer believes it is likely to be able to fulfill all its assigned roles, it chooses the one it is least likely to be able to fulfill and passes it to another robot that may be able to fulfill it. The team then passes the role around until either someone is found to fill it or all team members have been asked.

3.2 Handling Changes in Uncertainty

Robots monitor their localization and when their uncertainty associated with their localization exceeds a fixed threshold, the following algorithm is executed to allow the robot to attempt to find a team mate to which to offload the role:

```
Compute(ρ_i)
for all R_k ∈ AS_i do
  if (EU(Self,R_k,ρ_i)<Threshold) and
  (Avail(Teammates) ≠ ∅) then
    A_j ← PickOne(Avail(TeamMates))
    Send(RequestTakeRoleMsg(ρ_i,R_k)) to A_j
    while not ReceivedMsg(Msg) do
      Wait
    end while
    if Msg is AgreeMsg(TakeRole(R_k) then
      GiveUpRole(R_k)
    else
      UpdateAvailableTeamMate()
    end if
  end if
end for
```

When a robot A_j receives a RequestTakeRoleMsg(ρ_i,R_k) it decides if its expected utility for that particular role is higher then the sender robot, and if so accepts the role. Otherwise it will reject the request.

The function $EU : A \times R \rightarrow \Re$ defines the expected utility for an assignment $< A, R >$ with respect to the overall team goal. ρ_i are the particular parameters used in the expected utility calculation (e.g. the probability distribution of the robot position). When the robot proxy believes that the current utility value of the role assignment is under a desired threshold, it decide to ask to one of its mates to take over the role. The PickOne(Avail(Teammates)) is a function that returns one of the available team mate to which ask for the role switching. Depeding on the level of knowledge that the proxies have about its team mates this function could be designed so as to return the best team mate to ask for a switch. As an example if the proxies know at each time the position of its team mates it could choose to ask to the nearest one first. The UpdateAvailableTeamMate() remove from the list of available team mates the last team mate that have refused the TakeRoleRequest message.

In our case the robots are CHR and the role to execute is to pick up a specified CSR and bring it to recharge station. We assume that each CHR CH_i knows the battery level and the position of all the CSR assigned to CH_i. However each Helper Robot does not knows the assignments of other team mates and the information about the other CSR.

The expected utility for an assignment is measured according to the evaluation of parameters computed by the proxies. In our scenario those parameters take into account the level of uncertainty into the robot localization. The basic idea is to measure the weighted distance of a robot from a given CSR using the measure of the likelihood the robot has about its position. More in detail in our case the robots have a sampled distribution function that measure the likelihood for the robot of being in a certain sample point. We measure the weighted distance of the robot H_i from a CSR CS_j in the following way: $Wd(H_i, CS_j) = \int_{s \in S} pdf(s) d(s, CS_i) \delta s$ where $pdf(s)$ is the probability distribution function of the robot H_i and $Bl(CS_j)$ is the battery level of the CS robot CS_j. The expected utility of an assignment is measured as: $EU((H_i, CS_j)) = \frac{1}{Wd(H_i, CS_j) Bl(CS_j)}$, for the assignment (H_i, CS_j) and a role exchange is performed between two assignments (H_i, CS_j) and (H_k, CS_j)if $(EU((H_i, CS_j)) < EU((H_k, CS_j)))$.

4 Experiments and results

We have conducted several experiments in order to evaluate our approach, using a simulation of the distributed robotics domain. The simulator represents the building as a grid and the CHRs are able to move from grid location to grid location, pick up CSRs and re-charge CSRs by moving them to a re-charge station. While the details of robot control are not simulated, the uncertainty CHRs have about their position is modelled using a localization algorithms very similar to those used on real CHRs. In particular, the localization algorithm uses a well known markovian localization method [3], based on simulated landmarks in the building. Battery level in the CSRs decrease with uncertainty, thus it is not possible to predict its dynamic during the task execution.

We used two different kinds of simulation set up. In the first one the experiments are conducted without using the proxies for the role assignment. The role allocation approach is implemented in a software module inside the simulator. In the second setting the proxies have been connected to the simulator and execute the same approach for the role assignment.

While in the first set of experiments we are mainly focused on investigating how different parameter settings for the environment affect the performance of our approach the second experimental setting is used to validate the obtained results using the proxies framework.

In the first setting of experiments we tested four different algorithms: the allocation algorithm described in [11], the extension of this algorithm to handle the changing in capability estimation described in section 3.1 the mechanism for the uncertainty handling described in 3.2, and finally the combination of this two extensions. We decided to vary the amount of CHRs that can have a degradation on their localization capability during the experiments and investigate how this parameter affects the different algorithms performance.

For the first set of experiments we used an environment with 24 CHRs and 47 CSRs and each experiment is 6000 simulation steps long. For each different parameter setting we performed five repetitions. The result obtained are reported in table 2 and in table 3. Table 2 shows the results when five CHR can experience problem in their localization capability while table 3 reports the result with ten. In each table the first column shows the algorithm used, the second column shows the average battery level of CSRs over time. The third column shows the average of the minimum battery level of all the CSRs over time. In both the second and third columns, the averages exclude the battery levels of robots that have failed. The fourth column of the tables, shows the number of CSR that completely failed, i.e., the number of CSRs whose battery level falls to 0. Finally the last column shows the standard deviation computed over the five repetitions.

Algorithm	Avg B. L.	Min B. L.	Fail	Std Dev
Basic	0.644672	0.195902	13.6	2.8
Ovl Handl.	0.65997	0.223343	11.8	0.97
Unc Handl.	0.693892	0.229101	12.2	0.75
Ovl and Unc	0.684224	0.241805	9.8	1.47

Table 2: *Results for five CHR with localization problems*

Algorithm	Avg B. L.	Min B. L.	Fail	Std Dev
Basic	0.633321	0.17654	20.6	2.58
Ovl Handl.	0.65293	0.215206	17.6	1.85
Unc Handl.	0.691214	0.221129	15.8	2.64
Ovl and Unc	0.691896	0.219198	16.2	2.79

Table 3: *Results for ten CHR with localization problems*

Algorithm	Avg B. L.	Min B. L.	Fail	Std Dev
Unc Hnd. 5	0.695983	0.238347	14.6	1.36
Unc Hnd. 10	0.699272	0.237091	17.6	2.87

Table 4: *Results for the limited exchange*

The results show that the overall performance of the team is negatively affected, when more CHRs have their localization capability degraded

When comparing results obtained using the overload handling algorithm with the basic algorithm the number of failed CSRs results to be lower while both the average battery level and the average of the minimum battery level are improved. The improvement is similar both for the

Algorithm	Avg B. L.	Min B. L.	Fail	Std Dev
Unc Handl.	0.699902	0.236674	12.5	1.65

Table 5: *Results for the distributed setting*

case when five and ten CHRs can have a degrading localization capability. Moreover the overload handling algorithm results in a lower standard deviation from the average failure value, showing a better adaption to problematic situations.

Also the algorithm for uncertainty handling seems to improve the performance for the overall team. In particular for the results reported in table 2 we have a very low standard deviation, similar to the overload handling mechanism. However when the number of CHR that can have localization problems is higher we have actually a higher standard deviation but still acceptable results. The results reported in table 2 and 3 for the uncertainty handling algorithm, are obtained assuming that each CHR can ask and have a respond at each simulation step from all its team mates when trying to exchange a role. This is a very strong assumption and it is not likely to be met in the real application. Therefore we performed an experiments limiting the number of team mates that can be query during each time step. In table 4 we report the results for this set of experiments the first row of the table refers to the case where five CHR can have their localization capability degraded, while the second row reports the results for ten. In both the cases the results are worst if compared with the respective row of table 2 and 3. Those experiments show that the discussed approach could be not enough effective for our reference scenario, where the assumption made in the previous experiments could easily not be met.

In the second experimental setting we connected the proxies to the simulator. We decided to test the algorithm for the uncertainty handling, when five CHRs can have a degradation in their localization capability. All the parameters described in the previous set of experiments are used also in this set, except for the number of repetition that in this case is not five but two. Those experiments have been conducted in order to see how the overall performance of the algorithm could be affected using the actual proxy framework. In particular for our scenario a very important issue are the conflicts that can possibly arise among the proxies' information on the actual world state, due to the asinchronicity of the message passing approach. The results reported in table 5 show that the algorithm performance seems not to be heavily affected by this issue, however the small number of experiments conducted does not allow to draw a statistically significant conclusion, and further investigations need to be done.

5 Related Work

One of the first and best known approach for role allocation in a robotic team is represented by the ALLIANCE architecture [9]. Following the ALLIANCE approach each robot in the team during the role execution, considers a possible role reallocation iteratively at fixed time steps. At each time step all the possible roles are taken into account for reassignment. The role to execute is chosen accordingly on the measure of two parameters namely *acquiescence* and *impatience*, that each robot estimates, and that are used in order to compute the utility value of a given robot to execute a given role. The main drawback of this approach is that for a large team of robots executing a complex role, the tuning of parameters that rule the dynamic evolution of *acquiescence* and *impatience* is not trivial. Another very well known approach for the problem of dynamic role assignment is the Broadcast of Local Eligibility (BLE) [15] and the algorithm used in the coordination of the ART team in the RoboCup domain [1]. Both the approaches are based on the broadcast among the robots of a utility value for each role at each iteration steps. Each robot computes the role to be performed by comparing its utility value for each possible role with all of its team mates. For those approaches the main problem when scaling up with the team size is represented by the possible overwhelming of information that have to be sent at each iteration. Moreover for complex and structured roles the tuning of the utility function which is crucial for the effectiveness of the application, could be very complex. Our approach with respect to those works is explicitly designed for a large scale application, thus is characterized by a very low communication an computation requirement. By passing the role responsibility around the robots' proxy for the role assignment we need to communicate only to one of the team mate only the role that the robot is giving up, and each proxy decides to take the role responsibility by only considering the current state of its assigned robot. Thus we avoid a considerable amount of communication and computation that could easily overwhelm the team for a large scale application.

A different approach to the problem of dynamic role assignment for a team of robot is represented by auction-based methods [4, 16]. Those methods are based on the Contract Net Protocol [2]: each time a role is available for the team an auction is set up and each robot "bids" in order to execute the role accordingly to their role-specific utility estimates. The highest bidder wins a contract for the role and proceed to the execution. Among those approach the MURDOCH system does not allows for role re-allocation among the team members, while the work presented in [16] allows for re-allocation only when the role is completed. Thus with respect to this approaches our work allows to reallocate roles to robots in a more flexible fashion. In particular by considering a role re-

allocation when a robot estimate of its capability changes significantly our algorithm allows the team to avert failures before they occur. Moreover none of the previous approaches consider an explicit model of the uncertainty in the robot performance, in the reallocation algorithm. Our work attempt to explicitly deal with the uncertainty in the robot performance by proposing a novel algorithm for role exchange that meets the real time constraint imposed by the application.

6 Conclusions and Future Works

Role allocation is a critical challenge for robot teams in large-scale, real-world domains. This paper focuses on distributed algorithms for role allocation, where no centralized entity exists that can perform such allocations. While distributed algorithms offer significant advantages, the presence of significant uncertainty and dynamic changes in robots' own capabilities pose significant challenges in designing such algorithms. Previous works in multi-robotic and multiagent systems has provided algorithms for role allocation, but these algorithms often fail to explicitly represent and reason with uncertainty in a robot's own capabilities, and often do not address dynamic continuous changes in such capabilities. This paper presents two key contributions to address these limitations in large-scale settings. First, it presents a novel algorithm that explicitly reasons with uncertainty and dynamic changes in robot's own capabilities by exploiting domain level knowledge. Second, it presents the application of a proxy-based architecture (previously applied only in agent-human settings) in large-scale robotic domains demonstrating the application of a reusable infrastructure for multirobotic domains. This paper presents the role allocation algorithm, and an experimental study based on simulations. While the simulation experiments show that our approach seems to be promising, several issues should be addressed more in detail. In particular the approach used to handle the uncertainty of the robot about their capabilities, should be improved in order to be more efficient under a broader range of constraints. Moreover more experiments for the evaluation of the approach within the proxy framework are needed. One application of our work involves a real project involving 100 robots, which must guard a building for over 24 hours. Within this collaborative "SDR" project, role allocation and re-allocation is a significant challenge, and the work reported in this paper has taken a significant first step in addressing this challenge. Other applications of the work reported in this paper are on the horizon as well, and include recently popular work on "robot, agent, person" or RAP teams[11].

References

[1] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Coordination among heterogenous robotic soccer players. In *Proc. of International Conference on Intelligent Robots and Systems (IROS'2000)*, 2000.

[2] R Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[3] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[4] Brian P. Gerkey and Maja J Matarić. Sold!:auction method for multi-robot coordination. In *IEEE Transaction on Robotics and Automation*, volume 18(5), pages 758–768, October 2002.

[5] Barbara Grosz and Sarit Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.

[6] Luke Huhnsberger and Barbara Grosz. A combinatorial auction for collaborative planning. In *Proceedings of the International Conference on MultiAgent Systems*, 2000.

[7] N. Jennings. The archon systems and its applications. Project Report, 1995.

[8] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2003.

[9] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[10] David V. Pynadath and Milind Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Journal of Autonomous Agents and Multi-Agent Systems, Special Issue on Infrastructure and Requirements for Building Research Grade Multi-Agent Systems*, page to appear, 2002.

[11] P. Scerri, D. V. Pynadath, L. Johnson, Rosenbloom P., 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 (to appear).

[12] K. Seow and K. How. Collaborative assignment: A multiagent negotiation approach using bdi concepts. In *Proceedings of AAMAS'02*, pages 256–263, 2002.

[13] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[14] Gil Tidhar. Team-oriented programming: Preliminary report. Technical Report 41, Australian Artificial Intelligence Institute, 1993.

[15] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. In Springer Verlag, editor, *Proc. of Distributed Autonomous Robotic Systems*, volume 4, pages 347–356, 2000.

[16] R. Zlot, A Stenz, B Dias, and S. Thayer. Multi robot exploration controlled by a market economy. In *Proc. of the IEEE Intl. Conference on Robotics and Automation (ICRA)*, pages 3016–3023, Washington DC, May 2002.