A Decentralized Approach to Cooperative Situation Assessment in Multi-Robot Systems

Giuseppe P. Settembre
DIS Department
University "Sapienza" of Rome
settembre@dis.uniroma1.it

Paul Scerri
Robotics Institute
Carnegie Mellon University
pscerri@cs.cmu.edu

Alessandro Farinelli
ECS Department
University of Southampton
af2@ecs.soton.ac.uk

Katia Sycara
Robotics Institute
Carnegie Mellon University
katia@cs.cmu.edu

Daniele Nardi
DIS Department
University "Sapienza" of Rome
nardi@dis.uniroma1.it

ABSTRACT

To act effectively under uncertainty, multi-robot teams need to accurately estimate the state of the environment. Although individual robots, with uncertain sensors, may not be able to accurately determine the current situation, the team as a whole should have the capability to perform *situation assessment*. However, sharing all information with all other team mates is not scalable nor is centralization of all information possible. This paper presents a decentralized approach to cooperative situation assessment that balances use of communication bandwidth with the need for good situation assessment. When a robot believes locally that a particular plan should be executed, it sends a proposal for that plan, to one of its team mates. The robot receiving the plan proposal, can either agree with the plan and forward it on, or it can provide sensor information to suggest that an alternative plan might have higher expected utility. Once sufficient robots agree with the proposal, the plan is initiated. The algorithm successfully balances the value of cooperative sensing against the cost of sharing large volumes of information. Experiments verify the utility of the approach, showing that the algorithm dramatically out-performs individual decision-making and obtains performance similar to a centralized approach.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Coherence and coordination*; I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Experimentation

Keywords

cooperative perception, robotics, situation assessment

1. INTRODUCTION

Emerging, large multi-robot teams hold great promise for revolutionizing the way some important, complex and dangerous tasks, such as monitoring and surveillance[7], space exploration[4] and disaster response [15] are performed. Such teams must choose cooperative courses of action in the face of considerable uncertainty and time pressure. To coordinate their activities in complex unstructured scenarios, robots need the ability to aggregate information into estimates of *features* relevant to their mission and their interactions. The process of acquiring this knowledge is known as *situation assessment*[10]. The more accurately and more quickly the team can determine the situation, the better it can select a team plan to achieve its objectives.

Situations are organized into a hierarchy with the most general situations at the top of the hierarchy and the most specific situations at the leaves. There is a corresponding hierarchy of team plans, i.e., a plan for each class of situation. It is assumed that more specific plans are more effective, but it is better to execute a more general plan than a more specific one for the wrong situation. For example, in the rescue domain the presence of a victim is the most general situation, with an unconscious, badly injured victim and a conscious, uninjured victim being more specific situations. Based on belief in the current situation and a model of the reward for different plans in different situations an individual can compute the expected utility (EU) of each plan. However, based only on local, uncertain sensor readings, individual beliefs will be inaccurate. Even in the

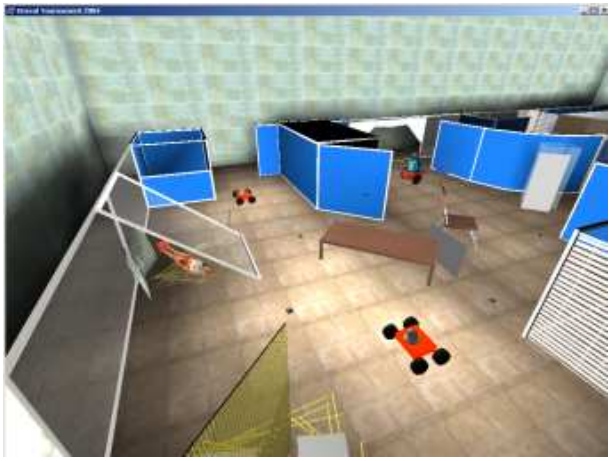

Figure 1: Example of simulated multi-robot rescue response in USARSim.

Cite as: A Decentralized Approach to Cooperative Situation Assessment in Multi-Robot Systems, G. P. Settembre, P. Scerri, A. Farinelli, K. Sycara, and D. Nardi, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

best case, this will lead to the individual deciding that more general plans have higher EU even though the team might have had enough information to select a plan more specific to the situation.

In many multi-robot environments, multiple robots will take sensor readings of the same situation over a relatively short time. Those readings will be from heterogeneous sensors with both systematic and random noise. In the cases of interest to this paper, if all the sensor readings could be provided to an appropriate filter an accurate assessment of the situation could be performed, but individually each robot would have a significant probability of incorrectly characterizing the situation. Unfortunately, communication bandwidth limitations typically preclude sharing all sensor data, especially in large teams. This paper presents an approach to cooperatively, distributedly and efficiently choosing an appropriate course of action in the face of this uncertainty thus improving team performance. The key hypothesis in this paper is that uncertainty must be explicitly dealt with in a distributed and coordinated way for most effective team behavior. Our current focus is robotic search and rescue, where robots need to explore an unknown environment, searching for victims identified via suggestive features (e.g., temperature, movement, shape, etc.). While various approaches have been developed for this problem [11, 20, 21], little attention has been devoted to distributedly and cooperatively dealing with uncertainty. Most previous work has either addressed uncertainty from an individual perspective [2, 3, 19], or has proposed centralized solutions [13, 1, 17].

The approach presented here works as follows. When a robot believes a plan should be initiated, before initiating the plan it creates a message with the proposed plan and sends it randomly to another team member. The robot receiving the message looks at the proposed plan and checks the proposal against its beliefs in the events constituting the situation for the plan. If its beliefs are in line with the proposed plan or it has no sensor readings to dispute the conclusion, it simply forwards the message on. Conversely, if its beliefs do not match those required for that plan initiation, it chooses some of the observations that lead to that belief, attaches them to the message and sends it back to the robot from which it was received. The robot receiving the challenge to its proposal, integrates the attached observations into its beliefs and reassesses the choice of plan. This process continues until a sufficient number of robots agree on the choice of plan. Notice that observations are only shared when beliefs differ enough to change the appropriate plan to be executed, minimizing unnecessary communication while still allowing the team to leverage collective sensing resources to make decisions. The overall process is inspired by work on argumentation [8] adapted to a large team setting.

To evaluate the effectiveness of the proposed approach, experiments were performed in a simulated search and rescue scenario. The approach is shown to perform almost as well as an approach that broadcasts every detected feature to every other robot, while using an order of magnitude less communication bandwidth. However, the performance of the team with the distributed protocol degrades when the environment is very sparse, since fewer robots have information about the same situation, and when the environment is very dynamic since team mates information is often outdated.

2. PROBLEM

This section formally describes the problem addressed by this paper. $R = \{r_1, \ldots, r_m\}$ is the set of mobile robots. $E = \{e_1^t, \ldots, e_n^t\}$ is the set of events which occur in the environment and can be imperfectly observed by robots. Observations are not direct sensor readings (e.g., laser scans) but the result of a feature extraction process indicating the presence of an interesting feature in a given location of the environment. Robots have a model of the noise in their observations and a model of how events evolve over time. By integrating observations over time, robots can have a belief over events $Bel_r(e_i^t)$ which represents the probability that the event e_i is true.

A *situation* represents the fact that a certain combination of events are true or false at the same time in a certain location. For example, a situation s might be defined as $e_i \wedge \neg e_j$. We indicate with $S = \{s_1^t, \ldots, s_m^t\}$ the set of situation instances, each one referring to a specific location. Situation instances are grouped in classes, where $S_{CL} = \{S_\perp, S_0, \ldots, S_k\}$ is the set of *situation classes*. A class of situations is a grouping of situations into semantically equivalent groups. For example, the situation class S might be defined as $e_i \wedge (e_j \vee \neg e_j)$, if e_j is irrelevant to any action that might be taken in that class of situation. For example, the situation of an unconscious victim would be defined by events *victim present*, *localized*,*35 < heat source < 42*, *C02 present* and *no movement*. Notice that if events are independent, then the belief in a situation is simply the product of the belief of the constituent events. However, it is more typically the case that events are not independent and that techniques such as Bayesian networks are required to compute the probability of a situation, given the probabilities of events. For example, if the team has a high belief that a victim is present in a location x, then it is more likely that heat will be detected in x, as compared to other locations. Multiple robots can sense the same events, taking observations that allow them to form belief distributions over events and, subsequently situations.

Each member of the team is provided with the same set of plans $\mathcal{P}$, containing a specified plan P_i for each possible situation class S_i. The plan $P_\perp$ (associated to situation class $S_\perp$) represents the default situation where the team will not take any particular action. For example, $S_\perp$ can represent a situation where there is no victim present. Plans are organized in a hierarchy. At the top level of the hierarchy there are the most general situations, while at the leaves are the most specific situations (those specified in the most detail).

A function $U : \mathcal{P} \times S_{CL} \rightarrow \mathcal{R}$ specifies the utility (reward or cost) for the team when executing the plan for a situation class. Since the situation class for each location is unique and independent of other locations, we can focus on one location only. Therefore, for ease of notation, we remove, when possible, the location subscript from situation instances and represent the situation instance simply as s. The function U must meet the following constraints:

- $s \in S_i$ and $s \in S_j$ and $S_i \subseteq S_j \rightarrow U(P_i, S_i) \geq U(P_j, S_i)$
- $s \notin S_i$ and $s \in S_j$ and $S_i \subseteq S_j \rightarrow U(P_i, S_j) < U(P_j, S_j)$
- $U(P_\perp, S_\perp) = 0$

The first constraint requires that higher reward will be received when more specific plans are executed in appropriate situations. The second constraint says that it is better to execute more general plans than inappropriate but more specific plans. The final constraint simply requires that there is no reward or cost for not acting when the situation does not require action.

Given the above definitions, the problem is for the team to choose the plan that maximizes their expected utility, given their belief in the current situation. Typically better information will allow more specifically tailored plans to be executed and when there is higher uncertainty more general, less effective plans will be used. Formally the team must maximize for each situation instance s:

$$\max_k \sum_t U(P_k^t, S_i^t) \quad (1)$$

where i is the most specific situation class s belongs to, and x is the estimated class for the situation by the team, at time t. Optimal performance will be obtained if $x = i$ for each time step and each situation instance.

More formally, to perform the maximization specified in Equation 1, robots should execute at each time step t, for each situation instance s, the plan P_k^* such that:

$$k^* = \arg\max_k \left(\sum_{S_x \in S_{CL}} Bel(S_x) * U(P_k, S_x) \right). \quad (2)$$

Since each situation class is related to a set of events, we can translate the condition on plan execution that derives from Equation 2 to a set of constraints of the form $m_{ij} < Bel(e_j) < M_{ij}$ for each e_j that relates to situation class S_i. This computation can be done off-line. In other words, for ranges of beliefs, the appropriate plan to execute can be determined without computing the EU of each plan. During the mission execution, robots will monitor their belief value over the possible events and instantiate the corresponding plan (i.e., the plan that meets the pre-computed constraints).

2.1 Example

For example, consider the simple situation where there are only two events e_1, representing presence of a human shape, and e_2, representing human like movement. $S_1 = victim$ is defined by $e_1 \wedge (e_2 \vee \neg e_2)$ and $S_2 = unconscious\_victim$ is defined by $e_1 \wedge \neg e_2$. Suppose the team has a plan P_1 for S_1 where the plan involves sending in a robot to try to lead the victim to safety, sending human rescuers only if this fails. A plan P_2 involves immediately sending in human rescuers and is best suited for S_2 where the victim is unconscious. Suppose the rewards are $U(P_1, S_1) = 2$, $U(P_1, S_2) = -1$, $U(P_2, S_1) = -2$ and $U(P_2, S_2) = 4$. Notice that, following the constraints specified in 2, for the team it is better to execute the more specific plan in the more specific situation (i.e. $U(P_2, S_2) \geq U(P_1, S_1)$) and it is better to act more general than wrong (i.e. $U(P_2, S_1) \leq U(P_1, S_1)$).

Figure 2 illustrates the situation and shows the robots (represented as circles) in the environment, attempting to understand the situation. Notice that multiple robots may have sensor readings that can help the plan selection, but, after communication, only the filled robot has enough information to decide that a plan is required and initiate the process.

In this case, following Equation 2 the team should execute P_1 when $Bel(S_1)*2+Bel(S_2)*(-1) > Bel(S_1)*(-2)+Bel(S_2)*4$ Assuming that e_1 and e_2 are independent, $Bel(S_1) = Bel(e_1)$ and $Bel(S_2) = Bel(e_1) \times (1 - Bel(e_2))$. Thus, the decision above can be broken down into the team should execute P_1 when $Bel(e2) > 1/5$.

3. DISTRIBUTED ALGORITHM

The aim of the distributed algorithm is to cooperatively choose the plan that maximizes the team utility, in the face of sensor uncertainty, without overloading communications networks. The basic approach is as follows: each robot individually performs a decision theoretic computation and decides to instantiate the plan that maximizes the expected reward. When a robot decides to instantiate a plan, it sends a plan proposal to a team mate. When receiving a plan proposal, the robot checks whether its beliefs agree with the plan selection, based on its own utility calculation. If the robot agrees or has no observations that disagree, it passes the plan on to another team member. The proposal is passed on until a fixed number of robots agree with the choice of plan. If the robot's beliefs support the choice of a different plan, it sends back a *challenge*, attaching to the message observations that caused its beliefs to differ in such a way that it would choose the different plan. In this respect, the approach is inspired by argumentation-based negotiation, where agents reach an agreement iteratively proposing possible alternatives and provide arguments in favour of their proposals.

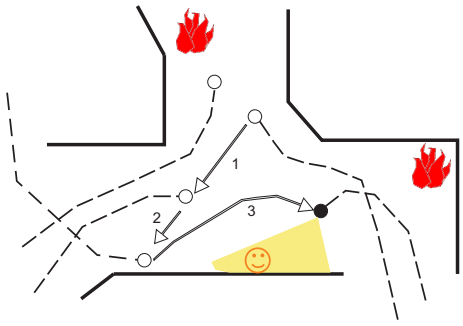

Figure 2: An example of cooperative perception in a simple search and rescue scenario.

The key to the efficiency of the proposed approach is that since the whole team shares the same taxonomy of situations and plans, then arguments are simply observations about the event that causes the disagreement. Such events can be easily identified as the events that do not meet the constraints deriving from Equation 2, for the proposed plan. In other words, they are those beliefs which fall in ranges that mean a different plan has a higher EU. Only these observations are shared, while many irrelevant observations are kept private, thus minimizing the information exchange. In many domains, the resulting message use is low enough to be practical. In the remainder of this section, the algorithm is described in detail.

When a robot makes an observation it integrates the new observation to update its beliefs about events. The belief update is performed using a standard Bayesian framework. Given the updated belief over events, the robot assesses, via Equation 2, whether executing some plan is appropriate. If the robot decides to instantiate a plan P_x it will create a proposal message for that plan. The message will have the following structure:
Msg=<*plan*,*status*,*TTL*,♯ *agree*,*observations*>
where
plan = ID of the proposed plan
status = {"PROPOSAL"|"CHALLENGE"}
TTL (time-to-live of the proposal) is the number of agents that must agree to the plan before it is initiated,
♯*agree* is the number of agents that have agreed so far
observations = $\{\langle e_i, obsList \rangle\}$ is a list of observations for e_i.
Unless there is an ongoing challenge, $obsList = \emptyset$.

Notice that since only contradicting observations are sent and only to the robot that disagrees, this message format scales with all key environmental and team variables.

When a robot receives a message it executes the procedure *On-MsgReceived* specified in Algorithm 1. The robot first checks whether the constraints related to the event beliefs for P_x are satisfied (line 2) that is, it checks whether its beliefs are in ranges that would make P_x the utility maximizing plan. If the constraints are satisfied, it will just forward the message randomly to another robot, in-

Algorithm 1: Algorithm executed by each robot

```
ONMSGRECEIVED(msg)
(1)   INTEGRATEBELIEFS(msg.obs)
(2)   planAgree ← EVALARGUMENTS(msg.plan)
(3)   if msg.status == PROPOSAL
(4)     if planAgree
(5)       msg.♯agree ← msg.♯agree + 1
(6)       if ♯ msg.agree < TTL
(7)         SEND(msg,nextAgent())
(8)       else
(9)         INSTANTIATEPLAN(msg.plan)
(10)    else
(11)      msg.status ← CHALLENGE
(12)      msg.obs ←RETRIEVEREFUTINGOBS(msg.plan)
(13)      SEND(msg,msg.sender)
(14)  else
(15)    /* msg.status == CHALLENGE */
(16)    if planAgree
(17)      msg.status ← PROPOSAL
(18)      msg.obs ←RETRIEVESUPPORTINGOBS(msg.plan)
(19)      SEND(msg,origMsg.nextAgent())
(20)    else
(21)      if origMsg.prevAgent() ≠ null
(22)        msg.obs ←RETRIEVEREFUTINGOBS(msg.plan)
(23)        SEND(msg,origMsg.prevAgent())
(24)      else
(25)        DESTROY(msg)
```

creasing ♯*agree* (lines 4- 7). If the constraints are violated, $status$ is changed to "CHALLENGE". Observations relevant to the event that caused the violation are inserted into $obsList$. For example, if the robot would have chosen P_j instead of the proposed P_i because $Bel(e_i) > 0.4$ then it sends observations that led to that belief. It then sends the message back to the robot it received it from (lines 11-13). The function *retrieveRefutingObservations* gets those observations from the agent history.

The robot receiving the challenge, integrates the observations in the message into its own beliefs and reconsiders the choice of plan. Due the integration, two possibilities exist: (i) the additional observations did not sufficiently changes its local beliefs to cause it to believe a different plan has higher EU; (ii) it now believes another plan has higher EU. In case (i) the robot clears the $obsList$ changes the $status$ back to "PROPOSAL" and forwards it randomly (lines 21- 25). The function *retrieveSupportingObservation* retrieves the list of events whose observations are needed by the challenger to be persuaded by the current plan. In case (ii), the robot attaches any additional observations to the message and sends it back to where it received it (16- 19). If it was the robot that initiated the proposal, the plan is changed and new message passing begins (line 25). A team plan is instantiated when the number of agreeing agents with a proposed plan reaches TTL ($\sharp agree == TTL$) (line 9).

Notice, that this approach potentially allows conflicting plans to be initiated by different robots in the same team. Solving such conflicts is outside of the aim of this work, but has been addressed in literature [16]. Notice, that if message passing was occurring for conflicting plans, the robots may be able to notice this and merge the conclusions of both sequences for more efficient performance.

3.1 Execution example

To illustrate execution of the algorithm, we present a simplified execution sequence for a single plan. There are 5 agents, 2 events and the plan needs 3 agents to agree before being instantiated. For simplicity, we show only observations about the situation of the example. Figure 3 shows the messages exchanged among agents over time. The observations each robot has about each event, are specified under the robot identifier. We indicate positive, T, and negative, F, observations, together with the event and the time step they refer to. Positive observations support belief in the event, negative ones undermine it. For example, T_1^5 is a positive observation, related to e_1 observed at time step 5. The example begins at time 5, when the robot Ag_1 has five observations indicating that a plan P_1 should be instantiated. The plan is for the situation class defined as $e_1 \wedge e_2$. The robot creates a new message, and sends it, without observations as a proposal to robot Ag_4. The robot Ag_4 has different observations to Ag_1, but it agrees with the overall conclusion about the plan, so it simply updates the number of agreeing robots, $\sharp agree$, in the message and passes it on to Ag_3. The robot Ag_3 disagrees with the plan P_1, because its observations say $\neg e_1$. Hence, it challenges e_1.

In order to do this, it changes the status of the message to "CHALLENGE" and attaches its own observations about event e_1. It then sends the message back to the sender, Ag_4. When Ag_4 receives the message, it updates his own list of observations adding the ones contained in the message; then re-evaluates whether it still finds P_1 to have maximum EU. In this case, Ag_3's observations did not change Ag_4's calculation that P_1 was appropriate. Ag_4 attaches its observations to the message and send it to Ag_3. Ag_3 receives the message again, this time with Ag_4's observations. With Ag_4's observations, Ag_4 now also agrees with the choice of P_1. It increments ♯ agree and moves the message on. Finally, the plan is initiated, when Ag_5 also agrees with the plan.

3.2 Discussion

In this section, some basic termination and performance properties of the algorithm are shown. First, we define the concept of "hard" challenge.

DEFINITION 1. *During the execution of the protocol, given a list of agreeing agents $[a_{k_1}, .., a_{k_n}]$ a challenge is "hard" if, none of the robots among $[a_{k_2}, .., a_{k_n}]$ is able to resolve the challenge for at least one of the events that are challenged.*

After a challenge is resolved, the two agents (the challenger and the solver) have the same observations and agree with the proposed plan. If a challenge is "hard", the message has to be passed back through all the robot chain, until it reaches the proposal initiator. Then, if the first robot still believes that the plan maximizes EU, the message will have to be passed forward through all the chain again to the challenging robot. If a challenge is not "hard", the agreement will be reached with some robot in the chain, saving the number of messages used.

We can now show that the algorithm always terminates.

THEOREM 1. *If TTL is fixed and no further observations are obtained during execution, Algorithm 1 always terminates.*

PROOF. The proof is based on the fact that there is an upper bound on the number of messages that are required by the algorithm to find an agreement among TTL agents, if TTL is a fixed number. In the worst case, the plan will be actually instantiated, but only after that each of the TTL agents starts a challenge message on at least one event; in particular, in the worst case, each new challenge has to be "hard" (so that the message comes back always to the initiator, and it is the only one that is able to solve the challenge).

In this case, to solve the challenge of the first robot receiving the message, needs one message back and one message forward; to solve the challenge of the second agent, needs two messages back and two messages forward, and so on. Therefore, in general, the

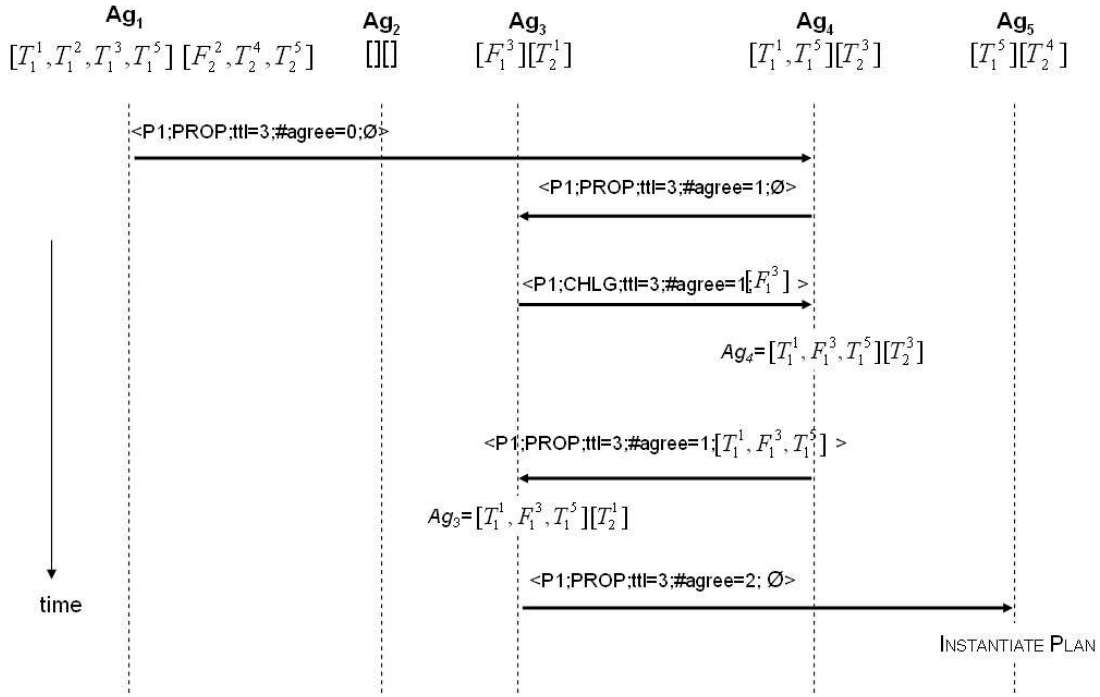

Figure 3: Example of an execution of the protocol

upper bound is:

$$S(TTL) = 1 + 2 + 4 + 6 + ... + 2 \times TTL = 1 + 2 \times \sum_{i=1}^{TTL} i =$$

$$1 + TTL \times (TTL + 1) \quad \square$$

The analysis of the worst case shows that the protocol requires in at most a number of messages polynomial in the TTL. Moreover, experiments in Section 4 will show that the average number of messages needed by the protocol is much lower than the one provided by this theorem.

4. EXPERIMENTS AND RESULTS

The approach was evaluated in an abstract simulation environment. The simulator abstracts the low level details of robots capabilities and focuses on coordination issues, thus allowing to efficiently run experiments with large number of robots (from 70 up to 120 members), under varying environmental conditions (e.g., world dynamism, world size, etc.).

To evaluate the agents' performance, we compute the reward that robots gain over time (according to Eq.1). In particular, we compute, at each time step, the ratio of obtained reward u^* to the highest possible reward u_{max} (the reward that would be obtained always executing the highest reward plan for each situation). Such measure will be named $prew = \frac{u^*}{u_{max}}$ (*percentage of reward*).

The communication overhead is evaluated using two measures: i) number of messages exchanged at each time step by each robot; ii) size of the messages (in bytes) exchanged at each time step by each robot. We count a broadcast message as point to point message times the number of robots. While for a more precise analysis of the overhead one should consider the specific network used , this provides a general cost model for communication which is suitable for our level of analysis.

The proposed approach (referred as *MAS_Policy* in the following) was compared to two different strategies. The first one, *Centralized*, requires each robot share all its observations with all other robots at each time step. Clearly, this type of approach is infeasible for large teams, but it provides an upper bound on the performance that can be achieved by the agents. Notice that the *Centralized* approach is not guaranteed to obtained the maximum reward. In the *Centralized* approach, the team activity is based on the perceptions of all the robots, therefore team performance is related to the available perceptions: e.g. if the density of the robots is very low (Figure 4) or the perception is very bad (Figure 6), the centralized approach will not make optimal decisions.

The second benchmark strategy is *Selfish_agent*, where the first agent that has enough information to initiate a plan, will just initiate it. The results of this policy provide a bound on the performance that can be achieved using a non-cooperative perception approach. The general performance of this approach illustrate the difficulty of the problem faced by robots.

Experiments have been performed in a 2D office-like environment. The simulated robots have limited knowledge of the overall team state and can communicate with only a subset of the overall team. In each experiment there were 70 simulated homogeneous robots, each with the same perception model. The perception model is based on a decreasing probability of correct detection with distance, i.e. robots are more likely to obtain correct observations when closer to the features. The initial distribution of robots in the environment is random. Each graph reports values averaged over 10 trials of the same experiment. Each experiments is simulation over a finite horizon of 100 time steps. When not explicitly stated, the TTL is set to $1/3$ of the team size, providing a balance between communication overhead and performance.

In all the experiments, the reward function was designed to assign to situations a reward (and a cost) that is proportional to their depth in the hierarchy. Therefore, in a hierarchy of situation classes with depth d, each situation class at depth i will receive a reward r if instantiated for the correct situation, and a cost c is instantiated for a wrong situation. The reward and cost are specified by the following equations: $r = k_1 \times (i+1) \times 1/d \; c = -k_2 \times (i+1) \times 1/d$. For example, leaves will have reward $r = k_1$ and cost $c = -k_2$, the nodes that are direct sons of the root will receive reward $r = k_1 \times 1/d$ and cost $c = -k_2 \times 1/d$ etc... Using this model for the reward function allows us to test our approach with different hierarchical structures, (i.e., varying the depth of the hierarchy) while meeting the constraints specified in Section 2. In particular, since we want to study the performance of the approach when specified situations might be chosen, we set the weights such that ($k_2 = 1/2 \times k_1$). In this way, partially specified situations will be frequently be the best choice for the team.

To exchange information about features present in the environment, robots need to share a common reference framework. To simplify the experimental setting, we do not explicitly consider localization errors. As a matter of fact, standard localization techniques [3] can be used for our experimental scenario, and localization errors can be taken into account in the error model of the feature extraction process.

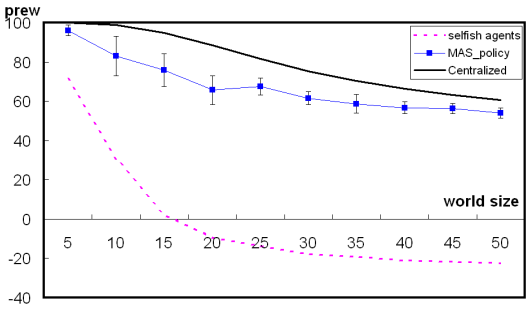

Figure 4: Performance comparison varying world size

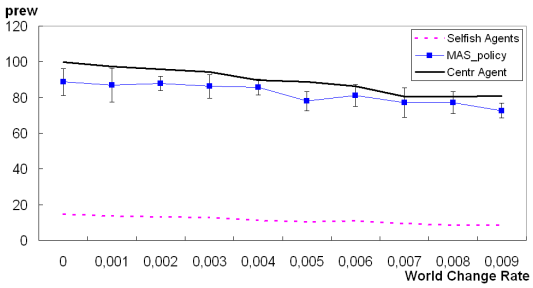

Figure 5: Performance comparison varying world dynamics

We first evaluate the performance of the approach varying key parameters of the environment, namely the size of the world in which robots operate and the dynamism of the world. Varying the world size and keeping the number of robots constant, we test how the approach behaves when the robots have less mutual observation of the same features (see Figure 4). Clearly, the performance of all the three compared policies degrade as the world size increases, however, the MAS_Policy is able to provide performance which

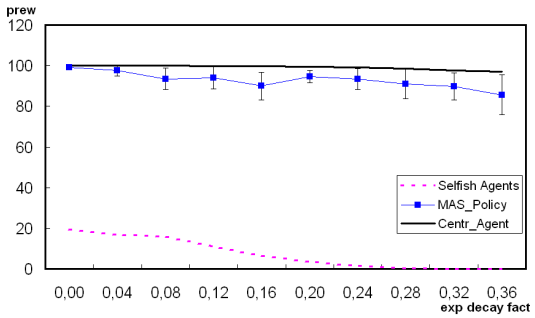

Figure 6: Performance comparison varying quality of perceptions

is very close to the one accrued by the centralized policy, while the single agent policy performs very badly. Varying the world dynamism is intended to test whether our approach is able to react to unexpected changes in the environment. In particular, the world dynamism in this experiment determines how frequently features appear and disappear from the environment. A world change rate of x means that at each time step, each feature has a probability of x to switch its state (i.e., appear in a given part of the environment if it was not present or disappear if it was present). Results reported in Figure 5, show that the approach is able to cope very well with dynamism of the world.

Next, we evaluate how the algorithm behaves when quality of the perception that robots obtain from the environment varies. As mentioned the detection probability is dependent on the distance from the observed feature. The law is a decreasing exponential and the parameter of the exponential is the decay factor that we vary in this experiments. Results reported in Figure 6 shows what happens when the decay rate is raised. The approach to situation assessment is able to provide good results even with very noisy perception.

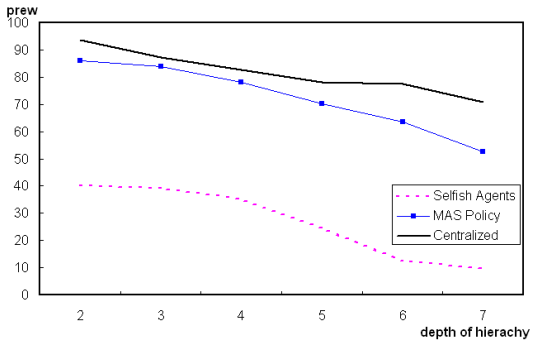

Figure 7: Performance comparison varying hierarchy depth

Next, we look at performance as the depth of the situation hierarchy is raised. Notice that increasing the depth of the hierarchy increases also the number of events that have to be considered to assess a situation. In fact, in the proposed model, the number of events that compose a completely specified situation is equal to the depth of the hierarchy. For example, for a hierarchy of depth 4, a completely specified situation is $e_0 \wedge e_1 \wedge \neg e_2 \wedge e_4$. Moreover, when the situation hierarchy is deeper situations will be more difficult to distinguish. In fact, when the situations are similar for more events, more specific observations are necessary to reach an

agreement among agents. Results reported in Figure 7 indicate that our policy scales well with the hierarchy depth. In fact, the approach has very similar performance to centralized decision maker. Conversely, the performance of the selfish agent policy is heavily affected by the increased complexity of the scenario.

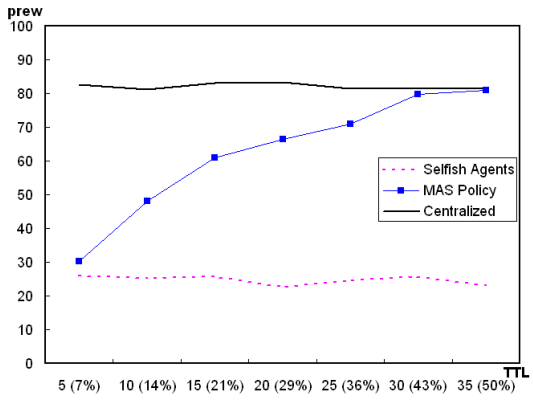

Figure 8: Performance comparison for different TTL

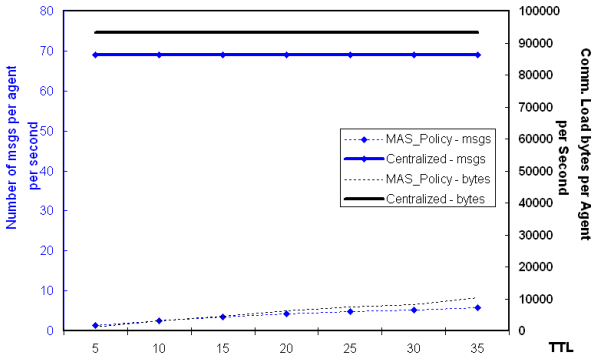

Figure 9: Communication comparison for different TTL (number of messages)

As previously mentioned, a key aspect of MAS_Policy is to minimize the amount of information exchanged among agents. To evaluate this we measured the amount and size of messages exchanged, as TTL was varied. The TTL is the key parameter that influences the amount of communication transmitted among robots, since more agents are required to agree on a plan and more probable is the occurrence of a challenge.

Figure 8 and 9 report the results of our method varying the TTL. In particular, Figure 8 show how the $prew$ measure changes with increasing TTL values. We varied the TTL between 5 ($1/14$ of the total number of agents) and 35 ($1/2$ of the total number of agents). When TTL is very low, the results become very similar to the selfish agent policy, because robots share very few observations, and thus make wrong plan instantiation. High TTL values provide results similar to the centralized strategy.

Figure 9 reports on the left y axis, the number of messages per agents per time step, and on the right y axis the communication load per agent per time steps (bytes). Each sensor reading was modeled as having 100 bytes. Results show that the proposed approach, not only requires a lower number of messages, but ensures also a smaller communication overhead in terms of message size. In particular, for this team size (70 robots), the communication gain is approximatively one order of magnitude over the centralized approach.

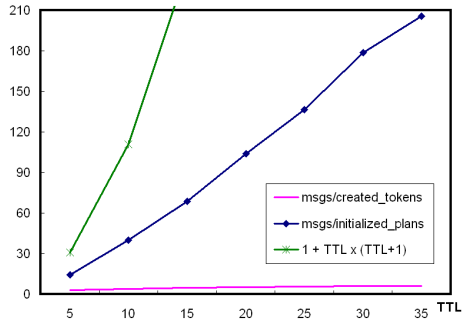

Figure 10: Number of messages required for each execution of the protocol

Finally, Figure 10 reports the total number of messages required for each execution of our algorithm, varying the TTL. In particular, we report the number of messages exchanged divided by the number of instantiated plans and the number of messages exchanged divided by the number of created plans. The first measures the average number of messages required for plans that are instantiated while the second measures the average number of messages exchanged by every execution of the algorithm. In Section 3.2 we claimed that the number of messages required by the protocol in the worst case is quadratic with respect to TTL. Figure 10, shows that this worst case scenario is very unlikely to happen in practice. Results indicate that the average number of messages required for a generic execution of the protocol is in fact less than TTL itself.

5. RELATED WORK

Coordination in multi-robot systems has been successfully addressed using frameworks based on Belief Desire Intention architecture and Joint Intention theory [18, 6]. In particular, the STEAM framework is based on the concept of Team Oriented Plan [18], which are activities that need to be jointly carried out by the agent team. Team Oriented Plans are decomposed into specific sub-activities called roles that individual robots can perform. Our concept of plan is related to the concept of Team Oriented Plans and our hierarchy of situations resemble the one used in STEAM. However, with respect to the STEAM architecture, our approach is specifically focused to address the impact that noisy perception have on the coordination process. Noisy perception results in misaligned, and possible conflicting agents' knowledge, and is likely to be cause of poor system performance. Cooperative perception techniques can be used to address this problem.

Several approaches use cooperative perception to deal with perception limitation of the single robot [13, 1, 17]. The general idea is to exchange sensor readings and aggregate them using different filtering techniques (e.g. Kalman filters [1, 17] or particle filters [13]). These approaches attempt to reduce the uncertainty before deciding how to act, by exploiting passive noise filtering techniques. Other techniques, explicitly deal with the uncertainty when choosing a course of actions, for example COM-MTDPs [12]. However, such approaches often require to exchange large amounts of data among robots. A key reason for this is that, typically, each robot attempts to maintain an accurate model of the complete state when, in practice, only a small part of the overall state might be relevant to its

activities. Some works exist which explore this possibility [14], but current results are still limited to small number of agents.

Recently, increasing attention has been devoted to the concept of situation assessment. However, the concept has been mainly investigated in centralized settings [10]. While several approaches are now able to integrate information at data level[1] among different robots [9], limited attention has been devoted to the problem of situation assessment in a distributed setting.

Finally, our approach is inspired by ideas taken from argumentation-based negotiation [8]. Specifically, as in negotiation, our approach is based on a sequence of one-to- one interaction. Differently then argumentation, however, agents are not self interested. Robots are willing to be totally cooperative with their team mates. In particular, our approach uses ideas from argumentation as a mean to restrict the amount of communication needed and thus avoiding sending irrelevant information when possible. Moreover, robots support their plan proposal and plan challenges, using arguments, which in our case are observations.

6. CONCLUSIONS

This paper represents an important first step towards a distributed approach to situation assessment in uncertain environments; moreover, to the best of our knowledge this is the first attempt to define a distributed approach that cooperatively deals with the uncertainty of team plan initiation. The approach explicitly and cooperatively addresses the uncertainty that robots have due to noisy observations and gains its efficiently by ensuring only useful observations are shared.

We presented an extensive evaluation of the algorithm across a wide set of interesting operational conditions. In particular, we compared the approach to a centralized and individual approaches. The approach presented in this paper performed almost as well as the centralized approach while using an order of magnitude less communication. It far out-performed the individual approach.

Future work will look at a range of issues to make the approach more relevant and more efficient for real robot teams. An immediate point of interest is whether TTL can be dynamically adjusted to account for the amount of agreement or disagreement between agents. Another area of interest is whether plan deconfliction algorithms can be combined with this algorithm, potentially simplifying overall coordination and improving efficiency in one step.

7. ACKNOWLEDGMENTS

This research was supported in part by AFOSR FA9550-07-1-0039, by L3-Communications grant number 4500257512 and by "Iniziativa Software CINI-Finmeccanica".

8. REFERENCES

[1] M. Dietl, J.-S. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS'01)*, Maui, Hawaii, 2001.

[2] H. Durrant-Whyte, D. Rye, and E. Nebot. Localisation of automatic guided vehicles. In *Robotics Research: The 7th International Symposium (ISRR'95)*, pages 613–625. Springer Verlag, 1996.

[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] D. Goldberg, V. Cicirello, M. B. Dias, et al. A distributed layered architecture for mobile robot coordination: Application to space exploration. In *Proc. of 3rd Int. NASA Workshop on Planning and Scheduling for Space*, 2002.

[5] D. Hall and J. Llinas, editors. *Handbook of Multisensor Data Fusion*. CRC Press, 2001.

[6] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. In *Arificial Intelligence*, 1995.

[7] K. Konolige, D. Fox, C. Ortiz, et al. Centibots: Very large scale distributed robotic teams. In *Proc. of the Int. Symp. on Experimental Robotics (ISER04)*, Singapore, 2004.

[8] S. Kraus, K. Sycara, and Evanchik. Argumentation in negotiation: A formal model and implementation. *Artificial Intelligence, (104)1-2, pp. 1-69,*, September 1998.

[9] A. Makarenko and H. Durrant-Whyte. Decentralized data fusion and control algorithms in active sensor networks. In *In The 7th Int. Conf. on Information Fusion (Fusion'04)*, pages 479–486, 2004.

[10] C. J. Matheus, M. M. Kokar, and K. Baclawski. A core ontology for situation awareness. In *Proceedings of the Sixth International Conference on Information Fusion*, 2003.

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

[12] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[13] M. Rosencrantz, G. Gordon, and S. Thrun. Decentralized sensor fusion with distributed particle filters. In *Proc. Conf. Uncertainty in Artificial Intelligence (UAI-03)*, Acapulco, Mexico, 2003.

[14] M. Roth, R. Simmons, and M. Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proc. of 4th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, The Netherlands, July 2005.

[15] R. Saha and K. Chang. An efficient algorithm for multisensor track fusion. *Aerospace and Electronic Systems, IEEE Transactions on Volume: 34 Issue: 1*, pages 200 – 210, 1998.

[16] P. Scerri, Y. Xu, E. Liao, G. Lai, and K. Sycara. Scaling teamwork to very large teams. In *Proc. of 3rd Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, New York,USA, July 2004.

[17] A. Stroupe, M. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proc. of Int. Conf. on Robotics and Automation (ICRA2001)*, volume 2, pages 1092–1098, 2001.

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

[19] G. Theocharous, K. Rohanimanesh, and S. Mahadevan. Learning hierarchical partially observable markov decision processes for robot navigation. In *IEEE Conf. on Robotics and Automation , (ICRA)*, Seoul, South Korea, 2001.

[20] 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)*, pages 347–356, Knoxville (TN), USA, October 2000.

[21] R. Zlot, A. Stenz, M. B. Dias, and S. Thayer. Multi robot exploration controlled by a market economy. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA'02)*, pages 3016–3023, 2002.

[1] see [5] for the definition of the different data fusion levels