Dynamically Formed Heterogeneous Robot Teams Performing Tightly-Coordinated Tasks

E. G. Jones, T. K. Harris, B. Browning, M. B. Dias, B. Argall, A. Rudnicky, M. Veloso, A. Stentz
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: {egjones,tkharris,brettb,mbdias,bargall,air,mmv,axs}@cs.cmu.edu

Abstract

As we progress towards a world where robots play an integral role in society, a critical problem that remains to be solved is the Pickup Team challenge; that is, dynamically formed heterogeneous robot teams executing coordinated tasks where little information is known a-priori about the tasks, the robots, and the environments in which they will operate. Successful solutions to forming pickup teams will enable researchers to experiment with larger numbers of robots and enable industry to efficiently and cost-effectively integrate new robot technology with existing legacy teams. In this paper, we define the challenge of pickup teams and propose the treasure hunt domain for evaluating the performance of pickup teams. Additionally, we describe a basic implementation of a pickup team that can search and discover treasure in a previously unknown environment. We build on prior approaches in market-based task allocation and Plays for synchronized task execution, to allocate roles amongst robots in the pickup team, and to execute synchronized team actions to accomplish the treasure hunt task.

Introduction

The vision that drives this work is that teams of heterogeneous robots will dynamically solve complex tasks by efficiently joining their complementary capabilities. The research challenges in realizing this vision include robust operation across multiple environments, building capabilities applicable across multiple robot types, and building teams of robots that improve over time.

Competitions, such as RoboCup, have been effective in focusing efforts to overcome some of these challenges (Noda *et al.* 1998). However, these competitions focus on part of the overall problem and do not generally address teams formed in an ad-hoc manner, complex environments beyond a well-defined soccer field, and the complexities of heterogeneous teams. The Pickup Team challenge is to dynamically form teams of robots (and possibly humans) given very little a priori information. That is, team members may have only minimal prior knowledge of each others behavior, the tasks at hand, and the environments they operate in, but are able to combine effectively.

There are several reasons why an increased understanding of pickup teams is needed. First, it is impractical to develop large teams or teams of expensive robots at the same site, at the same time. This currently hinders multi-robot research. Successful pickup teams will facilitate further research by allowing separate researchers to easily pool their robots to create teams for further study. Second, robots may be needed for emergency tasks where there may be insufficient time to hand-engineer the coordination mechanisms before task execution. Pickup teams enable robot teams to be formed on very short-notice for such tasks. Third, as robots fail, get lost, or otherwise malfunction, it is often necessary to substitute or add new robots. Successful pickup teams will allow the integration of new robots into existing teams, and also enable teams of heterogeneous robots to perform efficiently under dynamic and uncertain conditions. Thus, the overall research challenge is to provide a principled methodology for creating pickup teams. This paper presents a first approach to address this challenge.

The reported work focuses on dynamically forming teams of heterogeneous robots to perform tasks that require tight-coordination. The robots have limited individual capabilities, can sense different information about their environment, and can be assigned abstract tasks for execution. Robots can solve primitive tasks in different ways depending on the robot capabilities and prevailing environmental conditions. The implemented approach is demonstrated in a treasure hunt scenario. Thus, the contribution of this paper is threefold: defining the pickup team challenge, introducing the treasure hunt domain, and implementing a basic pickup team.

The Treasure Hunt Domain

To investigate the conduct of pickup teams, we require a domain which will allow for dynamic and heterogeneous team formation, encourage coordination and tight coupling between team members, and provide a metric against which to compare team performances. Our treasure hunt is a domain which provides for each of these characteristics.

The treasure hunt domain consists of robot teams competing in and exploring an unknown space. The teams are heterogeneous, where the particular capabilities necessary to accomplish hunt tasks are distributed throughout the robot team, often being unique to a single member. The hunt tasks

are executed towards the goal of acquiring specific objects, the treasure, within an unknown environment. Thus a representation of the world must be built as it is explored, and the treasure must be identified and then localized within the built representation. Team coordination follows as a direct necessity, as the abilities required to perform each of these tasks are distributed throughout the team members.

The ultimate goal with respect pickup team formation is speed and plasticity. Within this domain, not only are teams created quickly and on the fly, but each member has no prior knowledge about the abilities of its potential teammates; a robot knows of its own capabilities only. Inherent to the definition of a hunt task are the abilities necessary to accomplish it. Communication between potential pick-up team members is therefore carried out at the time of team formation, to ensure the satisfaction of all capabilities requirements.

This domain presents an adversarial environment in which to execute the hunt tasks. Teams currently compete against the clock, with the intent of collecting as much treasure as possible within the allotted time. Eventually the teams will compete against other dynamically formed heterogeneous pickup teams. That the environment is adversarial provides a metric for team performance, for the success of a pickup team may be measured in the amount of treasure it collected. Additionally, an adversarial environment encourages enhanced team performance through efficiency, and thus also tightly-coupled team coordination.

In our specific treasure hunt implementation, potential team members include Pioneer robots and the robotic Segway RMP platform provided by Segway, LLC 1. The Pioneer robot is equipped with a SICK laser and gyroscope, and is therefore able to both construct a map of an unknown environment, and localize itself upon that map. The Segway robot has been outfitted with two cameras, by which it is able to visually identify both the Pioneer robot and the treasure. The presented task is to search for and retrieve treasure. To explore an area while searching for the treasure, the Pioneer is able to navigate and build a map, while the Segway is able to follow the Pioneer and search for treasure. Upon treasure identification, the robots must return home with the treasure. By localizing on its created map, the Pioneer is able to determine the home location; the Segway then follows the Pioneer as it proceeds home. Team coordination between the robots during execution is accomplished jointly via the visual identification of the Pioneer by the Segway, and by communication between the two robots should this visual link be lost (in which case the Pioneer is commanded to pause, until seen by the Segway). Communication additionally occurs when the Segway informs the Pioneer that treasure has been found.

The treasure hunt domain satisfies the criterion set for the study of the performance of pickup teams. It offers a number of challenging aspects, including robust and efficient operation in unconstrained environments, and ad-hoc team formation. Efficient execution requires a coordinated search of the space and the maintenance of an accurate shared knowledge about the space. As such, this domain provides a rich environment in which to push the boundaries of adaptive, autonomous robotics.

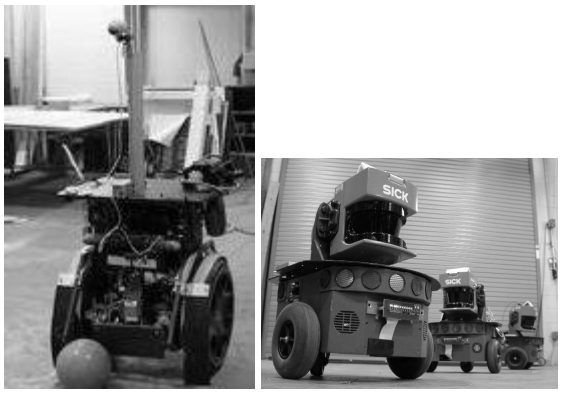

Figure 1: The left figure shows a Segway robot, while the right figure shows the pioneer robots.

Component Technologies

In this section we review our current approaches to teamwork – Skills, Tactics, and Plays (STP) for team coordination in adversarial environments, and TraderBots for efficient and robust role assignment in multi-robot tasks.

STP: Skills, Tactics, and Plays

Veloso *et al.* (Bowling, Browning, & Veloso 2004) introduce a skills, tactics, and plays architecture (STP) for controlling autonomous robot teams in adversarial environments. In STP, teamwork, individual behavior, and low-level control are decomposed into three separate modules. Relevant to our implementation are Plays which provide the mechanism for adaptive team coordination. Plays are the central mechanism for coordinating team actions. Each play consists of the following components: *(a)* a set of roles for each team member executing the play, *(b)* a sequence of actions for each role to perform, *(c)* an applicability evaluation function, *(d)* a termination evaluation function, *(e)* a weight to determine the likelihood of selecting the play.

Each play is a fixed team plan that describes a sequence of actions for each role in the team towards achieving the team goal(s). Each of the roles is assigned to a unique team member during execution. The role assignment is based on the believed state of the world and is dynamic (e.g. role A may start with player 1, but may switch to player 3 as execution progresses). Note that the role assignment mechanism is independent of the play framework.

The concept of plays was created for domains where tight synchronization of actions between team members is required. Therefore, the sequence of tactics to be performed by each role is executed in lock step with each other role in the play. Hence, the play forms a fixed team plan whereby the sequence of activities is synchronized between team members.

As not all plans are appropriate under all circumstances, each play has a boolean evaluation function that determines the applicability of the play. This function is defined on the team's belief state, and determines if the play can be executed or not. Thus, it is possible to define special purpose

plays that are applicable only under specific conditions as well as general-purpose plays that can be executed under much broader conditions. Once executed, there are two conditions under which the play can terminate. The first is that the team finishes executing the team plan. Each play includes an evaluation function that determines whether the play should be terminated. As with applicability, this evaluation function operates over the team's belief state. Hence, the second means of ending a play is if the termination evaluation function determines that the play should end, either because it has failed or is successful.

Team strategy consists of a set of plays, called a playbook, of which the team can execute only one play at any instant of time. A play can only be selected for execution if it is applicable. From the set of applicable plays, one is selected at random with a likelihood that is tied to the play's weight. The plays are selected with a likelihood determined by a Gibbs distribution from the weights over the set of applicable plays. This means the team strategy is in effect stochastic. This is desirable in adversarial domains to prevent the team strategy being predictable, and therefore exploitable by the opponent.

TraderBots

TraderBots, developed by Dias and Stentz (Dias 2004) is a coordination mechanism, inspired by the contract net protocol by Smith (Smith 1980), is designed to inherit the efficacy and flexibility of a market economy, and to exploit these benefits to enable robust and efficient multirobot coordination in dynamic environments. A brief overview of the TraderBots approach is presented here to provide context for the reported experimental results and analysis.

Consider a team of robots assembled to perform a particular set of tasks. Consider further, that each robot in the team is modelled as a self-interested agent, and the team of robots as an economy. The goal of the team is to complete the tasks successfully while minimizing overall costs. Each robot aims to maximize its individual profit; however, since all revenue is derived from satisfying team objectives, the robots self-interest equates to doing global good. Moreover, all robots can increase their profit by eliminating excess cost. Thus, to solve the task-allocation problem, the robots run task auctions, and bid on tasks in other robots task auctions.

If the global cost is determined by the summation of individual robot costs, each deal made by a robot will result in global cost reduction. Note that robots will only make profitable deals. Furthermore, the individual aim to maximize profit (rather than to minimize cost) allows added flexibility in the approach to prioritize tasks that are of high cost and high priority over tasks that incur low cost but provide lower value to the overall mission. The competitive element of the robots bidding for different tasks enables the system to decipher the competing local information of each robot, while the currency exchange provides grounding for the competing local costs in terms of the global value of the tasks being performed.

Teamtalk

Our vision of integrally social robots, dynamically formed and dynamically tasked, is the vision that both humans and robots are able to dynamically form teams, and that in these teams, humans are able to guide the robots toward goals by inserting tasks into the trading framework. The trading framework may then assign play roles and provide information to both the robot and the human team members, taking efficient advantage of the skills of both people and robots of various capabilities. To this end we have developed the human interface component, Teamtalk (Harris *et al.* 2004). Teamtalk is a multi–modal multi–agent dialog system, and it is a client of the *OpTrader* server. Each human team member can carry a tablet PC with them which:

- runs an instance of the Teamtalk software
- provides for pen–and–tablet–based IO with the robots, and
- provides for speech–based IO with an attached headset

Architecture

Each human team member (each with an instance of Teamtalk) is a Teamtalk client of the *OpTrader*, and each Teamtalk instance spawns a Ravenclaw dialog manager for each active robot. Thus, $n \times m$ stateful dialogs are held, where n is the number of human team members (and Teamtalk instances) and m is the number of robotic team members. Each Teamtalk instance contains a single automatic speech recognition engine, parser, natural language generation module, text–to–speech engine, and pen–and–tablet interface. For each robot that comes on–line, the interface spawns a separate dialog management system, and confidence annotation module.

Information Flow

A typical exchange that demonstrates the internal dynamics of the Teamtalk systems follows: A human team member utters a phrase, which in turn is decoded by Sphinx into one or more hypotheses along with associated confidence scores. The hypotheses are routed to Phoenix, which parses each hypothsized utterance, and for each parse, generates the concepts and additional confidence annotation.

These concepts are routed to m instances of Helios, one for each robot. Helios examines the acoustic and language model confidences from Sphinx, additional confidence annotations imparted by Phoenix's parser coverage of the hypotheses, and the expectation agenda generated by Ravenclaw based on that robot's dialog state. From all of this evidence, Helios picks concepts to pass to the dialog manager, and assigns a final combined confidence value for each unique concept that is passed along.

At this point each robot's dialog manager has received a set of concepts with associated confidences. Each dialog manager decides from the dialog state and the concepts whether it is the addressed robot in the dialog or it is simply eavesdropping on a conversation between the human and another dialog partner. This allows for the possibility of opportunistic eavesdropping, so that one currently not addressed robot could, for example, develop a better context

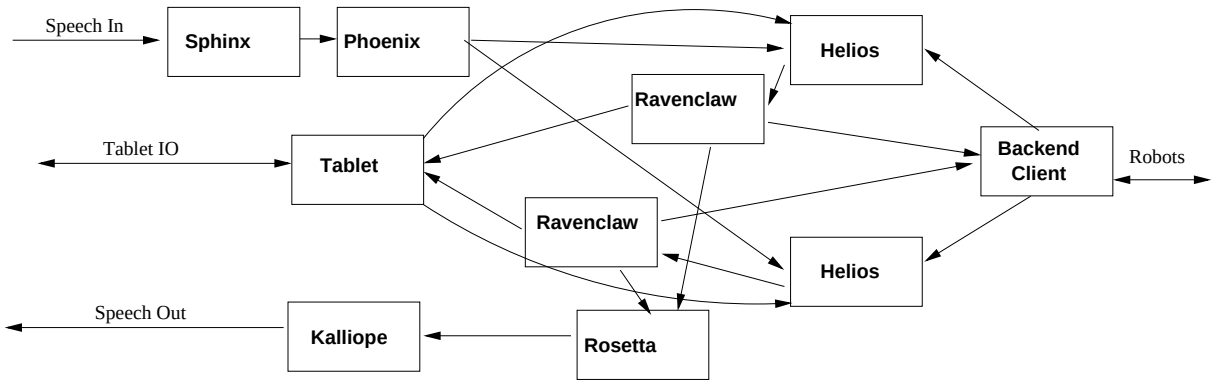

Figure 2: Teamtalk system architecture

from which to understand the next utterance that may actually be addressed to it. The dialog manager may act on the concepts, or it may engage in a grounding action on one or more of the concepts. This decision is made based on a policy that depends on the concept confidence history in the dialog, and the action to be performed (more on this below). The exact form of the grounding action, if made, ultimately depends on the Rosetta configuration, but may be something like, "did you say you wanted to go backwards". The dialog is a mixed–initiative dialog, so that if certain concepts are provided that only partially complete some agenda, the dialog system may take initiative with, for example, "how far do you want me to go backwards?".

The dialog manager, based on grounded concepts, sends messages to the back–end manager, which in turn inserts tasks to an *OpTrader* server. Task completion messages, as well as all other messages originated by the robots, are captured by the back–end manager, and those messages are routed to each robot's Helios just like the human utterances were. Helios generally assigns high confidence to concepts that are originated by the robots, since those messages do not incur the noisy speech channel.

The robot–originated messages, having been annotated by Helios, are routed to Ravenclaw. Ravenclaw treats most robot–originated messages as requests by the robots to speak, and generally passes those messages directly to Rosetta. It may also complete task agencies, taking these off of the focus stack and changing the expectation agenda as appropriate. These messages are also routed to the pen–and–tablet system, which in the case of a completed task, may change the color of the robot's icon or take some other suggestive actions.

Rosetta uses template–based natural language generation to render concept terms into strings of words. The responses generated by Rosetta are routed to Kalliope. Kalliope is configured to use Cepstral's Swift Synthesis engine, and it chooses a Swift voice for each robot. That way, each robot can speak in its own voice.

Challenges

The Teamtalk system addresses several challenges in multi–modal and multi–agent spoken dialog. Some of these challenges are currently addressed at a superficial level, as we believe that all of the elements must be in place to some degree in order to develop valid tests of the system–wide consequences of the other design decisions. Some challenges that we are addressing are outlined below.

How should a system manage multi–agent dialogs? We have chosen to implement a stateful dialog manager for each human–robot pair, and to have those dialog managers be capable of spawning upon the interfaces' contact with the robot. We have also decided that all messages pass through all dialog managers, regardless of the apparent intended recipient. It is up to each dialog manager to determine if it is being addressed or not, and what those consequences are. This choice allows us to handle more complicated addressing mechanisms (see below), allows the dialog manager's policies to be easily customized to the particular robot (also see below), and allows for the possibility of beneficial eavesdropping.

How should addressing mechanisms be interpreted by the robots? There are five manners in which a person can address a team. These are: *(i)* as individuals; *(ii)* as a whole team; *(iii)* as a team leader; *(iv)* to nobody in particular (talking into the air); *(v)* via a proxy.

Our system handles the first two manners, and could be modified to handle the third manner. The fourth and fifth manner seemed unusual in practice.

How should robot voices be handled in a multi–robot system? We have chosen to assign a voice for every robot, until the number of available voices is exhausted. At that point there is no choice but to recycle voices. Clarifying salutations are copious in our NLG (e.g. "clyde here, *message*"), so even when the voices themselves fail to distinguish the robot, the content of the message probably will. In our current system, voices are discrete and finite, and cannot be parameterized. A future version of the system may allow for parameterized voices so that the number of distinct voices will be limited only by the limit of the number of distinctions a person can make.

How should speech recognition be handled in a multi–robot system? We have decided to incorporate a single ASR per human team member. This decision is partly made

on computational grounds. ASR overshadows all of the other systems' computational requirements. Having a single ASR per human prevents the explosion of computational requirements where the number of robots grows, while at the same time allowing the ASR system to be tailored to the acoustic models that best fit that particular speaker.

How should one manage asynchronous messages generated by an agent back–end? We have chosen to route those messages through the same path that parsed human messages would have been routed, namely, through the confidence annotator and the dialog manager. Both the confidence annotator and the dialog manager are aware (through tags attached to the messages) of the origin of the message and may adjust their responses appropriately.

How should one handle robots entering and leaving the sphere of interaction? Our back–end manager, upon detecting a yet unseen robot, spawns the necessary components to fulfill that robot's dialog needs. This procedure fulfills the requirement that robots be capable of forming teams in an ad–hoc fashion.

How should one ground concepts? We have chosen to ground concepts based on the ASR confidence, the grammar coverage, the concept history, the task, and per–robot policies. In this way, we are able to conservatively ground concepts for high risk tasks, and yet liberally and efficiently execute low risk tasks. For example, in our field work, our Pioneer robot with its laser–based collision detection will translate on command, only asking for command confirmation when there is a very noisy speech signal; conversely, our Segway SMP, with its high momentum, poor odometry, and total lack of collision detection, almost always asks for translation confirmation.

How can one effectively use a pen–and–tablet interface with a multi–agent speech interface? Much like the manner that we broadcast all utterances to all dialog managers, human and robot originated alike, the tablet display is treated as a shared resource. The human team member may draw on the tablet at will, which sends a corresponding message to all of the dialog managers. Any of the robot dialog managers may send messages to the tablet display, which can in turn be interpreted by the other dialog managers.

Implementation

We have attempted to address two main challenges in our implementation. The first is the dynamic formation of pickup teams efficiently given heterogeneous robots. We have adapted the Traderbots system to perform this function. Tasks are matched with plays consisting of a number of roles; the roles of the play, when performed, should satisfy the requirements of the task. Each role in a play contains a sequence of action primitives that can actually be executed by the robots - but a role can also require a certain set of capabilities. Robots only bid on roles that they have the capability to perform, thus accommodating the heterogeneity of the robots and providing an efficient way for new kinds of robots with different sets of capabilities to represent themselves to the system. Traderbots also requires that robots have the ability to estimate the cost of actions - for instance, a cost may be the total distance that a role requires a robot to move. By minimizing cost in performing role allocations we hope to not only get feasible solutions but also to have high efficiency.

The heterogeneity of the pickup teams demands that much care be taken during play execution - roles may depend on each other, and all robots need to do their part to actually discover, localize, and retrieve treasure. Thus once the allocation has been performed, the tight coordination subsystem must monitor and direct play execution.

The following describes our implementation of the subsystems for dynamic pickup team allocation and tight coordination. The first part introduces the main components of implementation, and the second part of the section illustrates system performance by describing the life cycle of a treasure hunt task as it moves through allocation to execution.

Implementation Components

To realize the full functionality of our system a number of different processes must be run on each of the different robots, as well as on a human operator workstation. The following will describe only those processes essential to the function of the system.

The top layer of our implementation consists of a number of Traders. Each agent, including the human operator, is assigned a Trader. A Trader is the agent's interface to the market. The Trader can introduce items to be auctioned to other Traders by sending a call for bids, can respond to calls for bids with bids, can determine which bids are most beneficial for the auctioning agent, and can issue awards to bidders that have won auctions. In our system the human operator has a trader known as an OpTrader. Each robot agent has a trader called a RoboTrader.

The PlayManager forms the next component module. The primary role of the PlayManager is to select useful plays for a task and to coordinate the execution of activities in a play across a small sub-team to perform a won task. The PlayManager can select plays to be bid on for a specified task, coordinate the execution of a play with the play managers for the other assigned roles, execute the sequence of tasks for its particular robot and synchronize the activities, where required, between the different roles.

A final important component - the Robot Server - provides an interface between the PlayManager and the components on the robot responsible for controlling the robot. A strength of our system is that neither the PlayManagers nor the Traders need to know very much about how the control of the system is actually implemented - the Robot Server serves as the single point of contact. Thus the Robot Servers on the robots must understand a standard packet, cause PlayManager commanded actions to be performed, and to send action statuses, but beyond that robot platform developers are free to develop their system in any fashion they wish.

The Treasure Hunt Task Life Cycle

A task is issued and enters the Trading system Initially, the human operator designates a SearchArea task, embedding the points of a bounding polygon in the task data. The

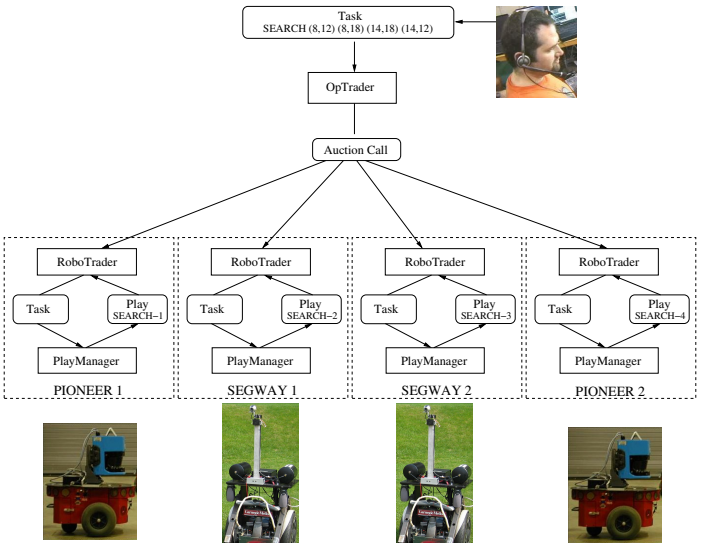

Figure 3: Parts (a) and (b) of the task life-cycle as discussed in the Implementation section.

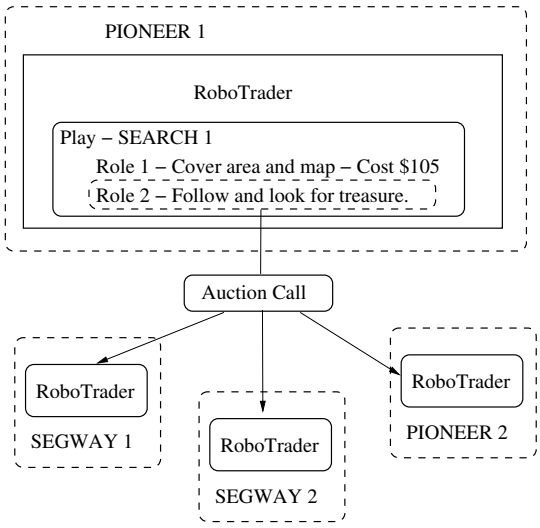

Figure 4: The RoboTrader, upon receiving a play from the PlayManager, selects a play for itself and produces a cost estimate. It auctions the other role in the play to the other robots.

task is sent to the OpTrader. The OpTrader creates an auction call with the task data, serializes it, and sends it via the wireless network using UDP to all RoboTraders (see Figure 3).

Each RoboTrader receives the task auction call and gets a matching play from the PlayManager The RoboTraders receive the call for bids from the OpTrader. They deserialize the call and pass the task specification to their individual PlayManagers over a UDP socket connection. Each contacted PlayManager will compare the task string against the applicability conditions for each play in its playbook. It will then select a play stochastically amongst the set of plays that are applicable, and return this play to the RoboTrader.

The RoboTrader assesses the play The RoboTrader now has a play matching the task and a set of roles. It then must select one of the roles for itself. For each role in the play, the RoboTrader first considers whether or not it possesses all the capabilities required to perform that role. For all roles for which it is capable the Trader then performs a cost evaluation.

In the treasure hunt we are largely concerned with minimizing the time it takes to accomplish the task - as our robots move roughly the same speeds during play execution, we try to minimize distance travelled as an approximation of time. If robots were heterogeneous with respect to speed this should be reflected in their costing function. For most roles cost is computed as total metric path cost for goal points to be visited in the role. Costing in our system is modular, so additional or different costing functions can be added easily as required.

Once a cost has been assigned to each role, the RoboTrader selects the lowest cost role for itself. It then prepares an auction call with all the remaining roles. This auction call is serialized and sent to the other Traders, as shown on the left in Figure 4.

The other RoboTraders bid on the role auction After the other RoboTraders receive and deserialize the role call, they determine their own cost for each role in the call. For any role in the call that requires capabilities the Trader does not possess it assigns an infinite cost. For all roles that the Trader can perform, it assigns the cost determined by the costing function. Note that the Trader must bid on all roles for which it is capable. The role bids are then returned to auctioneering RoboTrader (shown in Figure 5).

The RoboTrader bids on the task Once all bids are received or a timeout has expired the auctioneer RoboTrader then determines the winner or winners of the role auction. All role bids are considered, and the lowest cost bid is selected. If that bid is non-infinite, the role is designated as assigned to the bidding Trader. As a Trader can only win a single role in a play, that Trader's other bids are nullified. Additionally, all bids for that role from any other Trader are nullified. Then the lowest cost remaining bid is considered; this continues until all roles in the play are assigned or there are no remaining bids.

If all roles from the play have been assigned, the RoboTrader constructs a bid for the original task, with a cost that is summed over all assigned role cost estimates. The bid is sent to the OpTrader. This is shown in Figure 6.

Task and role awards are granted Once the OpTrader has received bids from all RoboTraders or a call timeout has expired it awards the task to RoboTrader with the lowest cost bid. An award message is sent to the winning RoboTrader.

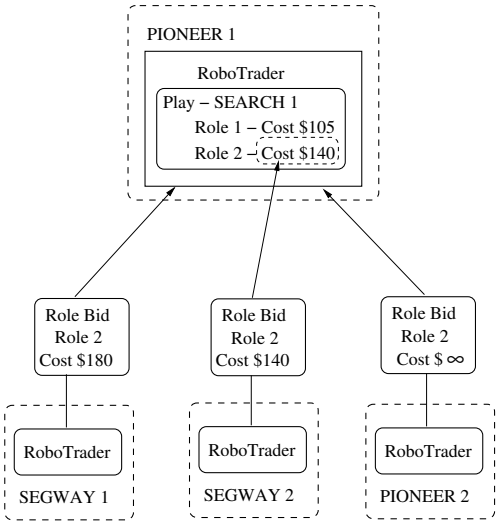

Figure 5: Receiving the bids for the remaining role, the OpTrader selects the lowest cost bid.

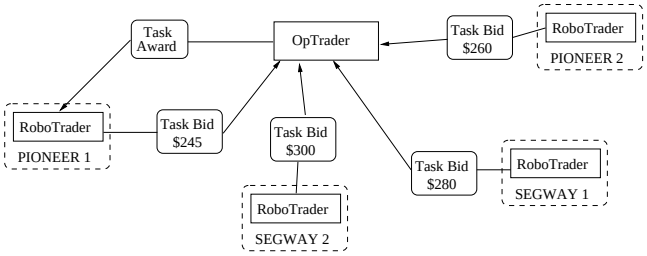

Figure 6: The bids from all plays are returned to the OpTrader, who selects the lowest cost task bid and sends a task award to the winning bidder.

All other RoboTraders are informed they lost the auction. The winning RoboTrader then sends role award messages to all RoboTraders that were assigned roles in the winning play. Note that at this point the RoboTrader still has not accepted the task award.

Awards are accepted and play execution begins The initial role bids made by the RoboTraders were non-binding. However, a role award is binding; once accepted, a role must be performed. If a RoboTrader has received no other role award in the interval between bidding and the arrival of this award, it accepts the role award. Otherwise it rejects the role award. If any of the role awards are rejected, or one of the Traders awarded a role does not respond to the award by a timeout, then the task award is rejected and the OpTrader must perform another auction.

If all role awards are accepted, then the RoboTrader sends a task acceptance message to the RoboTrader. Then it sends an execution message to the PlayManager detailing the final role assignments. This stage is shown in Figure 7. The RoboTrader's role in the pickup team allocation is concluded except for relaying status information to the OpTrader when

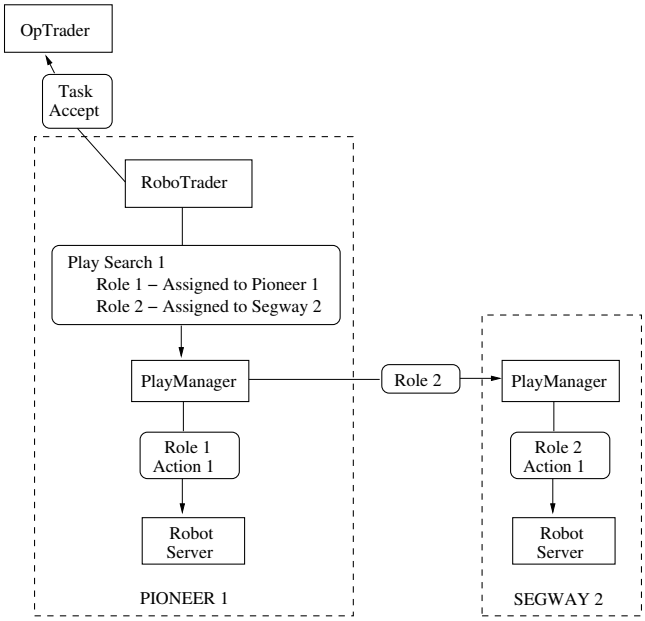

Figure 7: After the RoboTrader has confirmed the availability of the winning Role bidder, it can send notice of task acceptance to the OpTrader and forward the role assignments to the PlayManager. The PlayManager contacts the play managers of all robots assigned a role and sends them a description of the actions in their role. Action primitives are sent to the robot servers and the play execution begins.

the play completes. Future work, however, will examine dynamic re-assignment of roles if and when required.

The PlayManager begins play execution Once informed of the play to execute, and the list of assigned roles, the PlayManager forms the subteam to execute the play. It does this by contacting each of the subteam members RoleExecutors and transmits the play in compressed XML format to them. The RoleExecutor is responsible for executing the assigned role in the play. If any subteam members are essential and fail to acknowledge the play reception, or are not able to be contacted, the play is terminated and reported as such to the RoboTrader. Otherwise, execution proceeds by the PlayManager informing each RoleExecutor to start operation.

At this point, each RoleExecutor becomes loosely coupled to the PlayManager. The RoleExecutor will execute its sequence of actions and will only inform the PlayManager of termination (success or failure), or when it needs to synchronize with another role according to the play. To synchronize the RoleExecutor contacts the appropriate teammate's RoleExecutor and informs the PlayManager for status keeping purposes.

When each RoleExecutor reaches the end of its sequence of actions to perform, it informs the PlayManager of successful termination, and when the team is complete the PlayManager reports this to the RoboTrader. Alternatively, if a robot fails, or the time limit for execution is reached (as encoded in the play itself), the play is terminated and reported

as such. Taken together, execution is distributed, and loosely coupled.

Related Work

Within the field of robotics, there has been considerable research into multi-robot coordination for a variety of domains and tasks (Mataric 1994; Balch & Arkin 1998; Gerkey & Mataric 2003). Many groups have focused on research questions relevant to robot teams particularly (Balch & (eds.) 2001). RoboCup robot soccer has offered a standardized domain in which to explore multi-robot teamwork in dynamic, adversarial tasks (Noda *et al.* 1998; Nardi *et al.* 2004) (see also *http://www.robocup.org*). Segway Soccer (Browning *et al.* 2005) is a new league within this RoboCup domain, which addresses the coordination of heterogeneous team members specifically. The emphasis of these human-robot teams is equality, both physically, as both ride the Segway mobility platform, and with respect to decision making power and responsibility.

How to effectively coordinate heterogeneous teams has been an ongoing challenge in multi-robot research (Scerri *et al.* 2004; Kaminka & Frenkel 2005). However, no one has focused explicitly on the principles underlying the building of such highly dynamic teams when the a-priori interaction between individual robot developers is so minimal. Much of the existing research implicitly assumes that the robot team is built by a group of people working closely together over an extended period of time. While some previous research within the software agents community has addressed the coordination of simulated agents built by different groups (Pynadath & Tambe 2003), none has chosen to address this pickup challenge for the coordination of multiple robots. We believe this research direction of forming dynamic teams will greatly advance the science of multi-robot systems.

Teamtalk is a modification of the emerging Olympus framework (CMU), which is based on the MIT/MITRE Galaxy Communicator (MIT-MITRE) reference architecture for spoken dialog systems.

Conclusions

In this paper, we presented the concept of pickup teams, where teams are formed dynamically from heterogeneous robots with no a-priori experience of one another. We have presented an appropriate domain for exploring the research issues related to pickup teams – multi-robot treasure hunts. Based on our prior work with synchronized activities using STP with plays and tactics combined with robust multi-robot role assignment using the TraderBots market-based architecture, we have proposed a new technique to address the problem of pickup teams.

References

Balch, T., and Arkin, R. 1998. Behavior-based formation control for multiagent robot teams. *IEEE Transactions on Robotics and Automation*.

Balch, T., and (eds.), L. P., eds. 2001. *Robot Teams: From Diversity to Polymorphism*. AK Peters.

Bowling, M.; Browning, B.; and Veloso, M. 2004. Plays as effective multiagent plans enabling opponent-adaptive play selection. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS'04)*.

Browning, B.; Searock, J.; Rybski, P. E.; and Veloso, M. 2005. Turning segways into soccer robots. *Industrial Robot* 32(2):149–156.

CMU. `http://garlic.speech.cs.cmu.edu/rcwiki/index.php/Main_Page`.

Dias, M. B. 2004. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Gerkey, B. P., and Mataric, M. J. 2003. Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *Proceedings of ICRA'03, the 2003 IEEE International Conference on Robotics and Automation*.

Harris, T. K.; Banerjee, S.; Rudnicky, A.; Sison, J.; Bodine, K.; and Black, A. 2004. A research platform for multi-agent dialogue dynamics. In *Proc. of the IEEE International Workshop on Robotics and Human Interactive Communication*.

Kaminka, G., and Frenkel, I. 2005. Flexible teamwork in behavior-based robots. In *In Proceedings of the National Conference on Artificial Intelligence (AAAI-2005)*.

Mataric, M. J. 1994. *Interaction and Intelligent Behavior*. Ph.D. Dissertation, EECS, MIT, Boston, MA. Available as technical report AITR-1495.

MIT-MITRE. `http://communicator.sourceforge.net/`.

Nardi, D.; Riedmiller, M.; Sammut, C.; and Santos-Victor, J., eds. 2004. *RoboCup 2004: Robot Soccer World Cup VIII (LNCS/LNAI)*. Berlin: Springer-Verlag Press.

Noda, I.; Suzuki, S.; Matsubara, H.; Asada, M.; and Kitano, H. 1998. RoboCup-97: The first robot world cup soccer games and conferences. *AI Magazine* 19(3):49–59.

Pynadath, D. V., and Tambe, M. 2003. An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems* 7(1-2):71–100.

Scerri, P.; Xu, Y.; Liao, E.; Lai, J.; and Sycara, K. 2004. Scaling teamwork to very large teams. In *AAMAS'04*.

Smith, R. G. 1980. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers* C-29(12):1104–1113.