Self-Organization and Reorganization of Multi-AUV Systems: CoDA Project Overview[1]

Roy M. Turner
Elise H. Turner
Maine Software Agents and Artificial Intelligence Laboratory
Department of Computer Science
5752 Neville Hall
University of Maine
Orono, ME 04469

Abstract

The CoDA (Cooperative Distributed AOSN control) project at the University of Maine has as its goal the creation of intelligent control mechanisms for multi-AUV systems. Our target domain is the control of autonomous oceanographic sampling networks (AOSNs) [Curtin *et al.*, 1993]. We draw on and extend techniques from distributed artificial intelligence, particularly cooperative distributed problem solving, and multiagent systems. A primary focus of the work is developing protocols and mechanisms to allow AOSNs and other multi-AUV systems to self-organize to fit their mission and situation, then reorganize as necessary when there are changes.

Over the past several years, we have developed a two-level, protocol-based approach to AOSN control that provides both flexibility and efficiency. A *meta-level organization* (MLO) self-organizes from among the more intelligent agents present, then designs a *task-level organization* to fit the current situation and system capabilities. As necessitated by changes to the system and situation, the TLO can either adapt to changes or initiate a new MLO to design a new TLO to fit the changed situation.

This paper discusses the CoDA project. So far, we have concentrated on the development of the cooperation protocols for agents to use during the system's operation, a task-assignment mechanism to assign agents to mission tasks, and organization design techniques for multi-AUV systems. We discuss our approach, the project's current status, and plans for future work.

INTRODUCTION

Multiagent systems (MAS) are collections of autonomous or semi-autonomous agents that (typically) cooperate to carry out some set of tasks. For this paper, *agent* is used in its traditional artificial intelligence (AI) sense: an entity that perceives its environment, embodies a (rational) decision process, and takes action. Examples of agents are software robots (softbots), avatars, and other such software entities; humans; and, most germane to our purpose here, mobile robots such as autonomous underwater vehicles (AUVs) and other instrument platforms.

There is growing interest in and research on multiagent AUV systems, as evidenced by this workshop. There are many applications for multi-AUV systems, including uses in in ocean science, pollution monitoring, global change monitoring, industry, and the military.

Most current multi-AUV systems are capable of relatively simple missions involving only a few, usually homogeneous, AUVs. We are interested, on the other hand, in future AUV systems that:

- are able to perform their missions autonomously or semi-autonomously;
- are able to self-organize to fit their mission and situation;
- can be composed of many heterogeneous AUVs and instrument platforms;
- can be deployed for long-duration missions;

[1]We are deeply grateful to the Office of Naval Research for the generous support of this work through grants N0001-14-96-1-5009 and N0001-14-98-1-0648. The content does not necessarily reflect the position or the policy of the U.S. government, and no official endorsement should be inferred. The authors can be reached via e-mail at `rmt@umcs.maine.edu`. For further information, see `MaineSAIL.umcs.maine.edu`.

- are open—that is, agents can come and go as needed (e.g., as they fail or are needed elsewhere); and
- are highly flexible—capable of taking on different missions during their deployment and able to cope with changing situations, possibly by reorganizing.

Such systems could be deployed for long-duration ocean science data-gathering and monitoring, for surveillance of hostile territory, for detection of terrorist activity in a harbor, or even for aquaculture or other industrial purposes. For example, autonomous oceanographic sampling networks (AOSNs), first proposed by Curtin *et al.* [1993], are just such multi-AUV systems.

There are three major aspects of MAS: intelligent control of the individual agents, interagent communication, and cooperation mechanisms. Each of these is the subject of research projects at MaineSAIL (Maine Software Agents and Artificial Intelligence Laboratory). The Orca Project [e.g., Turner, 1995] focuses on intelligent mission control for AUVs, and other projects look at interagent communication. The subject of this paper, the CoDA Project, focuses on cooperation.

CoDA (Cooperative Distributed AOSN control) [e.g., Turner & Turner, 1998; Turner & Turner, 2001], has as its goal developing intelligent control mechanisms for multi-AUV systems, in particular AOSNs. The roots of CoDA extend back to the MAUV Project [Albus, 1988] at the Marine Systems Engineering Laboratory (MSEL),[2] which fielded two EAVE-III AUVs [Blidberg & Chappell, 1986], and the MAVIS Project [Turner *et al.*, 1991], which worked on techniques for cooperative photography by two AUVs. CoDA focuses on developing the organization mechanism and cooperation protocols to enable intelligent control of large, long-duration multiagent systems, including AOSNs.

In this paper, we describe the CoDA project. We discuss the project in general, then the major pieces that focus on cooperation protocols, task-assignment mechanisms, organization design and redesign, and the CoDA simulation testbed. We then discuss future work.

CoDA Overview: A Two-Level Organization Scheme

We are faced in our task domain with the problem of needing an organization that is both flexible and efficient. Flexibility is important initially, since the composition of the system may not be known a priori. For example, in a rescue or plane crash scenario, the AUVs might be deployed by air or by submarine, and all of the vehicles might not arrive at the same time, or at all. Flexibility is also important when the situation changes, since the way the vehicles were organized may no longer be appropriate. For example, if an AUV fails or is taken out of service (e.g., for preventive maintenance), its sensor suite will no longer be available to the AOSN, and the system must compensate. Efficiency, on the other hand, is important during accomplishment of the mission goals.

Unfortunately, there is an inherent trade-off in any organization between flexibility and efficiency. For example, a hierarchical organizational structure may be quite efficient for some tasks, yet not be able to adapt easily to changed situations—what Malone [1987] called *vulnerability costs*. Other structures, such as committees, may be quite adaptable and flexible, but not particularly efficient.

To address this trade-off, CoDA takes the approach of using a two-level approach to organizing the agents. When initially deployed, a subset of the agents follow a protocol to self-organize into a flexible, loosely-coupled *meta-level organization* (MLO). Which agents can participate (the *MLO agents*) is determined by which agents know the appropriate protocol, which in turn is determined in large part by the intelligence of the agent. The purpose of the MLO is to determine the resources available for carrying out the AOSN's mission, then to design an efficient organization to actually do the mission. This organization, the *task-level organization* (TLO), then receives control from the MLO and conducts the mission.

A TLO will be highly-tailored to fit the current mission and situation (e.g., available sensors, types of agents, environmental conditions, etc.). Most TLOs will have some inherent ability to handle at least slight changes to the situation or mission. However, since a TLO is designed for efficiency, there will occasionally be times when the situation or mission changes enough that the chosen organization is no longer a good fit. When this happens, the TLO initiates the formation of a new MLO, which then redesigns the TLO or creates a new one.

Figure 1 shows the overall two-level organization scheme.

Cooperation Protocols

In order for AUVs to participate in a CoDA-controlled system, they must be able to follow CoDA *cooperation protocols*. A protocol in this sense is essentially set of

[2] Indeed, the original name of CoDA was MAUV.

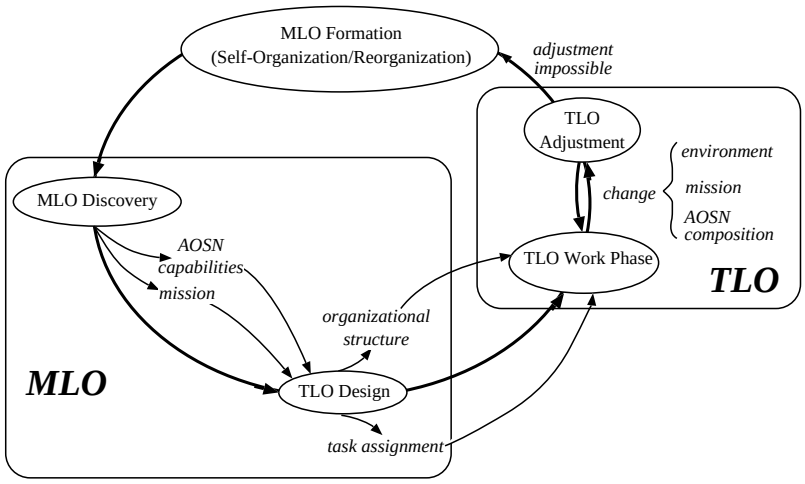

Figure 1: CoDA's two-level organization scheme. [From Turner & Turner, 2001; used by permission.]

rules that govern how an agent responds to and otherwise interacts with the other agents in the system.

Different agents have different protocols, based largely on their intelligence level. For example, sophisticated AI agents will be able to fully participate in a meta-level organization, so they will need protocols to allow them to know how to behave during MLO formation and operation. Other agents that may lack sufficient intelligence will need other protocols to allow them to play their more limited part in the MLO.

Agents need different protocols at different times, too. We divide the normal operation portion of a multi-AUV mission controlled by CoDA into the following phases:

- MLO formation phase
- MLO discovery phase
- TLO design phase
- TLO work phase

In addition, there are phases corresponding to when there are changes or things go badly:

- agent entry phase
- agent exit phase
- error phase
- MLO reformation phase

The following paragraphs deal with each of these in turn.

MLO formation. To give a flavor of what a protocol is, we will present in some detail the protocol for MLO formation; other phases will be described more briefly. This phase of operation begins when agents arrive at the mission work site and ends when an MLO has been formed. The protocols for this phase[3] assume that an agent has no knowledge of the other agents that might be present.

A state machine-like representation of the MLO formation protocol is shown in Figure 2. When an agent is initialized or arrives on-site, its first task is to determine if there is an existing organization present. If so, it should contact that organization to join; if not, it should attempt, with other MLO agents that may be present, to create an MLO.

Toward this end, the agent first broadcasts a special message type ("organization-present?"), then waits for a reply. During the wait, if it hears other agents also trying to determine if there is an existing organization, it takes no action other than to remember their identities. This is how it begins to build a picture of who is in the water with it.

If an existing organization (MLO or TLO) replies within the time-out window, then the agent follows a different protocol for entering an existing organization. If not, then the agent proceeds to the next step, broadcasting an "initiate-MLO" message. It assumes as a first approximation that the only MLO agents present are those from whom it received messages while it was waiting, and it proposes this set of agents, plus itself, as the members of the MLO. It then waits.

While it is waiting, if it receives an "organization-present?" message from an agent, it then adds that agent to the potential members of an MLO and re-broadcasts

[3]Here we discuss primarily the protocols followed by the MLO agents.

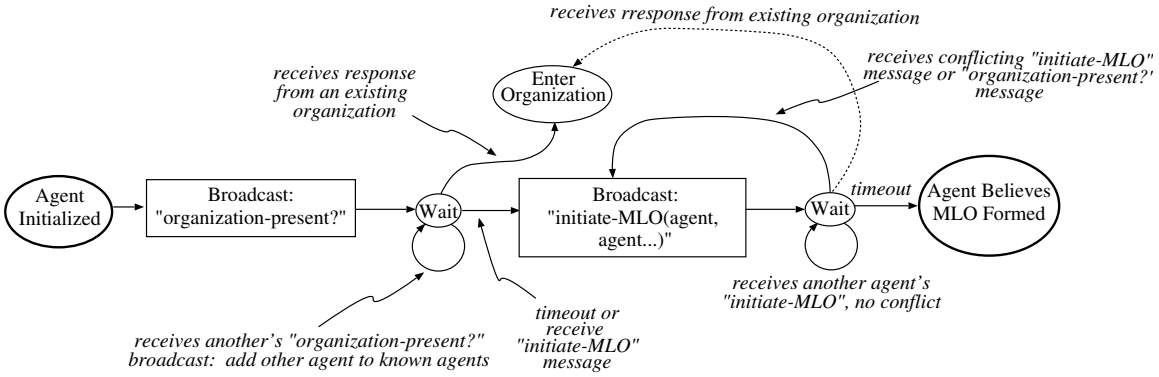

Figure 2: MLO formation protocol for MLO agents. [After Turner & Turner, 2001.]

its own "initiate-MLO" message with the updated list. If it receives another agent's "initiate-MLO" message, then there are two cases. If the members proposed by the other agent are a subset of what it proposed, then there is a meaningful conflict—the other agent does not know about one or more other agents. The agent then should re-broadcast its message to ensure that the other agent learns of those agents. If the other agent's message instead contains a superset of the agent's known MLO members, then it need do nothing except update its own idea of who should comprise the MLO. Note that since there are a finite number of agents and this process is monotonically increasing, it will ultimately terminate.

After a sufficient waiting period with no conflicting messages, then the agent decides that it and all the other MLO agents present have now come to common knowledge and agreement about who will comprise the MLO, and it proceeds to the next phase.

One flaw in the existing protocol is that there is no defined response when an agent is in the second wait state and receives a response from an existing organization. When this happens, the agent should abandon its goal of establishing a new MLO and join the existing organization. This is indicated by a dotted line in the figure, and will be added to a future version of the protocol.

Protocols are currently implemented in our simulator (see below) in the CLIPS expert system shell language [Giarratano, 1993]. The current MLO formation protocol requires 19 CLIPS rules, one of which is shown in Figure 3. This rule is tailored for a simulation of the aggregate properties of a system of agents following the CoDA protocols; a similar rule or its equivalent would exist in an agent that was actually following this protocol.

MLO discovery. After the MLO is formed, it needs to determine what the mission is and what resources it has available. We assume that at least one MLO agent knows the mission and can tell the others. Discovering the resources is more difficult.

First, a MLO agent broadcasts its own location to the MLO, then it broadcasts a message ("whos-there?") that will cause a non-MLO agent, following its own protocol, to respond with its location. There are two versions of the MLO discovery protocol currently implemented; the version discussed here is hierarchical. Since each MLO agent knows where it is and where all its peers are, it can decide when it hears a response from a non-MLO agent if it is the closest MLO agent to it or not. If it is, it becomes the "controller" for that agent, in the sense that it is responsible for knowing about its capabilities. Toward that end, it exchanges messages with the agent to learn its capabilities.

At the end of this phase, which is determined by coordination via broadcast messages and wait states, the total resources available to the MLO are known, although not necessarily by any one agent.

TLO design. This phase of the process is responsible for analyzing the needs of the mission and the available resources (agent capabilities), then designing a task-level organization that is appropriate. In the current version of the protocol, a single agent, called the planner, is chosen via a convention (lexically lowest name) to be responsible for designing the organization. In future versions, the task of design will be distributed among several of the MLO agents.

The planner determines which capabilities (e.g., sensors, behaviors, etc.) are needed for the current mission, then asks the other MLO agents which of themselves or their controlled agents can contribute one or more of those capabilities. Once it has received the

```
(defrule initiate-MLO-conflict1
    "Receive nth initiate-MLO message and experience a conflict."
  ?vip <- (vip (name ?agent) (status new)) ; if there is a new agent who
  ?msg <- (knows ?agent                    ; received an
                 received-message ?        ;  initiate-MLO
                 initiate-MLO $?members);  message with a member list
  ?fact <- (knows ?agent others-exist      ; and it already has knowledge of
         $?others)                         ;  the existance of others and
  (test (<> 0 (length $?others)))          ; my idea of the
  (test (<> (length $?others)              ;  mlo list is NOT a
            (length (union                 ;  subset of the
                     $?others              ;  set of
                     $?members))))         ;  $?members
  =>
  (retract ?msg)                           ; remove the message
  (mlo-msg "%s: initiate MLO received; conflict." ?agent)
  (kill-timer ?agent wait-for-response2) ; kill previous 2nd timer
  (retract ?fact)
  (assert (knows ?agent others-exist (union $?others $?members)))
  ;; broadcast nth "initiate an MLO" message:
  (mlo-msg "%s: scheduling re-broadcast of nth initiate-MLO message." ?agent)
  (broadcast ?agent initiate-MLO (union $?others $?members))
  ;; and set a timer that governs how long to wait:
  (mlo-msg "%s: setting timer 2 to wait for replies." ?agent)
  (set-timer ?*mlo-formation-no-response-timeout2* ?agent wait-for-response2)
  )
```

Figure 3: A CLIPS rule for the MLO formation phase.

replies, it designs the TLO accordingly.

In the current version, organization design is done very simply. First, agents are assigned to mission tasks based on their capabilities using a mechanism described in a section below. Then, a management structure is created. This is a simple hierarchy composed of agents having management capabilities. In the future, the range of organization structures possible will be expanded and the organization design mechanism will be enhanced along the lines discussed in a section below.

After the organization is created, the agents are informed of their assignments, the top-level manager is told to begin work, and the MLO is dissolved. In future work, we will investigate the utility of having the MLO remain active at a low level even while the TLO exists.

TLO work phase. During this phase, the mission is carried out. Protocols for this phase have not been well worked out yet, since we have so far concentrated primarily on organization and reorganization.

Agent entry. When a new AUV enters the AOSN, it needs to follow a protocol to allow it to become part of the existing organization. If there is no organization present, then this is the same as the MLO formation protocol. If there is a TLO present, then the agent tells the top-level manager about its capabilities, and the manager then decides what to do with it. One possibility is to add it to a list of slack resources for later use. Another possibility, to be explored in future work, is for the TLO (or MLO, should it be decided to leave it intact) to recognize that the new agent's capabilities will allow the AOSN to better accomplish its mission. In this case, agent assignments would be modified or, in the extreme case, the TLO would be dissolved and re-designed.

Agent exit. Agents will often leave a long-duration multi-AUV system. This could be due to failure, the AUV being needed elsewhere, the AUV requiring replenishing of resources (e.g., power), or the AUV requiring preventive maintenance. Agents will either know ahead of time that they are about to exit or else will exit abruptly, with no warning (e.g., an agent suddenly fails). A protocol is needed for agents when they know they need to exit, and protocols are necessary for other agents to follow to handle the agent exit. In addition, the top-level manager of the TLO[4] needs a protocol to follow when it knows it needs to exit.

[4]In the current version of CoDA; future organizations may not have the equivalent of a top-level manager.

In the current version of CoDA, if the top-level manager realizes it needs to leave the system, it causes the overall system to transition back to the MLO formation protocol to re-create the meta-level organization. This is a bit extreme, and should be reconsidered in future: it is quite possible that a better approach under some circumstances would be to select another agent to fill the top-level management role. If the manager exits abruptly, then, like some other MAS protocols [e.g., Smith, 1980], it is up to its subordinates to notice that the manager is gone. This may happen, depending on the work phase protocol in use, when the manager fails to respond to status messages, requests for information, etc. In this case, the subordinate's protocol directs it to first ask the other agents (via a broadcast message) if anyone disagrees with re-forming the MLO, waiting, then, if no one does, initiating MLO formation.

If an agent exits that is not the top-level manager, and it has time, then it tells the top-level manager (in a hierarchical TLO) that it is exiting. If it does not have time, then it is up to its manager to notice that it has exited. the manager then sends the top-level manager a message notifying it of the problem. It is then up to the top-level manager to decide if it can repair the TLO or not. If not, then it initiates the re-formation of the MLO.

Error. Errors in carrying out the mission are not addressed in the current CoDA protocol suite. This will require a more sophisticated model of the mission and the organization than currently exists.

MLO reformation. The re-formation of the MLO is triggered, as discussed above, by messages from an agent within the TLO when it recognizes that something is seriously wrong. Since we assume that the agents are cooperating and not malicious, the current protocols allow any agent to send the re-formation message, and, once it is sent, the other agents then switch to their MLO formation protocols. Unless the triggering agent is the top-level manager, it will first ask others if they disagree before issuing the message. In the future, we will need to address the situation in which the triggering agent is wrong about the need to re-form the MLO and how other agents can disagree. Depending on the application of the multi-AUV system being controlled, consideration may be needed to agents being non-cooperating or even malicious.

Organization of Multi-AUV Systems

A hard problem for multi-AUV systems is how organize the resources such that the mission is accomplished efficiently. This is the task of the MLO in CoDA. We first describe the current mechanism for organization design, then discuss our planned version.

Currently, all TLOs are hierarchical, and the mechanism for creating the TLO is very simple. As discussed above, a planner is selected from among the members of the MLO, and it designs the organization. First it runs the task-assignment algorithm (see below) to assign agents to mission tasks based on matching their capabilities to what the task needs. Then, based on the heuristic that every task needs to be managed, it creates the management structure from the bottom up by assigning agents management roles. An agent can have both worker and manager roles, since the requirements for these different roles (e.g., sensors versus communication and decision-making abilities) may not overlap significantly. When a task has more assigned agents than any single manager can manage, then a middle-management "pseudo task" is created. A heuristic used is to prefer managers for a task to be selected from among the agents assigned to the task. Other heuristics are possible, of course, for example, selecting nearby agents to manage tasks, etc.

The current organization design mechanism was meant to be a place-holder until we could turn our attention to the real thing. We have begun work on the actual organization design mechanism, which is based both on the organization design literature and on work done in a related project on context-sensitive reasoning.

Human organizations, although not completely isomorphic to multiagent systems, are still a rich source of ideas about how to organize cooperative collections of agents to solve problems. Early work on distributed AI began looking at human organizations [e.g., Malone, 1987; Fox, 1981]. In addition to making use of that work, we have begun looking at the organization design literature in order to identify kinds of organizations that may be appropriate for multi-AUV systems. For example, we are currently using a simple hierarchy, but other organizational structures that show promise include teams, committees, and markets, as well as mixed organizational types. Others have studied the use of some of these for MAS as well [e.g., Tambe, 1997; Smith, 1980; Sandholm, 1993].

We also look to the organizational design and distributed AI (DAI) literature to help determine the situational features that make one organizational structure favored over another. For example, markets are good for situations in which little may be known about some agents [Smith, 1980] or in which there are likely to be agent failures [Malone, 1987] or high complexity [Fox, 1981]. Alternatively, hierarchies are likely better in those situations in which it may be necessary to compel

an agent to perform a task that is not in its own best interest [cf Fox, 1981].

As a starting point for organization design, we are investigating the use of a technique for organization selection adapted from our other work on context-sensitive reasoning, *context-mediated behavior* (CMB) [Turner, 1998], which is part of the Orca Project. In CMB, an agent's knowledge of how to behave is largely contained in knowledge structures called *contextual schemas*, or c-schemas, that represent classes of situations in which the agent has or may operate. For organization selection, the c-schemas would in addition be indexed by features of the multiagent system (e.g., number of agents, their type, etc.) and would contain knowledge about the kind of organizational structure that is appropriate for the situation.

A major task in CMB is identifying the current context, which may include finding and merging several c-schemas that each represent the current situation. This is done using a diagnostic algorithm that uses the features of the currently-observed situation as "symptoms" that predict and confirm a diagnosis of the context [Arritt & Turner, 2003]. Situational features are used to probe a content-addressable memory of c-schemas. Building on work done in medical AI, the c-schemas are grouped into *logical competitor sets* (LCSs) [Miller *et al*., 1982; Feltovich *et al*., 1984], each of which contains c-schemas that (roughly) explain the same set of features. Each c-schema is scored based both on the *evoking strengths* of the features that retrieved it, i.e., how strongly those features "brought it to mind," as well as the predictions that were violated and confirmed. The topmost LCS is then "solved" by gathering additional information or making additional inferences to separate one diagnosis from the others. Then the process repeats, until there are no unexplained features and no LCSs that are unsolved.

In medical diagnosis, the solutions to the LCSs comprise disease or diseases the patient is suspected of having. In CMB, on the other hand, the c-schemas' information is merged to give an overall picture of the current situation. Information from the merged context is then used to control the agent's behavior or, in this case, to suggest appropriate organizational structures for the TLO. The organizational structures suggested would then be used as the starting point for designing the TLO, either by selecting the best one, or by merging two or more of them.

In future work, we will further elucidate and implement this scheme for organization design. In addition, we will investigate how to distribute the task of organization design among several or all of the agents in the MLO.

TASK ASSIGNMENT

A key task of multiagent system control is assigning the appropriate agent to each task. This is a particularly vexing problem in multi-AUV systems of the type in which we are interested, since the agents are heterogeneous. This means, for example, that most existing AOSN techniques [e.g., Phoha *et al*., 1997; Smith *et al*., 1996] would need to be extended for these types of multi-AUV systems.

The task assignment problem is amenable to constraint-based reasoning techniques. Tasks require particular capabilities, and agents provide those capabilities. Each capability (e.g., a particular sensor type, such as a side-scan sonar) required by a task can be considered a variable whose domain consists of those AUVs having that capability. Constraints would be represented as links (arcs) between variables or AUVs, for example, to indicate when two capabilities cannot be used simultaneously, to indicate an AUV cannot be in two places at once, or to represent resource constraints.

However, there are some properties of these multi-AUV systems that are problematic for standard constraint satisfaction problem (CSP) techniques [e.g., Freuder, 1988], such as those for job-shop scheduling [Fox, 1987]. For example, each agent has multiple, possibly independently-assignable capabilities (e.g., sensors, communication ability, localization ability, etc.), as well as resource constraints, and there are multiple ways to accomplish many tasks (e.g., use conductivity-temperature-depth (CTD) sensors or temperature sensors, etc.).

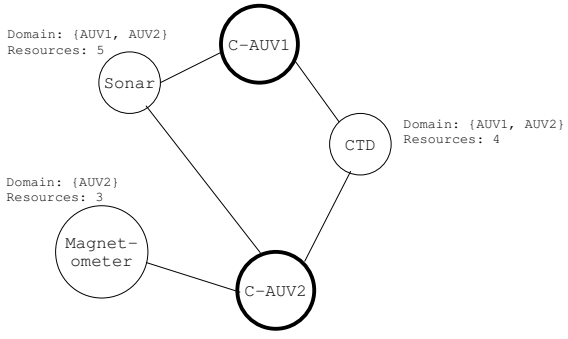

Figure 4: A constraint graph with two AUVs, two constraints, and three variables corresponding to capabilities required for a task.

In addition, constraints are not solely between CSP variables, as is standard, but also between an AUV and all the task capabilities (CSP variables) in whose domain

it appears. For example, Figure 4 shows a portion of a constraint graph. There are three task capabilities shown, each corresponding to a variable, representing the need for a sonar, a magnetometer, and a CTD sensor. Two AUVs are under consideration for these capabilities, as shown in the domains of the variables. The bold circles are constraint nodes. Each of these corresponds to one of the AUVs. They represent the constraint on the total resources (e.g., energy) that can be expended by the AUVs. For example, C-AUV1 ensures that if the resources needed for the sonar and the CTD capabilities exceeds the total resources available to the AUV, then AUV1 will not be selected for both roles.

We also assume that there may be multiple ways to achieve any particular task. For example, to take a temperature profile of an area, AUVs with CTD can be used, as can those with simple temperature sensors. This means that there are multiple ways to build the constraint graph representing the problem, depending on which task alternatives are selected. This is a problem for standard CSP approaches.

Our task assignment mechanism [Turner & Turner, 1999] begins with a type of constraint-based reasoning called *constrained heuristic search* (CHS) [Fox *et al*., 1989]. CHS combines heuristic search with CSP techniques, deciding what to do at any point based on heuristics based on the topology and other properties (*textures*) of the constraint network. We extended CHS in two ways: by adding the n-ary constraints needed to represent resource constraints of AUVs with multiple capabilities, and by allowing CHS-inspired heuristics and mechanisms to not only handle partial constraint graphs, but also to help select alternatives to build the constraint graph.

Our approach begins with a *task-decomposition tree* (TDT), an AND–OR tree representing all possible alternative ways of carrying out the mission (see Figure 5). This could be given to the system by a human, or it could be created using standard artificial intelligence planning techniques. Branch points without arcs represent choice points between alternatives. Branches of the tree connected by arcs are ones that all need to be done if the parent node is selected. The leaves of the tree are capabilities that are required for tasks. Since each of these leaves will ultimately become a variable in a CSP, they are also marked with the estimated resources needed for using that capability in the task[5] and with the set of vehicles that have that capability—the latter will be added by the MLO planning agent based on what is found out during the MLO discovery phase.

The overall form of the task-assignment mechanism is a heuristic search. Each state consists of a constraint graph representing the portion of the mission already considered and a task-decomposition tree for the rest. The start state has the initial TDT and an empty constraint graph. A goal state has an empty TDT and a constraint graph representing all the tasks necessary to accomplish the mission with each constraint variable having a one-element domain (i.e., each variable has a value). The available operators to move from state to state are to choose an OR branch in the current state's TDT, choose the AND branch to add next, or to set a variable to a value.

Selecting which operator to apply is guided by heuristics based on the structure of the constraint graph that is predicted to exist in the successor state, should the operator be applied. We base the heuristics on the textures of CHS. For example, the *value goodness* texture is used when selecting an OR node's branch to pick the alternative that can be satisfied by the most values. This gives the algorithm flexibility, since it keeps number of the ways to satisfy the constraints as large as possible. The *constraint reliance* texture is used at an AND node to pick the subtask with the fewest alternatives, i.e., that is most constrained. This helps insure that important, limiting constraints are present in the graph early, so that the algorithm can insure that they can be satisfied before doing too much additional work.

The textures in CHS were meant to guide the selection of variables in an existing constraint graph to set and which values to set them to. However, in our approach, we are interested in building the constraint graph. Consequently, we estimate the values the heuristic needs at the interior nodes of the graph by propagating values from the leaves upward, thus marking the tree. For example, for the *value goodness* texture, at the leaves, the size of the domain of the variable is used. This is propagated to higher nodes in the tree by taking the minimum of child nodes' values at AND nodes and taking the maximum at OR nodes. When selecting an alternative at an OR node, the one highest value computed in this way is selected. For *constraint reliance*, the domain size is again used, and the minimum is again used at AND nodes, but the sum is used at OR nodes. When selecting a subtask to work on at an AND node, the subtask with the lowest value computed this way is selected.

Selections from the TDT are made until a bottom-level subtask's capabilities' variables are added to the

[5]This is a drastic simplification of how resources will ultimately need to be represented. Making this more realistic is the subject of future work.

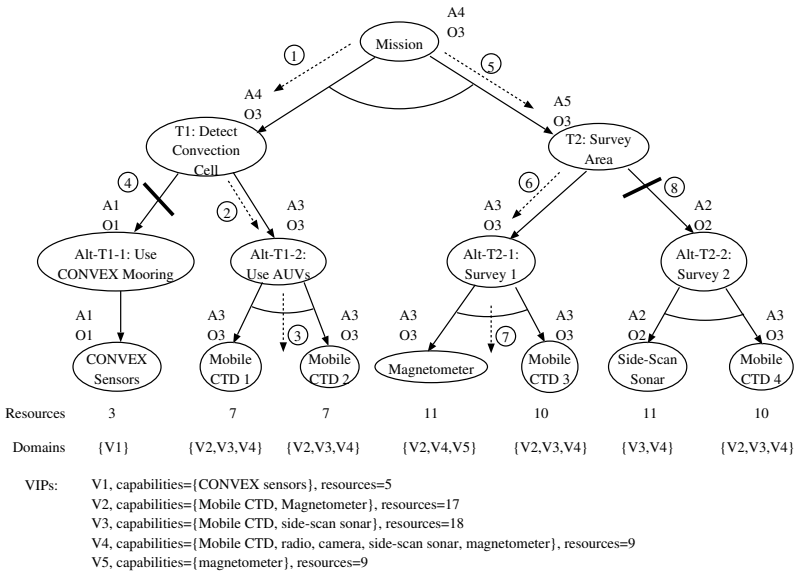

Figure 5: A task-decomposition tree (TDT) for an AOSN mission. [From Turner & Turner, 1999.]

constraint graph. At that point, the usual CHS algorithm is run. The result is either an inconsistent constraint graph, in which case the algorithm backtracks, or one that is consistent (and possibly solved). The overall algorithm terminates when all variables have been added to the constraint graph and it is solved.

THE CODA/CADCON SIMULATOR

Early in the project, we realized that we needed a simulator to allow us to develop and test the work. The CoDA simulator, which has been described elsewhere [Turner & Turner, 2000; Albert *et al.*, 2003], is a multi-fidelity simulator capable of simulating AOSNs or other multi-AUV systems at a variety of levels, from a high-level simulation of the aggregate properties of the system to a low-level, high-fidelity simulation of agents carrying out their tasks. This is useful, as it allows us to quickly simulate the effects of a group of agents following protocols without worrying about how the agents would actually make the decisions necessary to follow them. This allows us to see if we are on the right track. Later, we can add the decision-making mechanisms to the simulated agents that actual agents would use to help insure that the system would work in the real world. In addition, the simulator was instrumented so that we could run simulation experiments and gather useful statistics.

The first CoDA simulator used a variety of technologies. The expert system language CLIPS [Giarratano, 1993] was used to simulate the protocols as a set of rules. This worked in conjunction with versions of the task-assignment and organization-design algorithms written in Lisp. Communication between the two, due to some shortcomings of CLIPS for our purposes, was via Unix named pipes (FIFOs) and standard input/standard output. For debugging, CLIPS could run interactively and be in control of the Lisp code. This allowed the CLIPS interface to be used for debugging the ruleset. For experiment runs, a Lisp-based experiment harness was in control. The simulator was instrumented using the Lisp-based CLIP/CLASP [Anderson *et al.*, 1995] experiment and statistical package.

Recently, the CoDA simulator was interfaced to the Autonomous Undersea System Institute's (AUSI's) CADCON high-fidelity multi-AUV simulator [Albert *et al.*, 2003]. This gave CoDA the ability to run simulations incorporating hydrodynamics of vehicles, as well as providing a very nice interface and visualization tool. It extended CADCON by providing control algorithms for the simulated agents.

Currently, the simulator is being redesigned and rewritten entirely in Lisp to avoid the hodgepodge of languages and to avoid some of CLIPS' shortcomings. As CoDA is at the level of maturity needing it, we are rewriting the simulator first to perform agent-level rather than aggregate simulation. That is, simulated agents will be present as software agents, not simply as facts in a rule-based system's knowledge base. This will provide a higher level of fidelity of simulation than the

current simulator. Initially, the expert system language LISA [Young, 2004], a Lisp-based version of CLIPS, along with custom Lisp code, will be used to simulate the decision processes of each agent. Each agent will have its own Lisa/Lisp process as its agent program. Ultimately, other agent programs, for example, our Orca mission controller, will be used to control the agents. Later, we will add back the aggregate property simulation abilities to make the new version of the CoDA simulator multi-fidelity as well.

Conclusion and Future Work

At the time of writing, CoDA protocols, organization techniques, task-assignment mechanisms, and the simulator are at a state of maturity to allow simulation of aggregate properties of an AOSN following the CoDA protocols. Simulation experiments have been run [Turner & Turner, 2001] showing that a CoDA-controlled AOSN can self-organize and reorganize appropriately when the situation changes. The two-level organization approach is novel and effective in balancing the need for flexibility and efficiency in the types of AOSNs in which we are interested. The task-assignment mechanism is able to rapidly assign agents to tasks, choosing from among alternative ways of accomplishing tasks while taking into consideration constraints and properties of the resulting system. In sum, we believe the approach to multi-AUV control taken in the CoDA project is a very good one for heterogeneous, autonomous multi-AUV systems engaged in long-duration, complex missions.

Although much has been accomplished so far in the CoDA project, it remains very much a work in progress. Future work will focus on all parts of the project, including the protocols, organization design, task assignment, and the simulator.

With respect to the protocols, additional attention is needed in the short term to protocols for the mission work phase and for detecting errors in AOSN operation, as noted above. Also, as we simulate AOSNs at a higher level of fidelity, we will doubtless uncover flaws and shortcomings in the current protocol suite. As we expand the kinds of missions and agents considered, we will also discover the need for variants of protocols for different mission types and classes of agents.

Our laboratory is also deeply involved in research on modeling and using contextual knowledge to guide problem solving [e.g., Turner, 1998], and we foresee a role for context-sensitive reasoning in CoDA's protocols, as well. Agents of sufficient intelligence should be able to reason about the context in which they find themselves—the mission type, their own capabilities and status, who they are working with, and the environment they are in. If so, then when they recognize the context, one piece of information that should be associated with the contextual knowledge is which protocol to use in the current context. This can help the agent know when to switch protocols to switch phases of operation (e.g., from MLO formation to discovery) as well as to choose the appropriate variant of a protocol for the current situation.

Much work is also needed on the organization design mechanism, as noted above. Some work has already been done to identify features of the AUV and AOSN domains that would predispose to one type of organization over another, and work has also begun on identifying those types of human organizations appropriate for AOSNs. Additional work is needed on these areas. In addition, the outline for a context-based organization selection mechanism discussed above needs to be fleshed out and implemented, and mechanisms for creating an organization need to be developed. This includes ways to merge organizations, when contextual schemas suggest more than one.

With respect to task assignment, future work will focus on distributing the process among several MLO agents. Some work has been done on this already, based on work on distributed constraint satisfaction. As our approach is based in part on CHS, we will also consider the distributed version of this algorithm as well, DCHS [Sycara *et al*., 1991].

Work will also be done to merge the task assignment process with the organization design mechanism. So far, we have considered these as two separate processes in order to simplify the problem. However, the two are obviously interdependent: which organization is selected determines in part which decomposition of the TDT is desirable, as well as which roles are present that require agents to be assigned (e.g., management roles); task assignment also affects organization selection, since the assignment of AUVs to tasks may mandate particular management tasks be present. We will in the near future look at the interactions between these two processes, and how the two should be interleaved.

The CoDA simulator will also be the subject of future work. The agent-based simulator currently being implemented will be augmented by adding back the aggregate property simulation facilities of the old simulator, thus making it, too, a multi-fidelity simulator.

We will also begin to explore how to integrate CoDA into AUV control programs, both our own (Orca) and other laboratories'. This may include using versions of their programs in our simulator, or allowing other laboratories' programs to communicate over the Internet

to control simulated agents in the CoDA/CADCON simulator. We will examine among other things the question of whether it makes sense to have others' mission controllers incorporate CoDA protocols into the set of things they know how to do, to implement a standalone "CoDA module" that cooperates with the controllers, or both, depending on the situation.

Finally, further simulation experiments will be performed to gauge the usefulness of the CoDA protocols, organization selection/design techniques, and task-assignment mechanism. The results of these experiments will allow us to fine-tune CoDA and move toward in-water tests aboard actual AUVs engaged in multi-AUV missions.

ACKNOWLEDGMENTS

The authors would like to the other members of MaineSAIL, both those who have worked on this project and those who have offered helpful suggestions. We especially thank Erik Albert, Andrew Sylvia, and Andrew Desmarais for their work on the CoDA/CADCON simulator, and Robert Arritt for just generally keeping MaineSAIL's infrastructure running. We also thank Steve Chappell, Dick Blidberg, and Rick Komerska of AUSI for the use of the CADCON simulator and for helpful suggestions (and, in the case of Steve, programming) over the years.

REFERENCES

Albert, E., Bilodeau, J., & Turner, R. M. (2003). Interfacing the CoDA and CADCON simulators: A multi-fidelity simulation testbed for autonomous oceanographic sampling networks. In *Proceedings of the Thirteenth International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, NH. The Autonomous Undersea Systems Institute.

Albus, J. S. (1988). Multiple Autonomous Undersea Vehicles. Technical Report NIST Technical Note 1251, National Institute of Standards and Technology, Gaithersburg, MD 20899.

Anderson, S. D., Westbrook, D. L., Schmill, M., Carlson, A., Hart, D. M., & Cohen, P. R. (1995). *Common Lisp Analytical Statistics Package: User Manual*. Department of Computer Science, University of Massachusetts.

Arritt, R. P. & Turner, R. M. (2003). Situation assessment for autonomous underwater vehicles using a priori contextual knowledge. In *Proceedings of the Thirteenth International Symposium on Umanned Untethered Submersible Technology (UUST)*, Durham, NH. The Autonomous Undersea Systems Institute, Lee, NH.

Blidberg, D. R. & Chappell, S. G. (1986). Guidance and control architecture for the EAVE vehicle. *IEEE Journal of Oceanic Engineering*, OE–11(4):449–461.

Curtin, T., Bellingham, J., Catipovic, J., & Webb, D. (1993). Autonomous oceanographic sampling networks. *Oceanography*, 6(3).

Feltovich, P. J., Johnson, P. E., Moller, J. A., & Swanson, D. B. (1984). LCS: The role and development of medical knowledge and diagnostic expertise. In Clancey, W. J. & Shortliffe, E. H., editors, *Readings in Medical Artificial Intelligence*, pages 275–319. Addison–Wesley Publishing Company, Reading, Massachusetts.

Fox, M. S. (1981). An organizational view of distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, 11:70–80.

Fox, M. S. (1987). *Constraint-directed Search: A Case Study of Job-Shop Scheduling*. Morgan Kaufmann Publishers.

Fox, M. S., Sadeh, N., & Baykan, C. (1989). Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 309–315, Detroit, MI.

Freuder, E. C. (1988). Backtrack-free and backtrack-bounded search. In Kanal, L. & Kumar, V., editors, *Search in Artificial Intelligence*. Springer–Verlag, New York, NY.

Giarratano, J. C. (1993). *CLIPS User's Guide*. NASA, Information Systems Directorate, Software Technology Branch, Lyndon B. Johnson Space Center, Houston, TX.

Malone, T. W. (1987). Modeling coordination in organizations and markets. *Management Science*, 33(10):1317–1332.

Miller, R. A., Pople, H. E., & Myers, J. D. (1982). INTERNIST–1, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–476.

Phoha, S., Stover, J., Gibson, R., Peluso, E., & Stadter, P. (1997). Autonomous ocean sampling mobile network controller. In *Proceedings of the Tenth International Symposium on Unmanned Untethered Submersible Technology (UUST)*, pages 362–374, Durham, NH.

Sandholm, T. (1993). An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 265–262.

Smith, R. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE*

Transactions on Computers, C–29(12):1104–1113.

Smith, S., Ganesan, K., Dunn, S., & An, P. (1996). Strategies for simultaneous multiple AUV operation and control. In *IARP'96*, France.

Sycara, K., Roth, S., Sadeh, N., & Fox, M. (1991). Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1446–1461. (Special Issue on Distributed AI)

Tambe, M. (1997). Agent architectures for flexible, practical teamwork. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 22–28.

Turner, E. H. & Turner, R. M. (1999). A constraint-based approach to assigning system components to tasks. *International Journal of Applied Intelligence*, 10(2/3):155–172.

Turner, R. M. (1995). Intelligent control of autonomous underwater vehicles: The Orca project. In *Proceedings of the 1995 IEEE International Conference on Systems, Man, and Cybernetics*. Vancouver, Canada.

Turner, R. M. (1998). Context-mediated behavior for intelligent agents. *International Journal of Human–Computer Studies*, 48(3):307–330.

Turner, R. M., Fox, J. S., Turner, E. H., & Blidberg, D. R. (1991). Multiple autonomous vehicle imaging system (MAVIS). In *Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology (AUV '91)*, pages 526–536, Durham, NH.

Turner, R. M. & Turner, E. H. (1998). Organization and reorganization of autonomous oceanographic sampling networks. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA'98)*, pages 2060–2067, Leuven, Belgium.

Turner, R. M. & Turner, E. H. (2000). Simulating an autonomous oceanographic sampling network: A multi-fidelity approach to simulating systems of systems. In *Proceedings of the Conference of the IEEE Oceanic Engineering Society (OCEANS'2000)*, Providence, RI.

Turner, R. M. & Turner, E. H. (2001). A two-level, protocol-based approach to controlling autonomous oceanographic sampling networks. *IEEE Journal of Oceanic Engineering*, 26(4).

Young, D. E. (2004). Lisa - intelligent software agents for common lisp. On the World Wide Web at `http://lisa.sourceforge.net`, accessed June 10, 2004; last updated June 9, 2004.