Learning Organizational Roles in a Heterogeneous Multi-agent System

M V Nagendra Prasad, Victor R Lesser and Susan E Lander[1]
Department of Computer Science
University of Massachusetts

UMass Computer Science Technical Report 95–35

Abstract

Previous work in self-organization for efficient distributed search control has, for the most part, involved simple agents with simple interaction patterns. The work presented in this paper represents one of the few attempts at demonstrating the viability and utility of self-organization in an agent-based system involving complex interactions within the agent set. We present a multi-agent parametric design system called L-TEAM where a set of heterogeneous agents learn their organizational roles in negotiated search for mutually acceptable designs. We tested the system on a steam condenser design domain and empirically demonstrated its usefulness. L-TEAM produced better results than its non-learning predecessor, TEAM, which required elaborate knowledge engineering to hand-code organizational roles for its agent set. In addition, we discuss experiments with L-TEAM that highlight the importance of certain learning issues in multi-agent systems.

[1] This work is supported in part by the National Science Foundation Cooperative Agreement Grant No. EEC 9209623. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

1 Introduction

In this paper, we present a multi-agent heterogeneous system called L-TEAM that learns to organize itself to let the agents play the roles they are best suited for in a distributed search process. L-TEAM is an extension of the TEAM framework[4] for cooperative search among a set of heterogeneous reusable agents. A reusable agent system is an open system assembled by minimal customized integration of a dynamically selected subset from a catalogue of existing agents. Reusable agents may be involved in systems and situations that may not have been explicitly anticipated at the time of their design. Each agent works on a specific part of the overall problem. The agents work towards achieving a set of local solutions to different parts of the problem that are mutually consistent and satisfy, as far as possible, the global considerations related to the overall problem. As a part of this search process agents augment their local view of the composite search space with meta-level information about search spaces of other agents through negotiation to minimize the likelihood of generating conflicting solutions[4]. TEAM was introduced in the context of parametric design in multi-agent systems. Each of the agents has its own local state information, a local database with static and dynamic constraints on its design components and a local agenda of potential actions. The search is performed over a space of partial designs. It is initiated by placing a problem specification in a centralized shared memory that also acts as a repository for the emerging composite solutions (i.e. partial solutions) and is visible to all the agents. Any design component produced by an agent is placed in the centralized repository. Some of the agents initiate base proposals based on the problem specifications and their own internal constraints and local state. Other agents in turn extend and critique these proposals to form complete designs. An agent may detect conflicts during this process and communicate feedback to the relevant agents; consequently affecting their further search by either pruning or reordering the expansion of certain paths. The evolution of a composite solution in TEAM can be viewed as a series of state transitions as shown in Figure 1 (from [4]). For a composite solution in a given state, an agent can apply a set of negotiated search operators represented by the set of arcs leaving that state. An agent can be working on several composite solutions concurrently. Thus at a given time, an agent is faced with the problem of choosing an operator from a set of allowed operators that can be applied to one of the composite solutions that can potentially be worked upon. This decision is complicated by the fact that an agent has to achieve this choice within its local view of the problem-solving situations (see [4]).

The objective of this paper is to investigate the utility of machine learning techniques as an aid to such a decision process in situations where the set of agents involved in problem solving are not necessarily known to the designer of any single agent. The results in this paper demonstrate the effectiveness of learning techniques for such a task and then go on to provide empirical support to two important observations regarding learning in multi-agent systems:

1. A credit assignment scheme can lead to enhanced learning if it considers the relations between an action and the progress of the overall problem-solving process, in addition to the end result that the action may lead to. This may be especially true of systems with complex interactions. Models of interactions between agents and the type of meta-level information exchanged can be exploited to aid the learning process.

2. In multi-agent systems, an agent may experience divergence of local view of the global problem solving state and the actual global problem solving state. Such a divergence may

lead to a deterioration in the ability of an agent to exploit the knowledge that is learned.

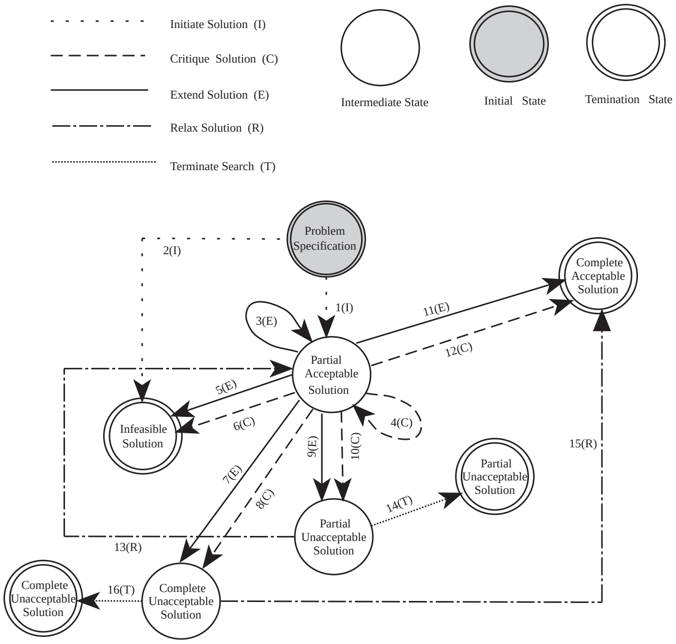

Figure 1: Negotiated Search

The rest of the paper is organized as follows. Section 2 discusses the characteristics of a distributed search space and Section 3 presents our use of the UPC formalism as a basis for learning organizational knowledge. The following section discusses an implementation of L-TEAM based on this algorithm and presents the results of our empirical explorations. We conclude by discussing some related work and the implications of this work.

2 Complex Distributed Search Spaces

Problem domains like those dealt with in TEAM can be viewed as comprising a set of interdependent subproblems. Overall solution to a problem is obtained by aggregation of solutions to its subproblems. In these domains, partial search paths over a composite search space are interrelated in such a way that the extension of a path in the search space of one subproblem may effect the results of extending another path, perhaps in another subproblem. In such complex search spaces, there is a need for organizing the search in such a manner as to choose those actions that lead to generation of helpful constraints for the subsequent searches for solving related subproblems. In multi-agent systems like TEAM, the situation is further complicated by the fact that the search space is distributed across many agents. In such systems, organizational knowledge can be described as a specification of the way the overall search should be organized in terms of which agents play what roles in the search process and communicate what information, when and to whom.

Each agent in TEAM plays some organizational role in distributed search. A role is a task or a set of tasks to be performed in the context of a single solution. A pattern of activation of roles in an agent set is a role assignment. All agents need not play all organizational roles; which in turn implies that agents can differ in the kinds of search operators they are allotted. Organizational roles played by the agents are important for the efficiency of a search process and the quality of final solutions produced.

During each cycle of operator application in TEAM, each agent in turn has to decide on the role it can play next, based on the available partial designs. An agent can choose to initiate a new design or extend an already existing partial design or critique an existing design. The agent needs to decide on the best role to assume next and accordingly construct a design component. It can be an extremely difficult task for a system designer to construct a generic rating function for each agent that takes into account the specifics of the agent set and the complexities of the search controlling the design of a specific artifact. In this paper we propose learning methods that let the agents construct these rating functions based on past problem solving experience.

3 Learning Role Assignments

The formal basis for learning search strategies adopted in this paper is derived from the UPC formalism for search control (see Whitehair and Lesser[17]) that relies on the calculation and use of the **U**tility, **P**robability and **C**ost (UPC) values associated with each $\langle state, op, final\ state\rangle$ tuple. Utility component represents the present state's estimate of the final state's expected value or utility if we apply operator *op* in the present state. Probability represents the expected uncertainty associated with the ability to reach the final state from the present state, given that we apply operator *op*. Cost represents the expected computational cost in terms of reaching the final state. These values comprise an explicit representation of the position of a search state with respect to the potential final states in a search space. Additionally, in complex search spaces, for which the UPC formalism was developed, an application of an operator to a state does more than expand it. The operator application may result in an increase in the problem solver's understanding of the interrelationships among states. In these situations, an operator that looks like a poor choice from the perspective of a local control policy may actually be a good choice from a more global perspective due to some increased information it makes available to the problem solver. This property of an operator is referred to as its *potential* and it needs to be taken into account while rating the operator. An evaluation function defines the objective strategy of the problem solving system based on the UPC components of an operator and its potential. For example, a system may want to reach any final state as quickly as possible with high quality solutions or it may want maximum utility per unit cost. The evaluation function is applied to all the operators applicable to any frontier states of the on-going search and an operator that maximizes the ratings of all the applicable operators is selected.

Starting from this core of UPC formalism, we modify it to suit our purpose of learning organizational roles in negotiated search in multi-agent systems. Our first modification involves classification of all possible states of a search into a pre-enumerated finite class of situations. These classes of situations represent abstractions of the state of a search. Thus, for each agent, there is UPC vector per situation per operator leading to a final state. A situation in L-TEAM is represented by a feature vector whose values determine the class of a state of the search. Note that in order to get the values of a situation vector at a node, an agent might have to communicate with other agents to obtain the relevant information regarding features that relate to their internal state. In L-TEAM, an agent responsible for decision making at the node obtains the UPC values based on the situation vector for all the roles that are applicable in current state. Depending on the objective function to be maximized, these UPC vectors are used to choose a role to be performed next. During learning, an organizational role is chosen probabilistically in the ratio of its rating to the sum of the ratings of all the possible organizational roles for an agent in the given situation. This permits the system to explore the contributions of all the roles probabilistically. Once the learning is done, an agent chooses the role with maximum rating in a given situation. This implies that after the learning phase, each agent organizes itself to play a fixed role in a given situation.

We use the *supervised-learning approach* to prediction learning (see [14]) to learn estimates for the UPC vectors for each of the states. At each node[1] of the distributed search tree, the corresponding situation vector of the features representing the relevant problem-solving activities at that time and an agent's choice of the role it plays are stored by that agent. The performance measures arising out of this decision will not be known at that time and become available only at the completion of the search. After a sequence of such steps leading to completion, the performance measures for the entire problem solving process are available. The system then back traces through these steps assigning credit for the performance to the decisions at the nodes involved (the exact process will be described below)[2]. At each node, values of the UPC vector for the operator corresponding to the situation at that node are adjusted. In our use of the UPC framework, we assume that there is a single final state — the generation of a mutually acceptable complete design.

Let $\left\{S_j^k\right\}$, $1 \leq$ j $\leq M_k$, be the set of possible situation vectors for Agent k where each situation vector is a permutation of the possible values for the situation-vector features and let op_i^k, $1 \leq$ i $\leq N_k$, be the operators Agent k possesses. Agent k has $M_k * N_k$ UPC vectors $\left\{op_i^k, S_j^k, Agent\ k, U_{ij}^k, P_{ij}^k, C_{ij}^k, Pot_{ij}^k\right\}$. Given a situation S_b^k, objective function $f(U, P, C, Pot)$ is used to select an operator op_a^k such that

$$\begin{aligned} Prob(op_a^k) &= \frac{f(U_{ab}^k, P_{ab}^k, C_{ab}^k, Pot_{ab}^k)}{\sum_i f(U_{ib}^k, P_{ib}^k, C_{ib}^k, Pot_{ib}^k)} \quad During\ \ learning \\ op_a^k &= {}^k f_{op}^{-1} \max_i f(U_{ib}^k, P_{ib}^k, C_{ib}^k, Pot_{ib}^k) \quad After\ \ learning \end{aligned}$$

where $1 \leq$ i $\leq$ N, and ${}^k f_{op}^{-1}(rating)$ represents the operator whose UPC values are such that $f(U, P, C, Pot)$ = rating.

Let $\mathcal{T}$ be the search tree where each node is annotated with the triple $\left\{op_i^k, S_j^k, A_k\right\}$ representing the application of operator op_i^k in situation S_j^k by Agent k. Let $\mathcal{F}(\mathcal{T})$ be the set of states on the path to the terminal state T. A terminal state is a state that is not expanded further due to detection of a success or a failure. A final state is a terminal state where the search ends successfully with a mutually acceptable design. For example, let the following sequence of operator applications leads to a terminal state - say a success.

[1] A node represents a decision point for an agent.

[2] Note that the supervised learning approach to prediction learning is different from reinforcement learning which assigns credit by means of the differences between temporally successive predictions[14]. In this paper we are primarily concerned with showing the benefits and characteristics of learning in multi-agent systems rather than with the merits of a particular learning method over others. The reason for choosing supervised learning method is simply that it worked for us.

$$[\{op_1, S_1, A1\}, \{op_2, S_2, A2\}, \ldots, \{op_m, S_m, Am\}]$$

When the search enters a terminal state, the performance measures are back-propagated to the relevant agents. In this case, the agents $A1, A2, \ldots, Am$ adjust the UPC values for their respective situation-operator pairs. Schemes for doing adjustments to various performance measures are discussed below.

There are various ways of doing the actual changes to the UPC and Potential values of each situation vector and we discuss some simple schemes here. Let ${}_{(p)}U^k_{ij}$ represent the predicted utility of the final solution achieved by Agent k using an operator i in a state n that can be classified as situation j, accumulated after p problem solving instances and $\mathcal{F}(\mathcal{T})$ be the set of states on the path to a final state F. Let U_F be the utility of the solution and let $0 \leq \alpha \leq 1$ be the learning rate. Then:

$$_{(p+1)}U^k_{ij} \;\; = \;\; {}_{(p)}U^k_{ij} + \alpha\,(U_F - {}_{(p)}U^k_{ij}), \;\; n \in \mathcal{F}(\mathcal{T}), \quad state \;\; n \; \in situation \; j$$

Thus Agent k that applied op_i, modifies the Utility for its op_i in situation j.

Let ${}_{(p)}P^k_{ij}$ represent Agent k's estimated probability that using an operator i in a state n that can be classified as situation j will lead to a final state, accumulated after p problem solving instances. Let $\mathcal{F}(\mathcal{T})$ be the set of states on the path to a terminal state T, and let $O_T \in \{0,1\}$ be the output of the terminal state T with 1 representing success and 0 a failure and let $0 \leq \alpha \leq 1$ be the learning rate. Then:

$$_{(p+1)}P^k_{ij} \;\; = \;\; (1-\alpha)_{(p)}P^k_{ij} + \alpha O_T, \;\; n \in \mathcal{F}(\mathcal{T}), \quad state \;\; n \; \in situation \; j$$

We will not dwell on the details of the Cost component update rule because the evaluation functions used in this work do not involve cost. In a design problem solving system, the computational costs are not a primary consideration. Successfully completing a good design takes precedence over computational costs involved as long as the costs are not widely disparate.

Obtaining measures of potential is a more involved process and requires a certain understanding of the system - at least to the extent of knowing which are the activities that can potentially make positive or negative contribution to progress of the problem solving process. For example, in L-TEAM, earlier on in a problem solving episode, the agents apply operators that lead to infeasible solutions due to conflicts in their requirements. However, this process of running into a conflict leads to certain important consequences like exchange constraints that were violated. The constraints an agent receives from other agents aid that agent's subsequent search in that episode by letting it relate its local solution requirements to more global requirements. Hence, the operators leading to conflicts followed by information exchange are rewarded by potential. Learning algorithms similar to that for utility can be used for learning the potential of an operator. Let ${}_{(p)}Pot^k_{ij}$ represent the predicted potential of the terminal state achieved by Agent k using an operator i in a state n which can be classified as situation j, accumulated after p problem solving instances. Let $\mathcal{F}(\mathcal{T})$ be the set of states on the path to the terminal state T, $Pot_T \in \{0,1\}$ be the potential arising from the state T, where Pot_T = 1 if there is is a conflict followed by information exchange else Pot_T = 0. Let 0 $\leq \alpha \leq$ 1 be the learning rate. Then:

$$_{(p+1)}Pot^k_{ij} \;\; = \;\; {}_{(p)}Pot^k_{ij} + \alpha\,(Pot_T - {}_{(p)}Pot^k_{ij}), \;\; n \in \mathcal{F}(\mathcal{T}), \;\; state \;\; n \; \in situation \; j$$

In the L-TEAM system, each operator is tagged with the result of its execution - either an added component to a partial design, or a conflict on certain local requirements along with the communicated violated local constraints, which can be used to determine the potential of a sequence of operators ending in a conflict.

4 Experimental Results

To demonstrate the effectiveness of the mechanisms in L-TEAM and compare them to those in TEAM, we used the same domain as in Lander[4] — parametric design of steam condensers. The prototype multi-agent system for this domain, built on top of the TEAM framework, consists of seven agents: pump-agent, heat-exchanger-agent, motor-agent, vbelt-agent, shaft-agent, platform-agent, and frequency-critic. The problem solving process starts by placing a problem specification on a central blackboard (BB). Problem specification consists of three parameters — required capacity, platform side length, and maximum platform deflection. During each cycle, each of the agents in L-TEAM can decide either to *initiate* a design based on the problem specification or *extend* a partial design on BB or to *critique* a partial design on BB. During the process of extending or critiquing a design, an agent can detect conflicts and communicate the cause of the conflict to other agents if it can articulate it. At present, an agent can communicate only single-clause numeric boundary constraints that are violated. If the receiving agent can understand the feedback (i.e. the parameter of the communicated constraint is in its vocabulary), it *assimilates* the information and uses it to constrain future searches. If such a conflict avoidance search does not work, an agent tries conflict resolution by resorting to *relaxing* soft constraints. In addition, if an agent detects stagnation in the progress of the local problem solving process, it relaxes the local quality requirement thresholds. The system terminates upon formation of a mutually acceptable design that satisfies the local quality thresholds of all the agents.

Each agent has an assigned organizational role in any single design. As mentioned before, TEAM identifies three organizational roles — initiate-design, extend-design, and critique-design. Learning the appropriate application of all these roles can be achieved but in this paper, we confine ourselves to two operators in each agent - initiate-design and extend-design. Four of the seven agents — pump-agent, motor-agent, heat-exchanger-agent, and vbelt-agent — are learning either to initiate a design or to extend an existing partial design in each situation. The other three agents have fixed organizational roles — platform and shaft agents always extend and frequency-critic always critiques.

In the experiments reported below, the situation vector for each agent had three components. The first component represented changes in the global views of any of the agents in the system. If any of the agents receives any new external constraints from other agents in the past m time units (m is set to 4 in the experiments[3]), this component is ‘1’ for all agents. Otherwise it is ‘0’. If any of the agents has relaxed its local quality requirements in the past n time units (n = 2) then the second component is ‘1’ for all agents. Otherwise it is ‘0’. Typically, a problem solving episode in L-TEAM starts with an initial phase of exchange of all the communicable information involved in conflicts and then enters a phase where the search is more informed and all the information that leads to conflicts and can be communicated has already been exchanged. During the initial phase of conflict detection and exchange of information, the third component is ‘0’. In the latter phase,

[3] m is empirically determined.

it is ‘1’[4] [5].

In design problem solving, the probability of successfully completing a design and obtaining a high utility design are of primary considerations. In addition, in a complex open environment like that in the L-TEAM system, some form of guidance to the problem solver regarding the intermediate stages of search that may have an indirect bearing on the final solution is helpful. So we used the following rating function:

$$f(U, P, C, potential) \quad = \quad U * P + potential$$

Learning rates were empirically determined and set to 0.1 for the Utility and Probability components and 0.01 for the Potential component.

We first trained L-TEAM on 150 randomly generated design requirements and then tested both L-TEAM and TEAM on the same 100 randomly generated design requirements different from those used for training. TEAM was setup so that heat-exchanger and pump agents could either initiate a design or extend a design whereas v-belt, shaft and platform agents could only extend a design. In TEAM, an agent initiates a design only if there are no partial designs on the blackboard that it can extend. We looked at two parameters of system performance. The primary parameter was the cost of the best design produced (lowest cost). The other parameter was the number of cycles the system went through to produce the best cost design. In TEAM (and L-TEAM) each agent in turn, gets a chance to perform an operation during a cycle. The number of cycles represents a good approximation to the amount of search performed by the entire system.

Figure 2 shows the difference in the costs of the best designs produced by L-TEAM and TEAM on each of the design requirements.

Average cost of a design produced by L-TEAM was 5587.6 and by TEAM was 5770.6. In the design domain, this difference of 3.2 % is considered a big win, especially because these designs may be mass-produced. Wilcoxon matched-pair signed-ranks test revealed that the cost of designs produced by L-TEAM was lower than those produced by TEAM at significance level 0.05 (p value was 0.00000003). Average cycles needed for L-TEAM to produce a design was 13.89, while TEAM needed 13.01 cycles. Table 1 shows the organizational roles for the TEAM system used in these tests. In L-TEAM, at the start of the learning phase, the four learning agents can either initiate a design or extend a design. Vbelt-agent and platform agents that can only extend a design and frequency-critic that can only critique a design are not learning. Learning leads to the organization shown in Table 2 (each cell shows the role with maximum rating for an agent in a given situation).

Next, we investigated the gain from situation specificity in L-TEAM. We ran L-TEAM with a null situation vector. Each agent could choose to either initiate a design or extend a design, independent of the situation. We trained the non-situation-specific L-TEAM to see if it could organize itself so that the agents played the roles that best suited them. The same 100 runs as used

[4] In the tables for organizational roles below, we represent a value of 1 for the first component as $newc$ whereas a value of 0 is $\neg newc$. A value of 1 for the second component is represented as rr whereas a value of 0 is $\neg rr$. A value of 1 for the third component is represented as sd whereas a value of 0 is $\neg sd$.

[5] In general, we expect that some characteristics of a partial design on which an operator is acting (like the goodness rating of the partial design) to also be a part of the situation vector. However, in L-TEAM we just consider learning two operators of which initiate-design initiates a solution independent of the existing designs. If there were other operators that interacted with extend-design based on the design which it extends, we might have to take these design characteristics into consideration as a part of the situation vector.

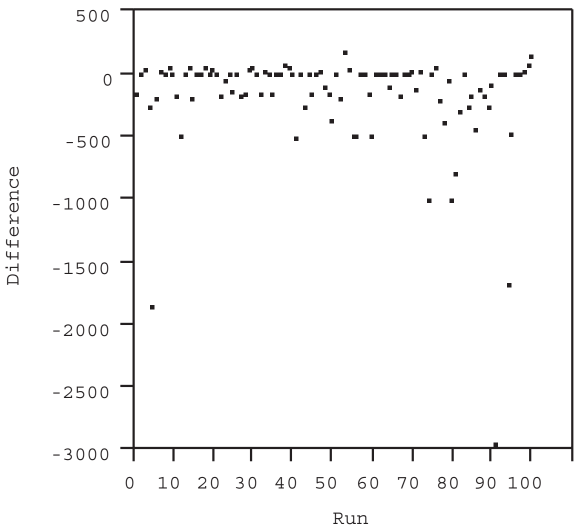

Figure 2: Y-axis shows the difference: lowest cost of design produced by situation-specific L-TEAM minus lowest cost of design produced by TEAM for the same design requirements. X-axis shows the run number.

agent	pump agent	heatx agent	motor agent	vbelt agent	shaft agent	platform agent	frequency critic
roles	initiate extend	initiate extend	extend	extend	extend	extend	critique

Table 1: Organizational roles for TEAM

	situation							
agent	$newc$ rr sd	$newc$ rr $\neg sd$	$newc$ $\neg rr$ sd	$newc$ $\neg rr$ $\neg sd$	$\neg newc$ rr sd	$\neg newc$ rr $\neg sd$	$\neg newc$ $\neg rr$ sd	$\neg newc$ $\neg rr$ $\neg sd$
pump agent	extend	initiate	initiate	extend	initiate	initiate	extend	initiate
heatx agent	initiate	extend	extend	extend	extend	extend	extend	extend
motor agent	extend	extend	extend	initiate	extend	initiate	extend	extend
vbelt agent	extend	extend	extend	extend	extend	extend	extend	extend
shaft agent	extend	extend	extend	extend	extend	extend	extend	extend
platform agent	extend	extend	extend	extend	extend	extend	extend	extend
frequency critic	critique	critique	critique	critique	critique	critique	critique	critique

Table 2: Organizational roles for situation specific L-TEAM after learning

agent	pump agent	heatx agent	motor agent	vbelt agent	shaft agent	platform agent	frequency critic
roles	initiate	extend	extend	extend	extend	extend	critique

Table 3: Organizational roles for non-situation-specific L-TEAM after learning

in the previous experiments were used for testing in this study. The average cost of the design produced by non-situation-specific L-TEAM was 5616.2 and by situation-specific L-TEAM was 5566.58. Wilcoxon matched-pair signed-ranks test revealed that the costs of designs produced by situation-specific L-TEAM were lower than those produced by non-situation-specific L-TEAM with a probability of 0.94 (p value was 0.06). This value is suggestive though not significant. Situation-specific L-TEAM was also better in terms of the number of cycles. Average cycles needed for situation-specific L-TEAM to produce a design was 13.89, while non-situation-specific L-TEAM needed 15.01. A Wilcoxon matched-pair signed-ranks test on these runs revealed that the number of cycles taken by situation specific L-TEAM to produce a design were lower than that taken by non-situation-specific L-TEAM at significance level 0.05 (p value was 0.049). Table 3 shows the learned organization. These experiments suggest (due to p value close to 0.05) that the situation specific L-TEAM is superior to non-situation specific L-TEAM that is superior to TEAM in terms of the cost of the designs produced. Situation-specific L-TEAM did a little more search than the TEAM system (difference of 0.88 cycles per design) but non-situation-specific L-TEAM did significantly worse than both situation-specific L-TEAM and TEAM in terms of the number of cycles.

At this point we could ask more detailed questions about why the situation-specific-L-TEAM perform better than the non-situation-specific-L-TEAM in terms of cost of designs? Even though pump-agent is the initiator of designs in an overwhelming number of cases, it turns out that the designs initiated by heat-exchanger-agent and motor-agent occasionally outperformed those initiated by the pump-agent. We have reasons to believe that a situation vector captures these subtleties, at least to a certain extent. In addition, it could also be the case that the initiations by motor-agent early on in the search led to a quicker discovery of conflicting requirements on shared parameters in certain problem runs. On rare occasions, (twice in our 100 experimental runs), situation-specific-L-TEAM performed worse than non-situation-specific-L-TEAM . We attribute this observation to the phenomenon of distraction frequently observed in multi-agent systems[5]. In the context of role assignments, this phenomenon maps to the ability of the agents to judge whether it is effective to work on its own designs or respond to the designs generated by the other members of the agent set in the present situation. It could be true that the situation vector we adopted may not have been sufficiently discriminating to eliminate such a distraction totally.

Next we investigated the role of the potential component in the evaluation function. We set up an experiment where situation-specific L-TEAM was trained with an evaluation function that did not take potential into consideration:

$$f(U, P, C, potential) \quad = \quad U * P$$

The system learned the organization shown in Table 4. Test runs on the same 100 problem specifications used for tests in the previous experiments showed that situation-specific L-TEAM without potential gave exactly the same results as non-situation-specific L-TEAM run-on-run. As can be seen from the previous discussion for experiments on non-situation-specific L-TEAM, situation-specific L-TEAM without potential produced designs of higher cost and required more number of cycles per design. This is not surprising given the organization in Table 4 is similar to that for non-situation-specific L-TEAM. Pump agent is always the initiator. Heat-exchanger agent and motor agent initiated designs in certain situations (bold-faced operators in Table 4) but these situations were the rarely occurring ones. The fact that potential leads to significant gains in the system performance brings us to an important observation. In complex systems like L-TEAM, it is often the case that the system performs actions that may only have an indirect bearing on the final solution requirements. Identifying such actions and rewarding the learning system for them can lead to an enhanced performance.

Agents in a multi-agent system are characterized by incomplete or inaccurate local views of the global problem solving process. Agents need to expend time and resources to communicate with other agents to obtain a more global view of the on-going problem solving process. This involves a trade-off where an agent has to distribute its limited resources between communication and assimilation of information received from other agents on one hand and its own local computational requirements on the other hand[3]. Given such a scenario, an agent may be forced to function in uncertain environments, where it may have an outdated or partial view of the global situation. We simulated such a divergence between an agent's subjective view of the global situation and the actual global situation by updating its subjective view every 3 cycles instead of every cycle. The organization learned by situation-specific L-TEAM as shown in Table 2 was used and during testing, an agent's view of the global situation vector was updated every three cycles instead of every cycle[6]. The system was tested on the same 100 problem specifications as in the previous experiments. The system performance deteriorated and it again performed identically to non-situation-specific L-TEAM. Situation-specific L-TEAM with updates for global situation vector

[6] The system may generally be permitted to spend more computational effort during learning and hence we use the knowledge learned with updates of situation vector every cycle.

	situation							
	$newc$	$newc$	$newc$	$newc$	$\neg newc$	$\neg newc$	$\neg newc$	$\neg newc$
	rr	rr	$\neg rr$	$\neg rr$	rr	rr	$\neg rr$	$\neg rr$
agent	sd	$\neg sd$	sd	$\neg sd$	sd	$\neg sd$	sd	$\neg sd$
pump agent	initiate	initiate	initiate	initiate	initiate	initiate	initiate	initiate
heatx agent	**initiate**	extend	extend	extend	extend	extend	extend	extend
motor agent	extend	extend	extend	extend	extend	**initiate**	extend	extend
vbelt agent	extend	extend	extend	extend	extend	extend	extend	extend
shaft agent	extend	extend	extend	extend	extend	extend	extend	extend
platform agent	extend	extend	extend	extend	extend	extend	extend	extend
frequency critic	critique	critique	critique	critique	critique	critique	critique	critique

Table 4: Organizational roles for situation specific L-TEAM with no potential component in the evaluation function

every three cycles produced designs of higher cost and required more number of cycles per design than situation-specific L-TEAM with updates for global situation every cycle.

Other experiments we conducted include investigating the generality of the learning mechanisms described here. We changed the problem specification to take one more component - the required power. Availability of this parameter permits pump and motor agents to perform more informed initiations of design. Situation-specific L-TEAM still produced an improvement of 41.39 units over TEAM in the cost of designs[6].

Performance of the L-TEAM and TEAM systems in the steam condenser domain relative to true optimality is not known but, as discussed in Lander[4], we suspect that there may be a floor effect in the data and the designs produced by the systems are close to optimal. This suggests that the gains from situation-specific learning that can possibly be achieved may be limited. If it is the case that such a floor effect exists, larger improvements can be expected in some other domains.

5 Related Work

Previous work related to learning in multi-agent systems is limited. Tan[16], Sandholm and Nagendra Prasad[8], Sandholm and Crites[7], and Sen and Sekaran[9] discuss multi-agent reinforcement learning systems. All these systems rely on reinforcement learning methods[1, 14]. While these works highlight interesting aspects of multi-agent learning systems, they are primarily centered around toy problems on a grid world. L-TEAM is one of the few complex multi-agent systems demonstrating the viability of such methods for interesting learning tasks in the domain of problem solving control, which is a notion that is not explicit in the above systems. Such an explicit notion of organizational control knowledge led us to studies on the importance of concepts like "potential" and divergence of subjective and objective views of the world.

Shoham and Tennenholtz[11] discuss co-learning and the emergence of conventions in multi-agent systems. However, the nature of the interactions is simple and the agents are fine-grained in their work. In this paper, we are concerned with large-grained agents and complex interactions between activities of the agents.

Shaw and Whinston[10] discuss a classifier system based multi-agent learning system that represents an interesting alternative to the learning mechanism we proposed in this paper but as discussed previously, in this paper our interest is more than just learning the organizational roles.

Sugawra and Lesser[15] discuss a distributed networking system where each local segment of the network has an intelligent diagnosis agent called LODES that monitors traffic on the network and uses an explanation-based learning technique to develop coordination rules for the LODES agents. TEAM-like systems may not be amenable to knowledge-intensive task of extracting coordination rules or situation-specific organizational rules from histories due to the heterogeneity of its agents.

Certain multi-agent learning systems in the literature deal with a different task from that presented in this paper. Systems like ILS[13] and MALE[12] use multi-agent techniques to build hybrid learners from multiple learning agents. On the other hand, L-TEAM learns problem-solving control for multi-agent systems.

6 Implications and Conclusion

Previous work in self-organization for efficient distributed search control has, for the most part, involved simple agents with simple interaction patterns. The work presented in this paper represents one of the few attempts at demonstrating the viability and utility of self-organization in an agent-based system involving complex interactions within the agent set.

L-TEAM is an example of an open system comprising reusable heterogeneous agents for parametric design. Agents in L-TEAM learn their organizational roles in a negotiated search for mutually acceptable designs. We tested the system on a steam condenser design domain and empirically demonstrated its usefulness. L-TEAM produced better results than its non-learning predecessor, TEAM, which required elaborate knowledge engineering to hand-code organizational roles for its agent set. However, the contributions of this paper go beyond just learning organizational roles. Experiments in the previous section taught us two important lessons with ramifications for issues of learning in multi-agent systems in general.

- It was noted that the performance was significantly better when an evaluation function took into consideration the potential of a role to make indirect contributions to the final solutions. In complex systems, recognition and exploitation of actions with potential can result in a better learning process. This observation encourages system designers to go beyond looking at the end result of a series of actions for credit-assignment schemes. They may also need to consider the role of meta-level information like relations of actions to the progress in the overall problem-solving process.

- Another important observation from the experiments in the previous section is concerned with the divergence of an agent's subjective view of the global situation from the actual global situation. In the L-TEAM system working on the steam condenser design, even a small divergence (update of the situation vector every three cycles instead of every cycle) led to significant deterioration in performance. However, in general, we expect the performance to decay gracefully with the divergence of the two views. In many real-life systems, especially in time-critical and resource-limited applications, this divergence may be inevitable and an agent has to be prepared to live with a trade-off between exploiting the knowledge that is learned and allocating resources to detect optimal instantiations of the learned knowledge.

References

[1] A. Barto, R. Sutton and C. Watkins, “Learning and Sequential Decision Making”, in M. Gariel and J. W. Moore (eds), *Learning and Computational Neuroscience*, MIT Press, Cambridge, MA 1990.

[2] D. D. Corkill, “A Framework for Organizational Self-design in Distributed Problem-solving Networks”, Ph.D. Dissertation, Dept. of Computer Science, University of Massachusetts, Amherst, 1983.

[3] E. H. Durfee and V. R. Lesser, “Predictability versus Responsiveness: Coordinating Problem Solvers in Dynamic Domains” in *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp 66-71, St. Paul, MN, 1988.

[4] S. E. Lander, *Distributed Search in Heterogeneous and Reusable Multi-Agent Systems*, Ph.D. Thesis, Dept.of Computer Science, University of Massachusetts, Amherst, 1993.

[5] V. R. Lesser, and L. D. Erman, “Distributed Interpretation: A Model and Experiment”, *IEEE Transactions on Computers*, C-29(12), pp 1144-1163, Dec. 1980.

[6] M V Nagendra Prasad, Victor Lesser, and Susan Lander, “Learning Organizational Roles in a Heterogeneous Multi-agent System” Forthcoming Technical Report, Department of Computer Science, University of Massachusetts, Amherst, 1994.

[7] T. Sandholm and R. Crites, “Multi-agent Reinforcement Learning in the Repeated Prisoner’s Dilemma” to appear in *Biosystems*, 1995.

[8] T. Sandholm and M V Nagendra Prasad, “MUSCLE: Multi-agent system for Coordinated Learning Experiments”, Unpublished memo, Department of Computer Science, University of Massachusetts, Amherst, 1993.

[9] S. Sen and M. Sekaran, “Learning to Coordinate without Sharing Information”, in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp 426-431, Washington, 1994.

[10] M. J. Shaw, and A. B. Whinston, “ Learning and Adaptation in DAI systems”, in *Distributed Artificial Intelligence*, vol. 2, Morgan Kaufmann Pub., San Mateo, California, pp. 413-429, 1989.

[11] Y. Shoham and M. Tennenholtz, “Emergent Conventions in Multi-Agent Systems: initial experimental results and observations”, in the Proceedings of KR-92.

[12] S. S. Sian, “Extending learning to multiple agents: issues and a model for multi-agent machine learning”, in *Machine Learning - EWSL 91*, Y. Kodratoff (Ed.), Springer-Verlag, pp. 440-456, 1991.

[13] B. Silver, W. Frawely, G. Iba, J. Vittal, and K. Bradford, “A framework for multi-paradigmatic learning”, in *Proceedings of the Seventh International Conference on Machine Learning*, Austin, Texas, pp 348-358, 1990.

[14] R. Sutton, “Learning to Predict by the Methods of temporal differences”, *Machine Learning* 3, pp 9-44, 1988.

[15] T. Sugawara, and V. R. Lesser, "On-Line Learning of Coordination Plans," Twelfth Annual Workshop on Distributed Artificial Intelligence, 1993.

[16] M. Tan, “Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents”, *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, Massachusetts, pp 330-337, 1993.

[17] R. Whitehair and V. R. Lesser, “A Framework for the Analysis of Sophisticated Control in Interpretation Systems”, Technical Report Number 93-53, Dept.of Computer Science, University of Massachusetts, Amherst, 1993.