Int. J. Human–Computer Studies (1998) **48,** 51–67

Learning organizational roles for negotiated search in a multiagent system

M. V. Nagendra Prasad and Victor R. Lesser
Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA. email: {nagendra,lesser}@cs.umass.edu

Susan E. Lander
Blackboard Technology Group, Inc., 401 Main Street, Amherst, MA 01002, USA. email: lander@bbtech.com

This paper presents studies in learning a form of organizational knowledge called organizational roles in a multi-agent agent system. It attempts to demonstrate 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. Introduction

Requirements like reusability of legacy systems and heterogeneity of agent representations lead to a number of challenging issues in multi-agent systems (MAS). Lander and Lesser (1994) developed the TEAM framework to examine some of these issues in heterogeneous reusable agents in the context of parametric design. TEAM is an open system assembled through minimally customized integration of a dynamically selected subset of a catalogue of existing agents. 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 that satisfy, as far as possible, the global considerations related to the overall problem. Reusable agents may be involved in system configurations and situations that may not have been explicitly anticipated at the time of their design. Adding a learning component to these agents so that they can modify their behavior based on the system configuration can lead to enhanced performance. In this paper, we present an extension of TEAM called L-TEAM that learns to organize itself to let the agents play the roles they are best suited for in such a multi-agent search process for constructing an overall solution.

The agents in TEAM (and L-TEAM) perform an asynchronous-distributed constraint-optimization search to obtain a good design. 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

1071-5819/98/010051 + 17$25.00/0/hc970160

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 Lander, 1994). For a composite solution in a given state, an agent can play one of a set of organizational roles (in TEAM, these roles are *solution-initiator*, *solution-extender* or *solution-critic*). An organizational role represents a set of tasks an agent can perform on a composite solution. An agent can be working on several composite solutions concurrently. Thus, at a given time, an agent is faced with the problem of (1) choosing which partial solution to work on; and (2) choosing a role from the set of allowed roles that it can play for that solution. This decision is complicated by the fact that an agent has to achieve this choice within its local view of the problem-solving situation (Lander, 1994).

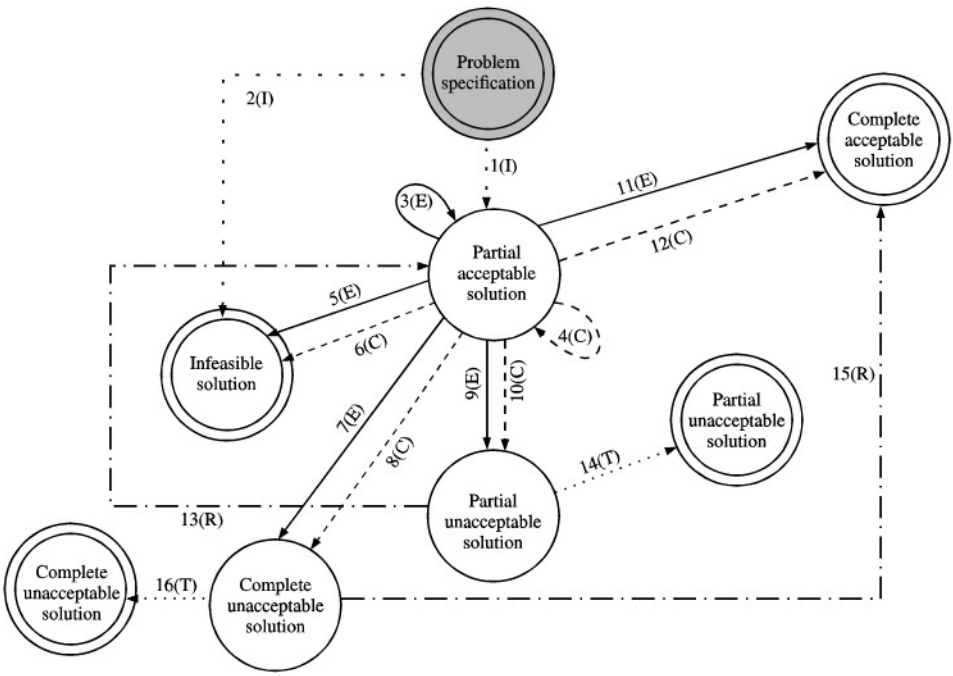

FIGURE 1. Negotiated search. · · · · · · Initiate solution (I); – – – – – – Critique solution (C); ——— Extend solution (E); — · — · — Relax solution (R); ····· Terminate search (T); ◎ Initial state; ◎ Termination state; ○ Intermediate state.

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 involving transmission of meta-level information.
2. Treating problem-solving control as situation-specific can be beneficial (discussed in detail in Section 3). In our case, situation-specific organizational roles led to better performance, especially in "harder" problems.

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 (Whitehair & Lesser, 1993) 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. Organizational roles in distributed search

Problem domains like those dealt with in TEAM can be viewed as comprising a set of interdependent subproblems. The overall solution to a problem is constructed by aggregation of solutions to each of the 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 to choose those actions that lead to generation of helpful constraints for 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.

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. It provides the agents a way to effectively and reliably handle cooperative tasks. Organizational roles represent a form of organizational knowledge that lets each agent in L-TEAM take part in the formation of a composite solution in a certain capacity. An organizational role is a task or a set of tasks to be performed in the context of a single solution. A role may encompass one or more operators, e.g. the role *solution-initiator* includes the operators *initiate-solution* and *relax-solution requirement*. 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 roles they are allotted. Organizational roles played by the

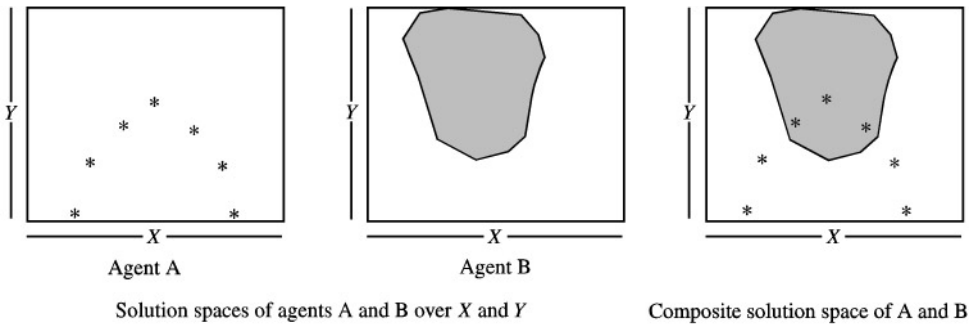

FIGURE 2. Local and composite search spaces.

agents are important for the efficiency of a search process and the quality of final solutions produced.

To illustrate the above issue, we will use a simple, generic two-agent example. Figure 2 shows their search and solution spaces. The shaded portions in the local spaces of Agents A and B are the local solution spaces and their intersection represents the global solution space. It is clear that if Agent A initiates and Agent B extends, there is a greater chance of finding a mutually acceptable solution. Agent A trying to extend a solution initiated by Agent B is likely to lead to a failure more often than not due to the small intersection space vs the large local solution space in Agent B. Note, however, that the solution distribution in the space is not known *a priori* to the designer to hand-code good organizational roles at the design time.

During each cycle of operator application in TEAM, each agent decides on the role it can play next based on the available partial designs. An agent can choose to be an *initiator* of a new design or an *extender* of an already existing partial design or a *critic* of an existing design. The agent needs to decide on the best role to assume next and accordingly construct a design component. This paper investigates the effectiveness of learning situation-specific organizational role assignments. No single organizational role assignment may be good for all situations. The agents adapt themselves to play roles that are better suited for the current problem-solving situation, so as to be effective (we will discuss situations in more detail in the following section).

3. Learning role assignments

Learning involves exploring the space of role assignments, i.e. developing rating measures for roles in various situations. The formal basis for learning role assignments is derived from the UPC formalism for search control (see Whitehair & Lesser, 1993) that relies on the calculation and use of the utility, probability and cost (UPC) values associated with each $\langle state, R, final\_state \rangle$ tuple. Utility represents an agent's estimate of the final state's expected value or utility if it takes on role R in the present state. Probability represents the expected uncertainty associated with the ability to reach the final state from the present state, given that the agent plays role R. Cost represents the expected computational cost 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 a role to a state does more than expand it. The role application may result in an increase in the problem solver's understanding of the interrelationships among states. In these situations, a role 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 a role is referred to as its *potential* and it needs to be taken into account while rating the role. An evaluation function defines the objective strategy of the problem-solving system based on the UPC components of a role 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. An agent applies the evaluation function to all the roles applicable at the present state of the on-going search and a role that maximizes the ratings of all applicable roles is selected.

Starting from this core of the 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 pre-enumerated finite classes 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 role 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 an agent, it 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 choosing a role indexes into a database of UPC values using the situation vector to obtain the relevant UPC values for the roles applicable in the current state. Depending on the objective function to be maximized, these UPC vectors are used to choose a role to be played 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 aproach* to prediction learning (see Sutton, 1988) to learn estimates for the UPC vectors for each of the situations. The agents collectively explore the space of possible role assignments to identify good role assignments in each of the situations. The role assignment at a particular agent is affected by the state of problem solving at the other agents and also the nature of the non-local search spaces. At each agent, 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

† Even though we describe L-TEAM as an off-line learning system, we could make it on-line by letting the system heuristically identify a point at which it could stop learning and switch to choosing roles to maximize their evaluations rather than choosing them probabilistically in ratio of their ratings.

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 agents then back-trace through these steps assigning credit for the performance to the roles involved (the exact process will be described below).† At each agent, values of the UPC vector for the role corresponding to the situation at that agent are adjusted. In our use of the UPC framework, we assume that there is a single final state—the generation of a complete design mutually acceptable to all the agents.

Let $\{S_j^k\}$, $1 \leq j \leq M_k$, 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 R_i^k, $1 \leq i \leq N_k$, be the set of roles an Agent k can play in a composite solution. Agent k has $M_k * N_k$ vectors of UPC and Potential (abbreviated as *Pot*) values: $\{R_i^k, S_j^k,$ *Agent k*, $U_{ij}^k, P_{ij}^k, C_{ij}^k, Pot_{ij}^k\}$. Given a situation S_b^t, objective function $f(U, P, C, Pot)$ is used to select a role R_a^k such that

$$Prob(R_a^k) = \frac{f(U_{ab}^k, P_{ab}^k, C_{ab}^k, Pot_{ab}^k)}{\sum_i f(U_{ib}^k, P_{ib}^k, C_{ib}^k, Pot_{ib}^k)} \quad \text{during learning,}$$

$$R_a^k = {}^k f_R^{-1} \max_i f(U_{ib}^k, P_{ib}^k, C_{ib}^k, Pot_{ib}^k) \quad \text{after learning,}$$

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

Let $\mathscr{T}$ be the distributed search tree where each node is annotated with the triple $\{R_i^k, S_j^k, A_k\}$ representing the role R_i^k played by Agent k in situation S_j^k (see Figure 3). Let $\mathscr{F}(\mathscr{T})$ 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 roles played by the agent set lead to a terminal state—say a success:

$$[\{R_1, S_1, A1\}, \{R_2, S_2, A2\}, \ldots, \{R_m, S_m, Am\}].$$

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–role 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 ${}_{(p)}U_{ij}^k$ represent the predicted utility of the final solution achieved by Agent k playing role R_i in a state n that can be classified as situation j, accumulated after p problem-solving instances. Let $\mathscr{F}(\mathscr{T})$ to be set of states on the path to a final state F. U_F represents the utility of the solution

† 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 (Sutton, 1988). 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.

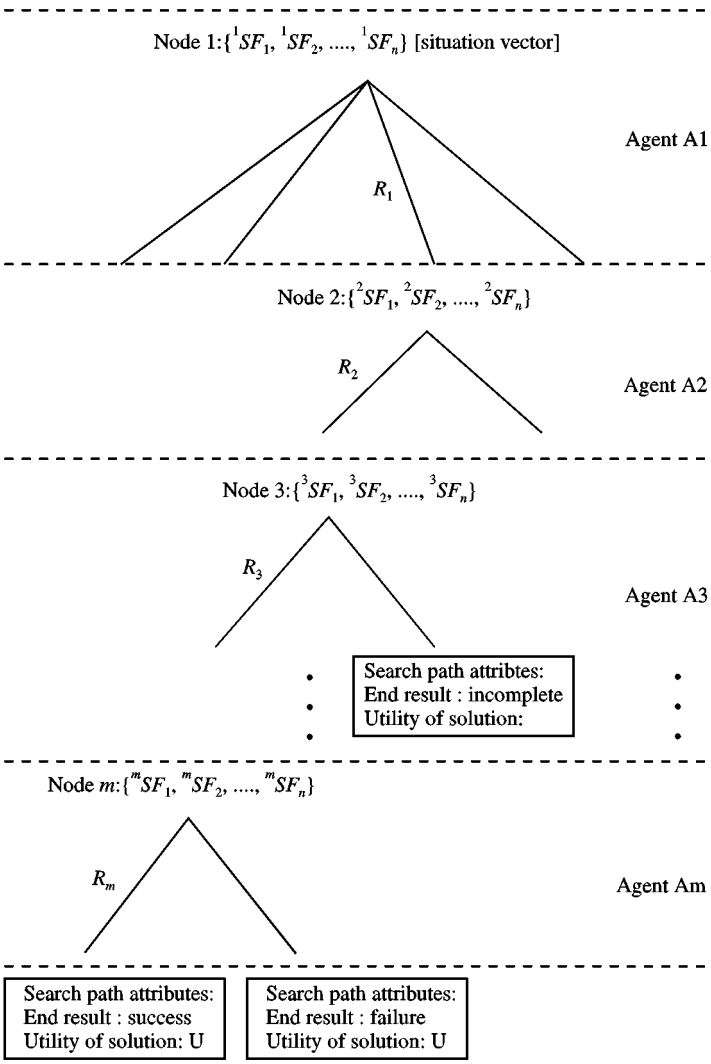

FIGURE 3. Distributed search over the space of possible role assignments.

and $0 \leq \alpha \leq 1$ is the learning rate. Then

$$ {}_{(p+1)}U_{ij}^k = {}_{(p)}U_{ij}^k + \alpha(U_F - {}_{(p)}U_{ij}^k),\ n \in \mathscr{F}(\mathscr{T}), \quad \text{state } n \in \text{situation } j. $$

Thus, Agent k, that played role R_i, modifies the Utility for its R_i in situation j.

Let ${}_{(p)}P_{ij}^k$ represent Agent k's estimated probability that playing role R_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 $\mathscr{F}(\mathscr{T})$ be the set of states on the path to a terminal state T. $O_T \in \{0, 1\}$ is the output of the terminal state T with 1 representing success and

0 a failure. $0 \leq \alpha \leq 1$ is the learning rate. Then

$$_{(p+1)}P_{ij}^{k} = (1-\alpha)_{(p)}P_{ij}^{k} + \alpha O_T, \quad n \in \mathscr{F}(\mathscr{T}), \quad \text{state } n \in \text{situation } j.$$

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 take on roles 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 of 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 roles 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 a role. Let $_{(p)}Pot_{ij}^{k}$ represent the predicted potential of the terminal state achieved by Agent k playing role R_i in a state n which can be classified as situation j, accumulated after p problem-solving instances. Let $\mathscr{F}(\mathscr{T})$ 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 a conflict followed by information exchange, else $Pot_T = 0$. Let $0 \leq \alpha \leq 1$ be the learning rate. Then

$$_{(p+1)}Pot_{ij}^{k} = {}_{(p)}Pot_{ij}^{k} + \alpha(Pot_T - {}_{(p)}Pot_{ij}^{k}), \; n \in \mathscr{F}(\mathscr{T}), \quad \text{state } n \in \text{situation } j.$$

In the L-TEAM system, each role 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 roles 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 (1994)—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, v-belt-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 the BB or to *critique* a partial design on the 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 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 roles in each agent: initiate-design and extend-design. Four of the seven agents—pump-agent, motor-agent, heat-exchanger-agent and v-belt-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. When an agent decides to extend or critique, it chooses the best partial design on the blackboard (i.e. lowest cost) and plays the role of extending or critiquing for that design.

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 was 4 in the experiments), 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". Receiving a new external constraint or relaxing local quality requirements change the nature of the local search space of an agent. This could prompt it to initiate designs to seed the blackboard with partial designs that take these constraints into consideration. Typically, a problem-solving episode in L-TEAM starts with an initial phase of generating seed designs, followed by a phase of exchange of all the communicable information involved in conflicts and then 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, the third component is "1". During the intermediate phase of conflict detection and exchange of information, the third component is "2". In the final phase, it is "3". During the initial phase, some agents may often play the role of initiators of designs so as to lead to discovery of conflicting requirements that can be exchanged during the intermediate phase to enhance each of the agents' view of the global requirements on its local search. It is important to note that these features are based on the negotiated search mechanisms rather than the underlying steam condenser domain. They are generic to the domain of parametric design that TEAM addresses.†

† While an understanding of the negotiated search is needed to get these features, we believe that it involves much lesser effort than identifying the exact nature of interactions in a steam condenser domain, that a human expert needs to know before she can assign good roles. For example, in TEAM, the human need to know the nature of the constraints in pump-agent, motor-agent, heat-exchanger-agent before she could assign roles to the agents.

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:

$$f(U, P.C, potential) = U * P + Potential.$$

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 set up so that heat-exchanger and pump-agents could either initiate a design or extend a design, whereas v-belt, shaft- and platform-agent 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 play a role in an evolving composite solution during a cycle. The number of cycles represents a good approximation of the amount of search performed by the entire system. It is important to note that design cost is different from search cost. Design cost is a representation of the solution quality, while the search cost is a representation of the computational cost of the design process. Even though the computational cost is not a consideration in the credibility function for chosing a role in design problem-solving, it is interesting to observe these measures as they are representative of the search efficiency of the underlying problem-solving process.

We ran L-TEAM and TEAM in two ranges of the input parameters. Range 1 consisted of required capacity 50–1500, platform-side length 25–225, platform deflection 0.02–0.1. Range 2 consisted of required capacity 1750–2000, platform-side length 175–225, platform deflection 0.06–0.1. Lower values of required capacity in Range 1 represented easier problems. We choose the two ranges to represent "easy" and "tough" problems. One can see from Tables 4 and 5 that the two learned organizations for Ranges 1 and 2 are different. In order to understand the contribution of situation specificity, we also set up L-TEAM to learn organizational roles in a situation-independent manner. Non-situation-specific TEAM learns the same organization, as shown in Table 6, over both the ranges. Table 1 shows the average design costs for the three systems—situation-specific L-TEAM (ss-L-TEAM), non-situation-specific L-TEAM (ns-L-TEAM) and TEAM—over the two ranges. Table 2 shows the average number of cycles per design for the three systems—ss-L-TEAM, ns-L-TEAM and TEAM.

Wilcoxon matched-pair signed-ranks test revealed significant differences (at significance level 0.05) between the cost of designs produced by all the pairs in the table except between situation-specific L-TEAM and non-situation-specific L-TEAM in Range 1 †

† Easy problems may not gain by sophisticated mechanisms like situation specificity. One interesting future research direction involves meta-learning of the difficulty of ranges of problems based on which different role tables can be selected.

TABLE 1
Average cost of a design

Range	ss-L-TEAM	ns-L-TEAM	TEAM
Range 1	5603.2	5616.2	5770.6
Range 2	17353.75	17678.97	17704.70

TABLE 2
Average cycles per design

Range	ss-L-TEAM	ns-L-TEAM	TEAM
Range 1	13.52	15.01	13.01
Range 2	15.0	21.0	15.0

TABLE 3
Organizational roles for TEAM

Agent	Pump -agent	Heat -agent	Motor- agent	V-belt- agent	Shaft- agent	Platform- agent	Frequency critic
Roles	Initiate Extend	Initiate Extend	Extend	Extend	Extend	Extend	Critique

(Table 3) and between non-situation-specific L-TEAM and TEAM in Range 2. The same test also revealed no significant differences between the number of cycles per design for situation-specific L-TEAM and TEAM over both the ranges, while showing significance in differences between the number of cycles per design for non-situation-specific L-TEAM and each of the other two systems over both the ranges.

These experiments suggest 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 but non-situation-specific L-TEAM did significantly worse than both situation-specific L-TEAM and TEAM in terms of the number of cycles (see Tables 4–6).

At this point we could ask more detailed questions about why the situation-specific-L-TEAM performs better than the non-situation-specific-L-TEAM in terms of cost of designs? It turns out that the pump-agent has a functional relationship between its parameters *water-flow rate* and *head*. This relationship, which constrains the set of acceptable solutions, cannot be communicated due to the restrictions on the representation of communicable information in TEAM; only single-clause numerical constraints can be communicated. Thus, as discussed in Section 2, it may be best that the pump-agent initiates a design because such a design then will have captured the relationship between the above two parameters. Even though pump-agent is the initiator of designs in

TABLE 4
Organizational roles learned by situation-specific L-TEAM for Range 1

	Situation								
	1	1	0	0	1	1	0	0	0
	1	0	1	0	1	0	1	0	0
Agent	3	3	3	3	2	2	2	2	1
Pump-agent	Initiate	Extend	Initiate	Initiate	Initiate	Extend	Initiate	Initiate	Initiate
Heat-x-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Initiate
Motor-agent	Extend	Extend	Extend	Extend	Extend	Extend	Initiate	Extend	Initiate
V-belt-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Shaft-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Platform-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Frequency critic	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique

TABLE 5
Organizational roles learned by situation-specific L-TEAM for Range 2

	Situation								
	1	1	0	0	1	1	0	0	0
	1	0	1	0	1	0	1	0	0
Agent	3	3	3	3	2	2	2	2	1
Pump-agent	Initiate	Extend	Extend	Extend	Initiate	Initiate	Initiate	Extend	Initiate
Heat-x-agent	Extend	Initiate	Extend	Extend	Extend	Extend	Extend	Extend	Initiate
Motor-agent	Extend	Extend	Extend	Extend	Extend	Extend	Initiate	Extend	Initiate
V-belt-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Shaft-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Platform-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Frequency critic	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique

TABLE 6
Organizational roles for non-situation-specific L-TEAM after learning

Agent	Pump-agent	Heat-x-agent	Motor-agent	V-belt-agent	Shaft-agent	Platform-agent	Frequency critic
Roles	Initiate	Extend	Extend	Extend	Extend	Extend	Critique

TABLE 7
Results for ss-L-TEAM without potential

	ss-L-TEAM with potential		ss-L-TEAM withoutpotential	
Range	Cost	Cycles	Cost	Cycles
Range 1	5603.2	13.52	5647.3	15.17
Range 2	17353.75	15.0	18105.56	25.88

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 a few occasions, 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 (Lesser & Erman, 1980). 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:

$$f(U, P, C, potential) = U * P.$$

The system learned the organizations shown in Tables 8 and 9. The boxed entries are default values. The agent never played these roles in the corresponding situation in any successful solution. So the system does not learn to discriminate between the roles. We just let the agent choose the role that it plays in most other situations as the default. The system was tested on the same 100 problem specifications used for tests in the previous experiments. Table 7 shows the results.

TABLE 8

Organizational roles in Range 1 for situation-specific L-TEAM with no potential

	Situation								
	1	1	0	0	1	1	0	0	0
	1	0	1	0	1	0	1	0	0
Agent	3	3	3	3	2	2	2	2	1
Pump-agent	Initiate	Initiate	Initiate	Initiate	Initiate	Initiate	Initiate	Initiate	Extend
Heat-x-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Motor-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Initiate	Initiate
V-belt-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Shaft-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Platform-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Frequency critic	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique

TABLE 9

Organizational roles in Range 2 for situation-specific L-TEAM with no potential

	Situation								
	1	1	0	0	1	1	0	0	0
	1	0	1	0	1	0	1	0	0
Agent	3	3	3	3	2	2	2	2	1
Pump-agent	Initiate	Extend	Initiate	Extend	Initiate	Initiate	Initiate	Initiate	Initiate
Heat-x-agent	Extend	Initiate	Extend	Initiate	Extend	Extend	Extend	Extend	Extend
Motor-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
V-belt-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Shaft-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Platform-agent	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend	Extend
Frequency critic	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique	Critique

In Range 1, ss-L-TEAM with no potential performs similar to non-situation-specific L-TEAM. This is not surprising because the organization learned by L-TEAM with no potential in Range 1 is similar to that for non-situation-specific L-TEAM, i.e. pump-agent is almost always the initiator. Motor-agent initiated designs in certain situations, but these situations were the rarely occurring ones. In Range 2, the organization learned by ss-L-TEAM with no potential is different from ss-L-TEAM with potential and it performs significantly worse than non-situation-specific L-TEAM with potential and situation-specific L-TEAM with potential.

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 (Tables 8 and 9).

5. Related Work

Previous work related to learning in multi-agent systems is limited. Tan (1993), Sandholm and Nagendra Prasad (1993), Sandholm and Crites (1995) and Sen, Sekaran and Hale (1994) discuss multi-agent reinforcement learning systems. All these systems rely on reinforcement learning methods (Sutton, 1988; A. Barto & Watkins, 1990). While these works highlight interesting aspects of multi-agent learning systems, they are primarily centered around toy problems on a grid world. Even though we do not deny the importance of such studies to a nascent field like learning in multi-agent systems, learning in complex systems can provide many challenges and interesting insights that may not be forthcoming in simple toy domains or homogeneous agent systems. L-TEAM is one of the few multi-agent systems demonstrating the viability of learning problem-solving control for realistic and complex domains. We believe that importance of concepts like "potential" become more apparent in such domains. Mataric (1994) discusses the concept of progress estimators akin to the idea of potential. Potential differs from progress estimators in that the latter was primarily used as a method speeding up reinforcement learning, whereas the former plays a more complex role. In L-TEAM, the concept of potential leads to different organizations and better quality results and is not a just a speedup device.

A related work using classifier systems for learning suitable multi-agent organizations is presented in Weiss (1994). Multiple agents use a variant of Holland's (1985) bucket brigade algorithm to learn appropriate instantiations of hierarchical organizations. Though Weiss (1994) studies this system in the blocks world domain, it could represent an interesting alternative to the learning mechanism we proposed in this paper for learning organizational knowledge.

Nagendra Prasad, Lesser and Lander (1996) and Garland and Alterman (1996) discuss issues in knowledge reuse in multi-agent systems. Sugawra and Lesser (1993) 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. Unlike these systems, TEAM-like systems may not be amenable to the know-

ledge-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 (Silver, Frawely, Iba, Vittal & Bradford, 1990) and MALE (Sian, 1991) 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 important lessons with ramifications for issues of learning in multi-agent systems in general.

- Different situations need different kinds of organizations in multi-agent systems. While this is not a new observation, our work takes this insight a step further and proposes exploiting learning techniques to provide multi-agent systems with non-situation-specific organizational knowledge. The need for such techniques becomes especially acute in multi-agent systems constructed form a set of reusable agents.
- Moreover, the role organizations produced by learning is different for the two different ranges implying that different nature of the problems being solved by the system may need different role organizations.
- 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.

This material is based upon work supported by the National Science Foundation under Grant Nos IRI-9523419 and EEC-9209623. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

A. BARTO, R. S. & WATKINS, C. (1990). Learning and sequential decision making. In M. GARIEL & J. W. MOORE, Eds. *Learning and Computational Neuroscience*. Cambridge, MA: MIT Press.

GARLAND, A. & ALTERMAN, R. (1996). Multi-agent learning through collective memory. In *Proceedings of the AAAI Spring Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*. Stanford, CA.

HOLLAND, J. H. (1985). Properties of bucket brigade algorithm. In *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, pp. 1–7. Pittsburgh, PA.

LANDER, S. E. (1994). *Distributed search in heterogeneous and reusable multi-agent systems*. Ph.D. Thesis, Dept. of Computer Science, University of Massachusetts, Amherst.

LANDER, S. E. & LESSER, V. R. (1994). *Sharing meta-information to guide cooperative search among heterogeneous reusable agents*. Computer Science Technical Report 94-48, University of Massachusetts. IEEE Transactions on Knowledge and Data Engineering, (to appear).

LESSER, V. R. & ERMAN, L. D. (1980). Distributed interpretation: a model and an experiment. *IEEE Transactions on Computers*, **C-29,** 1144–1163.

MATARIC, M. J. (1994). Reward functions for accelerated learning. In *Proceedings of the 11th International Conference on Machine Learning*, San Francisco, CA.

NAGENDRA PRASAD, M. V., LESSER, V. R. & LANDER, S. E. (1996). Retrieval and reasoning in distributed case bases. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, **7,** 74–87.

SANDHOLM, T. & CRITES, R. (to appear). Multi-agent reinforcement learning in the repeated prisoner's dilemma. *Biosystems*.

SANDHOLM, T. & NAGENDRA PRASAD, M. V. (1993). Muscle: multi-agent system for coordinated learning experiments. Unpublished working paper.

SEN, S., SEKARAN, M. & HALE, J. (1994). Learning to coordinate without sharing information. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pp. 426–431. Seattle, WA. New York: AAAI.

SIAN, S. S. (1991). Extending learning to multiple agents: issues and a model for multi-agent machine learning. In *Proceedings of Machine Learning—EWSL 91*, pp. 440–456. Berlin: Springer.

SILVER, B., FRAWELY, W., IBA, G., VITTAL, J. & BRADFORD, K. (1990). A framework for multi-paradigmatic learning. In *Proceedings of the 7th International Conference on Machine Learning*, pp. 348–358.

SUGAWARA, T. LESSER, V. R. (1993). On-line learning of coordination plans. In *Procedings of the 12th International Workshop on Distributed AI*, Hidden Valley, PA.

SUTTON, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, **3,** 9–44.

TAN, M. (1993). Multi-agent reinforcement learning: independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning*, pp. 330–337.

WEISS, G. (1994). *Some studies in distributed machine learning and organizational design*. Technical Report FKI-189-94, Institut für Informatik, TU München.

WHITEHAIR, R. & LESSER, V. R. (1993). *A framework for the analysis of sophisticated control in interpretation systems*. Computer Science Technical Report 93–53, University of Massachusetts.