A Distributed Problem-Solving Approach to Rule Induction: Learning in Distributed Artificial Intelligence Systems

Michael J. Shaw*
Riyaz Sikora**

CMU-RI-TR-90-28

*Robotics Institute, Carnegie Mellon University; on leave from The Beckman Institute, University of Illinois at Urbana-Champaign
** The Beckman Institute and Department of Business Administration, University of Illinois at Urbana-Champaign

November, 1990

Contents

List of Figures

List of Tables

Abstract

One of the interesting characteristics of multi-agent problem solving in distributed artificial intelligence(DAI) systems is that the agents are able to learn from each other, thereby facilitating the problem-solving process and enhancing the quality of the solution generated. This paper aims at studying the multi-agent learning mechanism involved in a specific group learning situation: the induction of concepts from training examples. Based on the mechanism, a distributed problem-solving approach to inductive learning, referred to as DLS, is developed and analyzed. This approach not only provides a method for solving the inductive learning problem in a distributed fashion, it also helps shed light on the essential elements contributing to multi-agent learning in DAI systems. An empirical study is used to evaluate the efficacy of DLS for rule induction as well as its performance patterns in relation to various group parameters. The ensuing analysis helps form a model for characterizing multi-agent learning.

1. Introduction

One of the interesting characteristics of multi-agent problem solving in a DAI system is that the agents are able to learn from each other, thereby facilitating the problem-solving process and enhancing the quality of the solution generated. By learning we mean an agent can improve its problem-solving performance through acquiring new knowledge, refining existing knowledge, using better strategies, or applying cases previously proven to be successful. In addition, the agents in a DAI system can learn from each other through interactions among the group members. These interactions may play a crucial role in guiding the multi-agent problem-solving process. Understanding how multi-agent problem solving is affected by the learning behaviors of the agents can lead to more effective design of DAI systems.

This paper aims at studying the multi-agent learning mechanisms involved in a specific group learning situation: the induction of concepts from training examples. It is an extension of the inductive learning process for rule generation, which is one of the most developed areas in machine learning(Michalski[1983], Rendell[1990]). By extending this rule-induction procedure to multi-agent environments, our objective is to gain more insights with respect to the group learning strategies and mechanisms that can be used in DAI systems.

The multi-agent version of the inductive learning procedure can be viewed as a distributed problem-solving approach(Decker[1987], Durfee et al.[1989]) to learning from examples. This approach not only provides a method for solving the inductive learning problem in a distributed fashion, it also help shed light on the essential elements contributing to multi-agent learning in a group problem-solving situation.

Understanding how the agents learn in group problem-solving situations can lead to valuable insight into how the nodes in a DAI system can continuously learn and refine its problem-solving capabilities. For example, we can use such knowledge to develop a network of cooperating expert systems(Shaw et al.[1990, 1991]) in which the nodes are capable of adaptively learning from each other, or a group of robot controllers that can learn to perform assembly operations in a concerted fashion. Moreover, the methodology also can be used for knowledge acquisition among a group of human experts, from whom the disparate sets of knowledge acquired can be systematically synthesized into a more coherent body of knowledge that integrates the individual experts' views.

The contributions of the resulting methodology are thus two-fold: it provides a new learning method for rule induction based on distributed problem solving; and it also provides a model for analyzing multi-agent learning. Such a methodology exemplifies the efficacy of DAI, as it manifests two of the reasons for applying DAI: to understand the interactions between multiple agents necessary for group problem solving, and to solve problems too large for centralized systems(Huhns[1987], pp.v - vi)

The distributed problem-solving approach consists of four steps: problem decomposition, task assignment, local problem solving, and, finally, solution synthesis. When applying it to inductive learning, the four steps correspond to: 1. decomposing the set of training examples into several subsets, 2. assigning each subset of training examples to an agent, 3. letting each agent solve the learning problem assigned, and 4. synthesizing the solutions from the agents into the group solution. The group solution is the concept descriptions, represented in disjunctive normal form, underlying all the training examples. An example of such a concept-learning problem is to learn the rules for predicting whether a company would go bankrupt; the training set in this case consists of historical financial data corresponding to both financially healthy companies and bankrupt companies.

The methodology incorporates various mechanisms in the four steps for developing the collective learning methodology. In the decomposition phase in step 1, the subsets of training data must be as representative as possible of the whole instance space; to achieve this, a statistical sampling technique called "jackknife" is used to extract each subset. In the local problem-solving phase, the probabilistic learning system, PLS, is used to derive concept descriptions for each subset of training examples. Lastly, in the final synthesis step, the group solution is synthesized through another step of induction; in our methodology, this synthesis of partial solutions submitted by individual agents into group solution is achieved by applying the genetic algorithm(GA), which gives interesting insight into the necessary operations for group synthesis.

An empirical study based on two rule-learning applications is used to analyze the learning characteristics of the distributed problem-solving approach, in comparison with the conventional single-agent approach to inductive learning. This analysis also helps us identify the factors contributing to the improved performance resulting from the distributed problem-solving approach to inductive learning. Specifically, the following characteristics of the system are shown to affect the learning performance:

- the diversity of the agents' views and knowledge,
- the ability to generate hypotheses[1] by the agents,
- the quality of the solutions submitted by the agents,
- the decomposition method used,
- the parallel pursuit of multiple search paths for learning, and
- the ability of performing synthesis to integrate the solutions submitted by the individual agents.

[1]A clarification of terms. In an inductive learning process, an agent would keep generating *hypotheses* about the correct *concept description* for the inductive rules. These hypothese can be viewed as various ideas about the solution. In group problem solving, and learning, the agents would share these hypotheses and treat them as the partial solutions for a group solution.

The remainder of the paper is organized as follows: Section 2 reviews group problem solving, the taxonomy, and the possible learning processes in DAI systems. Section 3 describes rules induction in multi-agent systems in general. In Section 4, the distributed problem-solving approach to inductive learning is described. Section 5 elaborates on the importance of the synthesis step on distributed learning. Incorporating decomposition and diversity as problem-solving strategies is discussed in Section 6. In Section 7, an empirical study for analyzing the collective leaning scheme is described, its findings discussed. Finally, Section 8 concludes the paper.

2. Distributed Problem Solving and Learning in DAI Systems

A DAI system consists of a group of intelligent problem-solving nodes, which will be referred to as *agents* in this paper. An example of such a system is a network of rule-based expert systems, or a group of intelligent controllers(Gasser and Huhns[1989]). There are two different types of DAI systems that have been developed:

(1) *Collaborative reasoning systems.* The agents in this type of systems would be solving the *same* problem collaboratively. The main issue here is not the decomposition into sub-problems assigned to the agents; rather, the focus is usually put on guiding and coordinating the interactions among the participating agents, so that the problem can be solved jointly by the group *simultaneously.* The PLEXYS system(Nunamaker et al.[1988]), COLAB(Stefik et al.[1987]), and the concurrent design system described in Bond[1989] all fall into this category.

(2) *Distributed problem-solving systems.* In these systems, the overall problem to be solved is decomposed into sub-problems assigned to the agents, each agent, *asynchronously,* would plan its own actions and turn in its solutions to be synthesized with the solutions of other agents. The agents in these systems use either *task sharing*(e.g.,Smith[1980]) or *data sharing*(e.g., Lesser&Corkill[1981]) to cooperate with other agents. The office information system described in Woo&Lachovsky[1986] and the scheduling system describe in Shaw&Whinston[1988, 1989] are examples of these systems.

The distributed problem-solving approach described in this paper is based on the task sharing method for distributed problem-solving systems, as articulated in Davis and Smith[1983]. In a DAI system, the agents, the way problems are solved, and the strategies used for getting the solutions all can be designed in different ways. Shaw[1990a,b] identifies the following dimensions of DAI systems that must be considered to make the group problem-solving activities effective.

(1) Goal Identification and task assignment;
(2) Distribution of knowledge;
(3) Organization of the agents;
(4) Coordination mechanisms; and
(5) Learning schemes.

There are two different types of problem-solving processes that can be used by the group of agents: the first type of processes are used in collaborative reasoning systems where a problem is presented to the whole group and the agents would collectively carry out the deliberation process - usually consisting of issues identification, proposing solutions, discussion, prioritizing, and finalizing the group solutions. The second type of processes are used in distributed problem solving systems where the common strategy used consists of four steps: *problem decomposition, task assignment, local problem solving, and solution synthesis.* In the latter type of DAI systems, negotiation is widely used for task assignment (Durfee&Lesser[1989], Laasri et al.[1990], Sathi&Fox[1989], Sycara[1989]).

The knowledge possessed by the group of agents can be distributed in different fashions. There may be different degree of redundancy in the agents' knowledge bases. In one extreme, the group of agents have exactly the same knowledge; in the other extreme, each agent in the group possesses completely different areas of knowledge(Weihmayer[1990]). As will be discussed in detail, we found that for a multi-agent learning system to perform rule induction efficiently, it is best to maintain the greatest degree of *diversity* possible.

The organizational structure of the agents would determine the amount of information processed and the coordination necessary for the agents to operate efficiently. Malone[1987] evaluated a variety of organizational structures and compared them on three dimensions: amount of processing needed, amount of coordination required, and degree of vulnerability. A widely used structure in DAI systems is the market structure, such as contract-net system(Smith and Davis[1981], Parunak[1987]). Fox[1981] articulated the two opposing forces of task complexity and uncertainty, and related them to the issue of bounded rationality of the agents in an organization(or a computer system). Cohen[1986], on the other hand, underscored the importance of considering the transitory nature of organizational decision making; he argued for incorporating flexibility and the dynamic performance of the agents in an organization. This school of thought has led to the recent emphasis on learning in organizational decision making(Ching et al.[1990]).

Coordination is necessary in DAI systems for resolving conflicts, allocating limited resources, reconciling different preferences, and searching in a global space for solutions based on local information(Durfee&Mongomery[1990]). Coordination mechanisms can be based on a variety of information passed among the agents, such as data, new facts just generated, partial solutions/plans, preferences, and constraints. A number of coordination mechanisms have been developed for DAI systems, each of which with its unique protocol for determining the timing of activities, triggering of events, action sequences, and message content in the coordination process. The role of coordination in DAI systems is discussed in Shaw[1990b].

The focus of this paper, the learning mechanism in a DAI system should be an integral part of the problem-solving activities for improving the task allocation, knowledge, coordination and organization involved. There are several learning processes that can be incorporated in DAI

systems. It can be in the form of data exchange, knowledge transfer, or heuristics migration, where the learning mechanisms involved are relatively simple. It can also be done by extending machine learning techniques developed for single-agent systems, such as explanation-based learning, case-based reasoning, or inductive learning, to the multi-agent systems, where one agent can learn by observing and interacting with other agents. In the organization context, the learning processes interact with the dynamic performance of the agents. Lounamaa and March[1987] showed how the learning effects can be affected by coordination among the agents. Of particular interests are the use of various forms of group-interaction processes, such as group induction, nominal group techniques, or brain storming, for achieving the learning effects among the whole group(Kraemer and King[1988]). These group processes use structured sessions of information exchange to develop new concepts, which would lead to solutions not attainable by any of the agents alone. They are also good examples for illustrating the complementary role played by the problem-solving and learning activities.

In a DAI system, learning may occur in two ways: the agents can learn as a group, while at the same time, each agent can also learn on its own by adjusting its views and actions(Shaw&Whinston[1989]). Among the group problem-solving activities in a DAI system, learning takes effect in the form of (1) better coordination, (2) more efficient task and resource allocation, and (3) more effective organization. The improved coordination can be achieved by information sharing, knowledge sharing, or more efficient communications among the agents. Whereas the task and resource allocation process can be improved by learning the specialization(i.e., knowledge distribution) of the agents(e.g., agent x is good at performing task A), by learning the group characteristics(e.g., agents y and z work well as a team), by learning the patterns of tasks(e.g., for a given type of problem, the past experience indicated that it is easier to break the problem into two tasks, C and D, and do D first), and finally, by learning such environmental characteristics as user preferences, processor reliability, or future jobs. In his society-of-mind model, Minsky[1985] articulated the learning of "administrative ways" in group problem solving, which essentially includes the learning of coordination, task allocation, and organizational structures among the agents.

The machine learning literature to date has primarily focused on learning processes of a single agent. How a problem-solving agent can learn and adapt by interacting with other agents in a DAI system posts an important research question. This paper attempts to address this by investigating the rule induction mechanisms in multi-agent problem solving.

3. Rule Induction and Its Extension to Multi-Agent Systems

Concept learning is one of the most developed areas in machine learning, where the objective is to derive concept descriptions, through rule induction, to satisfactorily describe and explain a set of training examples given(Quinlan[1986]). Starting with the given set of training examples, the learning process employs a series of generalization and specialization steps to search through

the space of all possible concept descriptions; this process continues until the concept descriptions that satisfy all the descriptions of the training examples are found(Michalski[1983]), Shaw[1987]). The whole process consists of the following main steps:

Algorithm 3.1: Concept-Learning;
Input: a set of training examples $\Psi = \{E(t) \mid t = 1,...,T\}$;
Output: learned hypothesis Ω satisfying all the training examples;
Begin
(0) initialization: $P = \Phi$, $Q = \Psi$;
(1) read in the next uncovered training example from Q, add it to the set P;
(2) generating a new hypothesis H about the possible concept descriptions based on P;
(3) evaluating H by matching it with the examples in Q ;
(4) remove all the examples in Q satisfied by H and put them in P;
(4) if $Q = \Phi$, then Ω = H and stop; otherwise. go to (1);
End

In any machine learning program, there are two major components of the program that are crucial to the success of the learning performance. First, the proper mechanism for evaluating the hypotheses. This is called the *credit assignment* function. Second, the proper mechanism for generating new hypotheses based on existing ones. This is called the *hypotheses transformation* function. These two functions are the key to all the machine learning programs developed. In the Concept-Learning Algorithm described above, step (2) is for hypotheses transformation and step(3) is for credit assignment. Since this type of programs attempt to derive descriptions that can explain the given set of observations, or examples, the method is referred to as *inductive learning*.

To extend such a learning procedure to the multi-agent environment of a DAI system, additional *coordination* would be needed so that both credit-assignment and hypotheses-transformation functions can be performed in a globally conherent fashion.

A direct extension of the concept-learning algorithm for single agent is the *group induction* process. In group induction, the agents of the system engage in collecting evidence, generating hypotheses, and evaluating hypotheses in order to discover new concepts collectively. Each agent makes its individual observations, forms hypotheses based on the observations, and keeps searching for new evidence for modifying the hypotheses. As more observations are made, evidence may confirm or disconfirm hypotheses, thus strengthening or weakening the agents' beliefs about the hypotheses. This process continues until new concepts are derived that are supported by the observations and the agents' beliefs. The crucial design in such a group induction process is the *coordination mechanism* incorporated for the agents to interact with each other. Proper coordination can greatly accelerate the learning process by having the agents share their beliefs as well as the newly generated hypotheses that appear to be successful.

Since there have been little work done on developing group induction algorithms in the machine learning literature, a possible source of inspiration comes from the psychology literature where researchers have been conducting behavioral experiments to study group induction. For example, Laughlin&Shippy[1983] described an experiment on group induction to address the issue whether " a cooperative group is able to induce a general principle that none of the group members could have induced alone, or the group merely adopt an induction proposed by one or more of the group members."

The experiment conducted was to ask a group of human subjects to identify the rule used in partitioning decks of bridge playing cards. The group induction procedure begin with a known positive example of the rule. The objective is to ask the group to induce the rule in as few trials as possible. A trial consists of the following five stages: First, each individual group member wrote down an hypothesis (proposed induction). Second, the group members discussed until they reach a consensus on a group hypothesis. Third, the group members discussed until they reached a consensus on a play of any of the cards, which was shown to the experimenter. Fourth, the experimenter classified this card as either a positive or a negative example. Fifth, the experimenter checked the group hypothesis. if the group hypothesis was correct, the problem was solved; if the group hypothesis was incorrect, a new cycle of the five stages followed, consisting of a second cycle of individual hypothesis, group hypothesis, experiment with a play of a card, feedback on the status of the card, and check of the group hypothesis as correct or incorrect. The process terminated when a correct group hypothesis was found.

Laughlin&Shippy[1983] concluded that the group were remarkably successful in recognizing and adopting a correct hypothesis if the hypothesis was proposed by one or more members. In contrast, group induction in the strong sense that a correct group hypothesis was derived that none of the group members had proposed individually was extremely rare. Their findings can be summarized by the following propositions. (1) Group induction is advantageous over single-agent induction primarily due to the agents' sharing of the larger set of hypothesis and other information, not necessary because of that the group together can generate new hypothesis not derivable by the individual members; and (2) In the group induction process, the sharing of hypothesis and other information among the agents would result in the identification of correct group hypothesis in fewer iterations.

The first proposition attempts to identify the key element of group induction; the second proposition attempts to explain the underlying reason for the phenomenon. Together, based on the evidence presented by the experiment, these two propositions point out the importance of *hypotheses generation* and *information sharing* in a group induction process. Designing algorithms for achieving the same learning effects, therefore, requires the incorporation of effective mechanisms that would facilitate information sharing. The basic group induction process can be represented algorithmically as follows:

Algorithm 3.2: Group-Induction

Input: a set of training examples $\Psi = \{E(t) \mid t = 1,...,T\}$

a group of N problem-solving agents;

Output: learned concept description Ω generated by the group;

Begin

(0) Let P = Φ, Q = Ψ;

(1) The first training example E(i) in Q is presented to the group;

(2) P = P + {E(i)} and Q = Q - {E(i)};

(3) For each agent j, j = 1... N, a new hypothesis H(j)
is generated to satisfy all the examples in P;

(4) For each agent j, j = 1...N, the agent's hypothesis is broadcast to the whole group;

(5) The whole set of hypotheses Ω = H(1) v H(2) v...v H(N) are evaluated against training examples in Q;

(6) Those examples in Q satisfied by Ω are removed from Q and put in P;

(7) If Q = Φ, stop; otherwise go to step (1).

End

Note that in this group induction algorithm, the learning process is carried out by the set of agents collectively. This algorithm has the flavor of a parallel algorithm in steps (3) and (4) when the operations are performed on the set of agents concurrently. The major benefit of having the group of agents is their sharing of the newly generated hypothesis in each iteration, which increases the probability that a correct hypothesis can be identified sooner. In this particular algorithm, information sharing is achieved by a simple broadcasting mechanism. In a DAI system, since each AI nodes, a problem-solving agent, has a different knowledge base, the rules they use in generating new hypothesis would be different. Information sharing thus would provide greater probability to identify the correct hypothesis for concept descriptions sooner.

This group induction algorithm is primarily designed to capture the characteristics of group problem solving in the process of inductive learning. By incorporating the collective induction process used in the experiment described in Laughlin&Shippy[1983], this algorithm models the learning behavior of a group of human decision makers, each of whom has different domain knowledge, and would generate different H(j). It is this variation in the H(j)'s - i.e., the diversity - that provides the major advantage for group induction.

We can extend these views and make the conjecture that group problem solving can find solution quicker because of information sharing. New facts, hypothesis, and partial solutions generated by one agent in its problem-solving are shared by other agents. These additional pieces of information shared among the agents would trigger the set of agent to new problem-solving and evidence-gathering activities which would not have been performed without the agents' interactions. In turn, these triggered activities would generate new facts, hypothesis, and evidence that are shared among the agents. Thus, the amount of information shared in the agents' problem-solving efforts grows exponentially. And this gives the group an advantage to reach a

solution much sooner. On the other hand, the exponentially growing information would mean substantially more searching efforts in screening through all the information generated. A properly designed *coordination mechanism* can lead the group to evaluate all the relevant facts, hypothesis, new evidence, and partial solutions more systematically.

The primary objective of coordination, therefore, is for focusing the agents' attention on the relevant information. It also can be used to constrain the search space engaged by the group. The Delphi method is an example for such a group problem-solving method incorporating a structured coordination mechanism. Linston&Turoff[1975] define the Delphi technique as " a method for structuring a group communication process so that the process is effective in allowing a group of individuals, as a whole, to deal with a complex problem." In the Delphi method there are four phases to the group's interaction process: 1. Exploration of the subject to be discussed by the group; 2. reach understanding of how the group views the issues; 3. if disagreements exist, explore reasons and evaluate; 4. when all information has been analyzed and evaluations have been fed back for consideration, make a final evaluation. To accomplish structured communication, the coordination mechanism underlying the Delphi technique needs to provide: some exchange of individual contributions of information and knowledge; some assessment of the group judgement and view; and some opportunity for individuals to revise views(Linston&Turoff[1975]). This four-stage Delphi procedure is very similar to the aforementioned group-induction process in that the emphasis of group interactions is placed on collecting ideas and sharing information. This similarity also underscores the tight relationship between group problem solving and group learning. Instead of using collaborative reasoning such as the Delphi procedure, we can also use distributed problem-solving mechanisms - such as task sharing - for achieving multi-agent learning. This will be discussed next.

4. A Distributed Problem-Solving Approach to Inductive Learning

As mentioned earlier, the Group-Induction algorithm is suitable for the learning situations where a group of heterogeneous agents are participating in the induction process. Alternatively, the learning process can further be facilitated by properly *decomposing* the problem and letting each agent handle a subset of the training examples. In this case, the group learning process would achieve greater speed-up than the afore-mentioned Group-Induction Algorithm because each agent is solving a learning problem with smaller size and reduced complexity. The algorithm can be described as follows:

Algorithm 4.1 Cooperative-Induction
Input: a set of training examples $\Psi = \{E(t) \mid t = 1,...,T\}$;
a group of N problem-solving agents, π;
Output: learned hypothesis Ω generated by the group;
Begin

(0) Decompose Ψ into N subsets, $\Psi(1)$, $\Psi(2)$, ...$\Psi(N)$, where $\Psi = \Psi(1) \vee \Psi(2) \vee ... \vee \Psi(N)$;

(1) *Initiation*: The set of actively learning agents $\mathcal{A} = \pi$, the set of finished agent $\mathcal{F} = \Phi$, $P(j) = \Phi$, $Q(j) = \Psi(j)$;

(2) For each agent j in $\mathcal{A}$, $P(j) = P(j) + \{E(i)\}$ and $Q(j) = Q(j) - \{E(i)\}$, where E(i) is the first example in Q(j);

(3) For each agent j in $\mathcal{A}$, a new hypothesis H(j) is generated to satisfy all the examples in P(j);

(4) For each agent j in $\mathcal{A}$, H(j) is evaluated against the training examples in Q(j), and those examples in Q(j) satisfied by H(j) are removed from Q(j) and put in P(j);

(5) For each agent j in $\mathcal{A}$, if $Q(j) = \Phi$, remove j from $\mathcal{A}$ and put it in the set $\mathcal{F}$;

(6) If $\mathcal{A} = \Phi$, go to (7); otherwise, go to (2);

(7) $\Omega = H(1) \vee H(2) \vee ... \vee H(N)$

End;

This algorithm applies distributed problem solving to the induction process. The cooperation strategy incorporated is rather like the one used in the Contract-net approach developed by Smith[1980] - that is, it uses the *task-sharing* approach to achieve cooperative problem solving. Smith[1980] characterized the task-sharing strategy as consisting of the following four steps:

(1) problem decomposition;
(2) task assignment;
(3) local problem solving by the agents involved; and
(4) solution synthesis.

Cooperative-Induction Algorithm achieves these steps by (1) decomposing the training examples into N subsets; (2) assigning a subset of training examples to each agent; (3) letting each agent perform inductive learning on the subset of training data assigned to it; and (4) synthesizing the concept descriptions generated by the agents to be the group concept description(by taking the union).

Although the Cooperative-Induction Algorithm achieves greater speed-up than the Group-Induction Algorithm due to the use of task sharing, there are, however, two major problems. First, the agents do not take advantage of the information learned by other agents. Each of them just goes about solving its assigned sub-problem and then synthesize its results with those of the others to obtain the group solution. Second, the final synthesis step for combining all the learning results is rather primitive in that it just takes the disjunction of the concept descriptions generated by all the agents. It is quite likely that the synthesized concept description Ω would still be too long a description to be the induction result.

An improved algorithm will be described shortly that will let the agents share their partial learning results. Moreover, an additional step would be used to synthesize the hypotheses(i.e.,

concept descriptions) generated by the agents. The critical step there, as will be explained, is to use a procedure for *synthesizing* the learning results of the individual agents.

This synthesis step can be incorporated by the genetic algorithms(Holland[1975], Holland et al.[1986], Goldberg[1989]). The genetic adaptation plan is performed on the group of agents to refine the knowledge (that is, knowledge about solving the problem as well as group meta-knowledge) and its distribution.The transformation of agents' knowledge is achieved by such genetic operators as reproduction, crossover, and mutation. The Appendix describes the basic genetic adaptation procedure using GA. Section 5 provides a detailed analysis on the synthesis step and its incorporation of GA.

To incorporate a group-synthesis mechanism in the Cooperative-Induction Algorithm(4.1), we algorithmically combine the Cooperative-Inductive process and GA. To begin with, we can have a group of agents sharing the tasks involved in solving the rule-learning problem by decomposing the learning problem into sub-problems. Each agent would solve the sub-problem, generating the concept descriptions based on the subset of training examples, and then apply the genetic adaption plan. There are many merits in combining the two approaches, among them are the speed-up due to the use of multiple agents, reduced complexity of the learning processes because of the use of decomposition, the abilities of the genetic adaptation scheme to transform concept-descriptions in forming more concise descriptions, and the more globally oriented search achieved by genetic adaptation(Sikora&Shaw[1990a,b]). The procedure can be described by the following DLS (Distributed Learning System) Algorithm:

Algorithm 4.2: DLS

Input: a set of training examples $\Psi = \{E(t) \mid t = 1,...,T\}$;
a group of N problem-solving agents,π;

Output: concept-description Ω learned by the group;

Begin

(0) Decompose Ψ into N subsets, $\Psi(1)$, $\Psi(2)$, ...$\Psi(N)$, where $\Psi = \Psi(1) \vee \Psi(2) \vee ... \vee \Psi(N)$;

(1) For each agent j in π, execute the Concept-Learning Algorithm(Algorithm 5.1) on $\Psi(j)$ and obtain the concept description $\xi(j)$;

(2) Execute the Genetic Adaptation algorithm(Appendix) by using $\{\xi(j) \mid j = 1,..., N\}$ as the initial population;

(3) Ω = the concept description generated by the Genetic Adaptation algorithms;

End

This multi-agent induction process is depicted in Figure 4.1. Sikora and Shaw[1990] describes the implementation of the DLS Algorithm. The decomposition in step(0) is achieved by incorporating a *statistical sampling technique* called the *jackknife* technique[2], which randomly

select a fraction of the training examples $\Psi(j)$ for each agent j. For step(1), the *probabilistic learning system(PLS)* developed by Rendell[1983, 1990] is used for concept induction. By applying the genetic adaptation algorithm in step(2), Sikora and Shaw show that the learning performance of DLS is much improved over any of the single-agent learning algorithm.

The success of the DLS Algorithm highlights the efficacy of the distributed problem-solving approach to rule induction. The approach in its essence can be described in a more generic fashion, in which the procedure consists of four phases: I. problem decomposition, II. task assignment, III rule induction from the subset of examples allocated to each agent, and IV. concept synthesis. This procedure is illustrated in Figure 4.1.

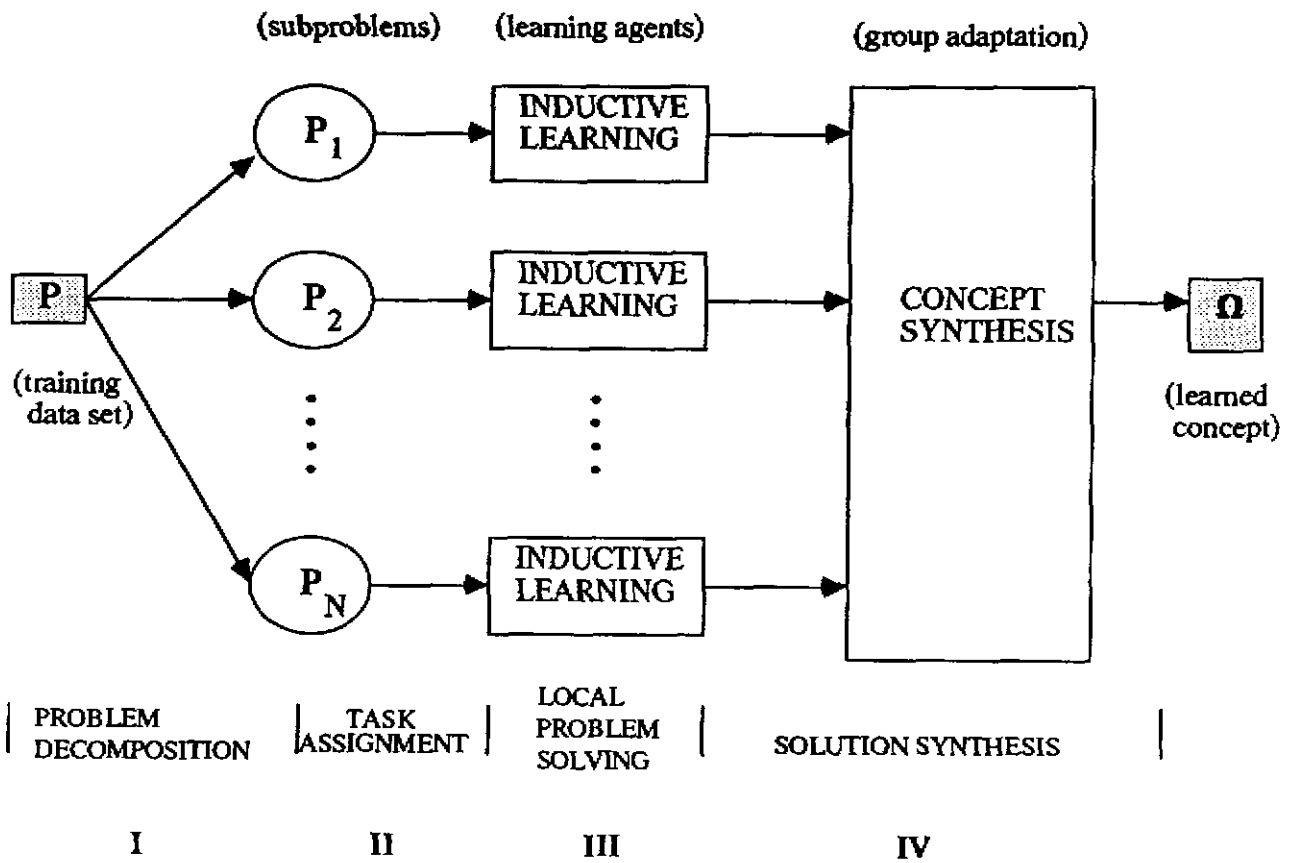

Figure 4.1 DLS: The Distributed Problem-Solving Approach to Inductive Learning

5. Synthesizing the Learning Results

[2]The jackknife technique selects a random sub-sample which is (1 - r) of the original data set , where r is the portion of the data set that will be left out, $0 \leq r < 1$.

Algorithms 3.2, 4.1, and 4.2 represent three different types of group learning processes in multi-agent problem-solving situations. They differ in the ways decomposition and synthesis are incorporated. Let's call them types A, B, and C of group learning for ease of reference.

Type A process models after the collective induction process described in Laughlin&Shippy [1983], as discussed in Section 3. The group of agents are presented with the same examples, simultaneously, and the group solution to the learning problem is produced through an iterative process of refining the hypotheses generated by the group members. This type A process for group learning is often used in collaborative reasoning systems described in Section 2.

Type B process applies distributed problem solving by assigning a subset of the training examples to each agents. The learning results from the agents are collected and together they form the group solution(mathematically, in *disjunctive* form) without any additional efforts to summarize- i.e., synthesize- them into more concise form. This group learning process is not a complete one.

Type C process, which is used in our methodology, applies the same problem decomposition, task assignment, and local problem-solving procedures as Type B, except that it incorporates a more elaborate step in synthesizing the partial solutions submitted by the individual agents. This synthesis step involves collecting the partial solutions, evaluating their pros and cons, and coming out with a group solution that keeps the desirable components(i.e., good ideas), perhaps adding in some variations for possible improvements, and taking out the unfavorable components. This synthesis step is highlighted by Figure 5.1.

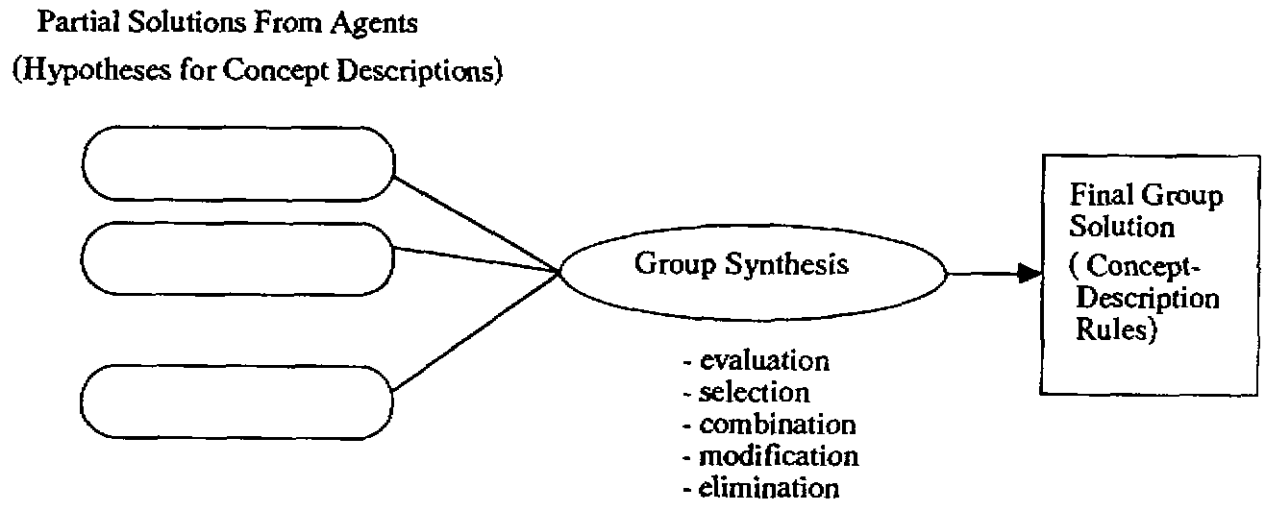

Figure 5.1 The Synthesis Step in Distributed Problem Solving and Multi-Agent Learning

The group synthesis itself can be viewed both as an *induction* operation and as a *collective evaluation* process conducted by the group. This step can be summarized as follows: Group synthesis is the step that collects the partial solutions from the individual agents, evaluating these

partial solutions, and deriving the final group solution based on these partial solutions. In other words, the partial solutions represent *hypotheses* proposed by the individual agents. The synthesis step evaluates these hypotheses and search the hypothesis space to derive the final solution.

In the context of mathematical logic, each partial solution given by agent i - which can be viewed as a hypothesis Γ^i, is in the following disjunctive normal form(DNF):

$$\Gamma^i = \Gamma^i_1 \vee \Gamma^i_2 \vee \ldots \vee \Gamma^i_p$$
$$= (\xi_{1,1} \& \ldots \& \xi_{k_1,1}) \vee (\xi_{1,2} \& \ldots \& \xi_{k_2,2}) \vee \ldots \vee (\xi_{1,p} \& \ldots \& \xi_{k_p,p}),$$

The synthesis step is to derive the final solution out of the set of Γ^1, Γ^2, ... up to Γ^N, N is total number of agents.

Let's use an example to illustrate what the DLS algorithm achieves in this synthesis step. Say a manager wants to apply distributed problem solving and uses a group of agents to perform the task of developing a proposal. He asks 10 different people to submit several preliminary proposals, each with k sections. Using the above formulation, each proposal prepared by an agent corresponds to a Γ^i_j. Each person i can submit p different versions of the proposals. Out of these Nxp proposals, the manager then needs to *synthesize* them into the final proposal. The way this can be done is to have several iterations of evaluation, modification, and combination. In each iteration, the manager evaluates the proposals, giving a score to each. Based on the scores, he then combines the features of the better ones and sometime switches sections between two good proposals. He would stop this process until a satisfactory proposal emerges.

Likewise, the synthesis operations in multi-agent learning should be an iterative process that includes evaluation, selection, combination, modification, and merging. This iterative process of synthesizing is essentially what the genetic algorithm in our DLS algorithm would help achieve. Taking the partial solutions from the agents as hypotheses, it evaluates each individual hypothesis, assigns a fitness score to each, and generates a new set of hypotheses based on such transformation operators as reproduction, crossover, and mutation. The process stops when a satisfactory concept description emerges from the hypotheses.

The fitness function is used to evaluate the hypothesis. The reproduction, crossover, and mutation operators can achieve transformational effects similar to selection, combination, modification, and merging. The fact that we can apply the genetic algorithm for this synthesis step is interesting and merits further elaboration. At first glance, genetic algorithms seem to imply very specific representation(i.e., genes) and transformation(i.e., the genetic operators). But if we take a closer look, the functional requirements of this synthesis step consist of "...collecting the partial solutions, evaluating their pros and cons, and coming out with a group solution that keeps the desirable components(i.e., good ideas), perhaps adding in some variations for possible improvements, and taking out the unfavorable components". These operations are quite

consistent with the activities performed by a genetic algorithm, namely, to evaluate the fitness of the individual components, and transform them into new forms which have more desirable features.

In terms of knowledge representation, solution synthesis using the genetic algorithm fits perfectly into the mathematical representation used by single-agent inductive learning as well. Using PLS, each agent's partial learning results can be represented by a DNF, which is a disjunction of terms Γ^i_j representing the jth hypotheses generated by agent i, i = 1, ...,N, and j = 1,..., p. Each Γ^i_j is a conjunct represented as $(\xi_{1,j} \& ... \& \xi_{k_j,j})$. The genetic algorithm takes the pool of all the terms in these DNFs and treats them as the initial population for further induction. As a result, the initial population of hypotheses serving as input to the synthesis step has Nxp members. These hypotheses will then be used as the basis for getting the group solution through evaluation, selection, combination, merging, and modification, as performed by the genetic algorithm.

6. Distributed Problem Solving Strategies: Decomposition & Diversification

There are a number of reasons for taking the distributed approach to problem solving, among them are two fundamental ones: first, the problem at hand may be too large and complex for a single agent to solve the whole thing at once; second, the chance of successfully getting the solution can be improved by searching through multiple paths for the solution. These two strategies - identified as decomposition and diversification - have been used previously for problem solving. A closer look at them would help identify the general issues involved in the distributed problem-solving approach.

The decomposition used in distributed problem solving basically is a divide-and-conquer method to break the problem up into simpler sub-problems to tackle. In a similar vein, it has been used quite intensively in solving large-scale optimization problems. Interestingly, just as the way we aim at gaining insight into multi-agent problem-solving systems through the distributed approach, the decomposition approaches to solving optimization problems are, besides being used as a solution strategy for solving large-scale mathematical programming problems(Dantzig& Wolfe[1961], Lasdon[1968]), also used to model decision making within decentralized organizations. On one hand it helps solve difficult problems by transforming them into several sub-problems which have simpler structures and are easier to solve; on the other hand, the decomposition process and the way partial solutions are transferred correspond nicely to the way problems are solved in a decentralized organization, in which the individual departments would be assigned with tasks/resources and coordinated by a pricing mechanism called transfer prices(Burton et al.[1974]). The optimization problems based on mathematical programming can search through variables represented in the primal or the dual problems. In a similar fashion, the

rule-learning problems can be represented in the instance-space(I-space) or the hypothesis-space(H-space). In DLS, the agents perform learning in the I-space and the group synthesis step is performed in the H-space. The symmetrical relationships between the decompositional approach to optimization and the distributed approach to rule induction are summarized in Table 6.1. By looking at the similarities of the two methodologies, we hope to take advantages of the experiences gained in the research on distributed decision making and coordination in decentralized organizations. It also helps highlight the dual purposes of DLS serving both as a solution method and as a model for multi-agent learning.

Problem	Math Programming	Rule Induction
Function of the Model	Organization Optimization	Group Learning
Master Prob./Search Space	Organization-Level/ Dual	Group -Level/ H-Space
Sub-Prob./ Search Space	Department-Level/ Primal	Agent-level/ I-Space
Constraints for Allocation	Resources	Training Examples
Coordination	Dual Prices	Fitness Measures
Objective	Max. Organizational Profit	Max. Concept Accuracy

Table 6.1 The Structural Correspondence Between the Decompositional Approach to Optimization and the Distributed Problem-Solving Approach to Rule Induction

One of the advantages of taking the distributed problem-solving approach, as in a human group situation, is that more diverse viewpoints and sources of ideas are present in guiding the problem-solving activities, thereby increasing the chance of success. An interesting source of literature related to the analysis on the diversity among a group of individuals and its impacts on group performance comes from the study on strategies for biological evolution. Brady[1985], for instance, showed that strategies used in biological evolution - such as speciation, having a large population and helping weaker individuals to survive - can also be used as strategies for solving optimization problems. Brady also articulated the importance of *diversification* in solving optimization problems such as the traveling salesman problem. He showed that the performance can be improved simply by dividing computer time equally between two independent trial solutions(i.e., two agents) and select the best solution. This strategy can be generalized into the N-agent case - according to Brady's analysis, even though these agents have to share the computation time and resources, they still result in better performance on balance. This finding is quite consistent with our empirical results reported in Section 7. Kuhn[1970] noted the correlation between the diversity of theories existing in a society and the speed of its scientific progress; this finding is somewhat similar to the result of our analysis on multi-agent learning in DAI systems, which is not surprising since Kornfeld&Hewitt[1981] already pointed out the

scientific community as a metaphor for DAI systems.

Several parameters need to be taken into account to make DLS effective, such as the decomposition ratio, diversity of the learning agents, number of learning agents used, and the pool of hypotheses generated. These issues will be addressed by the following empirical analysis.

7. Empirical Study and Analysis With Learning Applications

This section will evaluate DLS as a rule-learning method, the performance patterns of the distributed problem-solving approach, and the use of DLS to characterize multi-agent learning. Two concept leaning problems were used to evaluate the DLS algorithm. One is bankruptcy prediction based on financial data; the other is a process control problem. Both can be stated as the problem of generating classification rules from a set of training examples. The group performance in each case is evaluated by the prediction accuracy of the rules it generates on the companion set of training examples. To average out the random errors, in each case five different pairs of training sets and testing sets were used and the results are reported using the averages of the 5 trials.

The DLS algorithm based on multi-agent learning is implemented on a Texas Instruments Explorer, in Common Lisp. The bankruptcy-prediction application has a data set consisting of 58 training examples with two classes. This data set, described in detail in Shaw and Gentry[1989], corresponds to a set of financial data profiling companies that went bankrupt and companies that have been financial healthy. The task of learning from this data set is to derive concept-descriptions - i.e., rules - characterizing the financial profiles of bankrupt companies. The classifications involved, therefore, consists of bankrupt (i.e., positive examples) and healthy (i.e., negative examples) companies.

The data set is divided into 10 sub-sets by the jackknife sampling technique, to be assigned to 10 *learning agents*. Each agent learns the classification concept based on the training examples assigned to it by applying the PLS procedure. The PLS procedure generates concept descriptions in disjunctive normal form. Each of the individual disjuncts is treated as a single member in forming the initial population for synthesis applying the genetic algorithm. Among the 10 agents, a total of 42 disjuncts are generated, which are then treated as the *initial population* for genetic adaptation. The probability of crossover is set at 0.7 and the probability of mutation is set at 0.01. Figure 7.1 shows the *learning curve* - the convergence of the maximum level of fitness of the population - of the DLS procedure vs. when GA is applied alone.

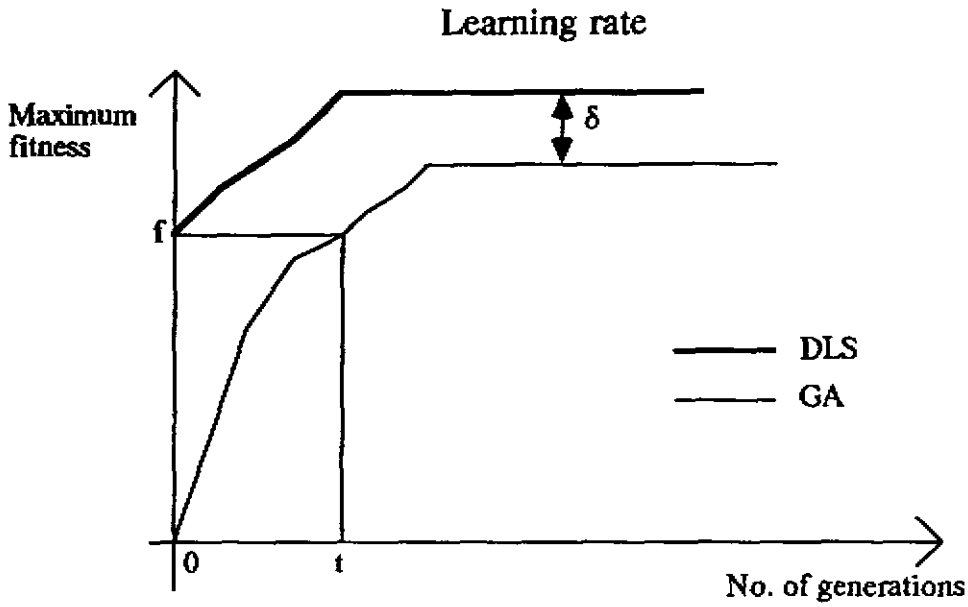

Figure 7.1 Performance Comparison Between the Distributed Learning Approach and GA

Figure 7.1 also shows the improvement in terms of solution qualities. The improvement in the fitness measure(i.e., the learning performance in terms of the prediction accuracy) in this case is about 27.7% over the original version of the GA algorithm. The multi-agent version using DLS also converges much faster. This improvement in the quality of learning can be attributed to two factors: (1) Each learning agent starts out with hypotheses generated by PLS performed by the agent, providing a more accurate pool of hypotheses than the randomized initial descriptions used by the basic GA. (2) The multi-agent approach offers a more *diverse* set of population for executing the genetic adaptation procedure, comparing to the initial population of the basic GA algorithm. In genetic adaptation, a more diverse population usually results in better learning results. This aspect will be further discussed again.

Figure 7.2 shows the improvement of the multi-agent approach based on the DLS procedure over the single-agent version of the concept-learning procedure(i.e., PLS). Again, DLS achieves better learning accuracy as well as more concise concept descriptions. These results can be explained by several reasons: (1) the genetic algorithm component can apply more globally oriented search, thus enabling DLS to achieve better learning quality than single-agent learning; (2) the genetic adaptation component conducts searches with finer granularity, thus reducing the length of the final concept description; and (3) the multi-agent learning process using the DLS Algorithm synthesizes the results of N learning agents, thus is more likely to reach better performance. These performance results are further confirmed by a more detailed empirical study described in Sikora and Shaw[1990a].

Significance Level	Method	Accuracy	Rule Size
0.0	PLS	55.7%	15.2
	DLS	64.2%	7.0
0.1	PLS	54.3%	14.4
	DLS	64.3%	6.6
1.0	PLS	59.5%	8.5
	DLS	65.5%	6.5

Figure 7.2 Performance Comparison Between The Distributed Learning Approach and Single-Agent Learning

As a rule learning algorithm, the multi-agent learning process using the DLS Algorithm exemplifies the efficacy of the DAI approach to problem solving. In this case, the problem - the task of inductive learning - is possible to be handled by one agent alone. But the DAI approach can be used to reduce the complexity by decomposing it to N sub-problems, each of which is assigned to a learning agent. After the concept learning is completed by all agents, their results are synthesized by another induction step, which would serve to further refine the multi-agent learning results. Such a DAI approach enables *multiple searches* to be conducted simultaneously, and doing that in both the instance space and the hypothesis space(Rendell[1990]). The performance results therefore is predictably good.

Moreover, DLS also emulates a multi-agent environment for rule induction. It can be used as a model to characterize the essential features of multi-agent learning. To verify the contributing factors accounted for the performance improvements of the distributed problem-solving approach, the second rule-induction application related to the learning of process-control rules(Sikora&Shaw[1990c]) is used for evaluating DLS.

The problem involved learning rules for formulating the relationships between the amount of a chemical ingredient used in a complex industrial process and the other process variables. The process requires the use of an expensive chemical, X, to eliminate a undesirable by-product of the process. The objective is to find out the dependence relationship between X and the set of controllable variables, $y_1, y_2, ..., y_9$, of the process so that the amount of X used is as minimal as possible. Since it is impossible to derive any mathematical equations between X and y_i, inductive learning is used to generate decision rules describing in DNFs the relationships between the amount of X and y_1 to y_9. A threshold value e is used to dichotomize the decisions. The training examples then consist of a set of data cases, each with an array of values assigned to y_1 to y_9 as well as a class variable indicating whether the corresponding X is over or below e.

The objective is to learn the various conditions, described by the value-ranges of y_1 to y_9, that can ensure that the amount of X used is below e. There are a total of 574 data cases available, among which we used 459 for training and 115 for testing- 5 trials were used for each test case. We designed experiments using DLS to evaluate in particular two aspects of the distributed problem-solving approach: the degree of diversity and the degree of decomposition.

N = 5; Decomposition Ratio = 0.20			
No. of Agents with Different Solutions	Composition of Agents	Prediction Accuracy By the Group	Rule Size
1	5	79.3%	2.8
2	2+3	81.1%	2.2
2	1+4	81.2%	2.2
3	1+1+3	83.5%	2.8
3	1+2+2	84.4%	4.0
4	1+1+1+2	82.3%	2.8
5	1+1+1+1+1	86.9%	3.2

Figure 7.3 Performances Result of Group Learning with Varying Degrees of Diversity

Figure 7.3 presents learning performances when the partial learning solutions generated by the agents having varying degrees of diversity, indirectly reflecting the diversity of the agents' background knowledge for performing the learning processes. The purpose is to isolate the effects of diversity from the effects of using parallel searches. We evaluated DLS on different degrees of diversity in the composition of the agents. In each composition, there may be some agents, with size ranging from 0 to 5, that share the same partial solutions. The numbers in the figure represent numbers of agents sharing the same results, so (1+1+3) means that there are three different sets of partial solutions generated among the 5 agents- two agents have two different sets of partial solutions and the other three agents share the same partial solution.

As shown in Figure 7.3, the group's learning performance is closely correlated to the degree of diversity in the composition of agents. When the group result is synthesized from 5 completely different partial solutions, the learning performance is better than that of the cases when some the agents' solution are the same. This further confirms the assertion that *diversity* among the agents' hypotheses is a key factor to the performance of the distributed problem-solving approach.

The way the training examples are decomposed into sub-sets also affects the group performance. The learning problem can be decomposed into sub-problems with varying sizes. In one extreme, each agent can solve the same original problem(i.e., collective reasoning); on the

other hand, the problem can be decomposed into n sub-problems distributed to the n agents- so the total size of training examples used is equal to that in the single-agent case. If the latter case can be achieved with comparable, or better, learning results, then such a distributed approach becomes computationally more efficient.

No. of Agents / Decomposition Ratio	N = 2		N = 5		N = 7		N = 10	
	Predict. Accuracy	Rule Size	Predict. Accuracy	Rule Size	Predict. Accuracy	Rule Size	Predict. Accuracy	Rule Size
0.05	70.4%	2.2	73.0%	2.0	77.2%	3.0	77.4%	2.8
0.10	72.1%	2.4	78.8%	2.8	82.6%	3.4	78.4%	3.8
0.20	79.8%	3.2	86.9%	3.2	80.7%	2.8	80.0%	3.0
0.40	81.1%	3.6	84.1%	2.8	80.9%	2.8	82.3%	2.8
0.60	83.9%	2.8	81.9%	2.0	81.6%	3.0	80.5%	3.5
0.50	85.1%	3.4	----	----	----	----	----	----
0.15	----	----	----	----	80.5%	3.0	----	----

Figure 7.4 The Learning Results for Different Decomposition Ratios

As shown in Figure 7.4, we ran DLS with different number of agents; for each case, we try breaking the problem into sub-problems of different sizes. For instance, when the decomposition ratio is equal to 0.20, this means that each agent is assigned with 20% of the original set of training examples; if N is further specified to be 5, then the five agents together need the same number of training examples as the original set. The results in Figure 7.4 indicate that , although there is no consistent patterns pointing to the best decomposition ratio, it usually results in good results to use a decomposition ratio of 1/N, where N is the total number of agents. When N = 2, 5, 7, and 10, this produces 85.1%, 86.9%, 80.7%, and 78.4% of prediction accuracy, respectively.

Within the distributed problem-solving framework, there are two opposing factors to consider when deciding on the number of agents to use. On one hand, one would like to have more agents to increase diversity; on the other hand, one would like to have fewer agents so as to assign as many training examples as possible to each agent. In our case, letting N to be 5 and assigning 20% of the training examples to each agent result in the best overall learning performance in terms of prediction accuracy. This points to the plausibility of applying the distributed approach *without* using redundant resources.

It is still unresolved as to what is the optimal number of agents for a given learning problem.

For this particular problem, the best value for N is empirically shown to be 5. But whether 20% of the training set would be sufficient for the local problem-solving phase is really a problem-dependent issue. This issue of decomposition is also closely related to the issue of the global coherence issue in distributed problem solving in a DAI system: that is, how much should each agent know, in its local view, so that its partial solution can be as globally coherent as possible. In the DLS implementation, we used a statistical sampling technique, the jackknife technique, for decomposing the training set. The jackknife technique is known to give unbiased estimates of the sample means, which is one interesting way to ensure global coherence.

To ensure that the performance improvements of DLS over single-agent oriented methods, such as PLS and the pure GA, are due to the distributed, multi-agent nature of the methodology, not to the particular way we combined two types of rule-induction algorithms- PLS and GA - together, we also tested a single-agent version of DLS which basically runs PLS through the whole set of training examples, and then the GA is executed with the learning results from PLS as the input. The prediction accuracy is 82.1%, worse than the 2-agent and the 5-agent cases when distributed problem solving is used. But why would the learning performances deteriorate when N = 7 or 10?

No. of Agents	N = 1	N = 2	N = 5	N = 7	N = 10
Learning Performance of the Group	82.1%	85.1%	86.9%	80.5%	78.4%
No. of Hypotheses Generated By the Group	15	21.4	25.8	29.8	31.6
Avg. No. of Hypothese Generated By Each Agent	15	10.7	5.2	4.2	3.2

Figure 7.5 Comparison of Learning Performances in Relation To The No. of Agents and The Size of the Hypotheses Generated

As shown in Figure 7.5, there is a difference in the size of the pool of hypotheses generated by the agents. When a single agent is used to learn the concept description(i.e., the rules) from the whole set of training examples, the pool of hypotheses generated, in DNF, has 15 different conjuncts. Each conjunct is a hypothesis for the final concept description. On the other hand, when the 5-agent version of DLS is run, with each agent assigned with 20% of the training examples, the pool of hypotheses generated has an average of 25.8 different concept hypotheses.

A trend emerging from the Figure7.5 is that the marginal contribution of additional hypotheses from each agent decreases gradually as N increases. Moreover, the group learning

performance starts to decline after its peak, at around 5 in this case. The same type of trade-offs exist in both multi-agent learning and distributed problem solving. When there are more agents participating, more ideas(that is, hypotheses or partial solutions) can be generated by the group. Such an increasing pool of ideas contributed by agents help finalize the group solution faster. At the same time, Since a fixed amount, R, of resources are allocated to the group, the resources available to each agent is R/N. With the amount of resources for getting its solution getting smaller, the quality of the contributions by each agent would decrease as N increases. In the case of rule learning, the resources are the training examples given to the group. Because of these two opposing forces, both affected by N, that influence group performance, the group performance would peak at a certain N and then decline, which is exactly what happens in Figure 7.5.

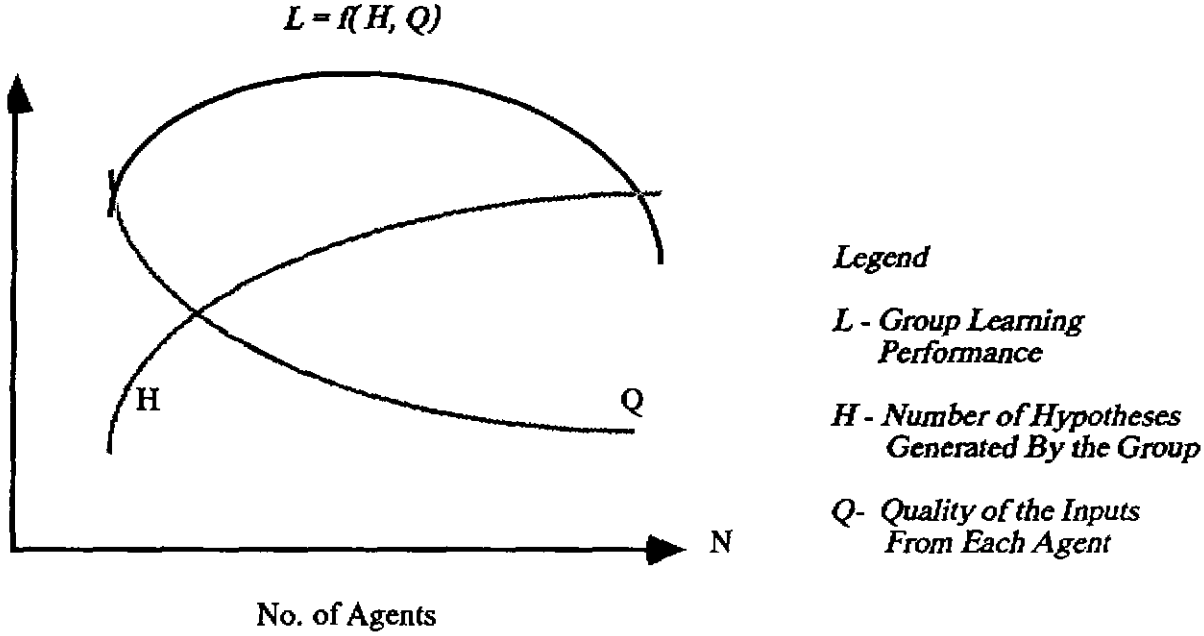

Figure 7.6 The Relations Between The Group Learning Performance and the Number of Agents

The findings and the ensuing analysis above point to the importance of the contributions from the agents in terms of the number of hypotheses generated. This explains why the distributed problem-solving approach performs better than the single-agent version. It also explains the previous observation, as shown in Figure 7.3, that greater diversity results in better group performance. Let the number of hypotheses generated by the whole group be H, then H can be represented by the quadratic functional form in relation to N as shown in Figure 7.6, with a decreasing gradient since the marginal contribution from each agent decreases as N increases. The other factor affecting the group performance is the quality of the hypotheses submitted by the agents. In rule learning, the quality of the hypotheses are directly determined by the number of training examples allocated to each agent, which can be viewed as the resources allocated. More agents deplored means less resources allocated to, therefore poorer quality from, each agent in

forming the group solution. In the rule learning case, when each agent is assigned with smaller set of training examples, the individual agents' learning results become less accurate. Let E be the total number of training examples, then the resources available to each agent is represented by the number of training examples assigned to each agent, E/N. Since the quality, denoted by Q, of the hypotheses generated by an agent's learning process is positively proportionate to the number of training examples allocated, Q would follow the functional form of Y = E/N, as shown in Figure 7.6. The group learning performance L will be a function of these two factors, i.e.,

$$L = f(H, Q),$$

and can be represented in the functional form shown in Figure 7.6.

The dynamic of the group learning performance with respect to the number of agents as captured by the model, shown in Figure 7.6, explains quite nicely the empirical observation in Figure 7.4. The model underscores the key role played by the pool of hypotheses generated by the agents for reaching a group solution, as has been observed by Laughlin and Shippy[1983] and by our empirical results reported in Figure 7.5. This aspect of the model also explains the importance of diversity among the learning agents, as observed from Figure 7.3, to the group performance, since the more diversity among the agents, the more varieties of hypotheses can be generated, resulting in larger H. The dependence of L on the quality of the hypotheses submitted by the agents confirms the view that the sub-problems assigned to each agent should be as representative of the global view as possible.

7. Conclusion

This paper has shown that the distributed problem-solving approach to rule induction, as implemented in DLS, is useful in two ways: (1) it can be used as a new rule learning algorithm, with better performances than existing algorithms based on single-agent learning models; and (2) it provides a rich model for multi-agent learning, enabling us to analyze the important parameters and mechanisms in multi-agent learning.

As a new learning method, the distributed learning approach performs well computationally because of (1) it's ability to pursue multiple search paths in parallel, and (2) the combination of inductive learning mechanisms searching through the instance space in the agent level and the hypothesis space in the group level, resulting in more likelihood to reach a good concept description. Besides producing better learning results as measured by prediction accuracy, the DLS method also gives more concise descriptions than PLS in the rules it generates, and quicker convergence than GA.

The empirical results of DLS also reveal interesting characteristics of multi-agent learning. The contrasts between existing inductive learning method and DLS is analogous to those between inductively learn concept-description rules from E examples by a single agent vs. letting N learning agents inductively generate rules from the same E examples together, but each agent is

assigned with E/N examples to produce its partial solutions. We showed that the distributed version has more favorable learning performances than the single-agent learning method. However, the group performances vary with different values for N. The group learning performance peaks at a certain value of N, N*, the optimal number of agents for the learning task, and then starts to deteriorate.

In view of the importance of generating and sharing the hypotheses in the multi-agent learning system, a group mechanism must be used to help the group sort through the pool of hypotheses generated, evaluating them, and making decision on what the correct concept description should be in the group solution. In DLS we incorporated a solution-synthesis procedure using the genetic algorithm, which plays an important role in generating the group solutions based in the individual agents' results. This synthesis step should be generalizable to other distributed problem-solving domains.

Based on the observations, we have constructed a model relating the group learning performance to two factors: (1) the amount of hypotheses generated by the group of learning agents, and (2) the quality of hypotheses generated. The model confirms the empirical findings that the diversity among the group is important since it affects the number of hypotheses that can be generated. It also explains why the group learning performance would decline when N goes over N*. Since the sub-sets of examples allocated to each agent would become smaller, thus affecting the accuracy of the rules generated, while at the same time the marginal contributions of new hypotheses by an additional agents is declining. This model shows in a formal manner the performance characteristics of the distributed approach to rule induction.

Appendix

The basic Genetic Algorithm (GA) can be described by the following procedure:

```
PROCEDURE GA (initial population of n members { gi, i = 1,..., n })
begin;
no-of-generations = 0;
repeat;
    for each member  gi of the population;
        compute f( gi), the fitness measure for each member; /* evaluation */
    repeat;
        stochastically select a pair of members g1, g2
            with probability increasing with their fitness f; /* reproduction */
        using the genotype representation of g1 and g2,
            mutate a random bit with probability pu; /* mutation */
        randomly select a crossover point and
```

```
        perform uniform crossover on g1 and g2 to
         give new genotypes g'1 and g'2; /* crossover */
    until the new population is filled with n individuals g'i;
    no-of-generations = no-of-generation + 1;
until the number-of-generations has reached Ng ; /* termination */
end;
```

Reference

Bond, A., " The Cooperation of Experts in Engineering Design," *Distributed Artificial Intelligence*, II, Gasser and Huhns, Pitman, London, 1989.

Brady, R.M., " Optimization Strategies Gleaned from Biological Evolution," *Nature*, V. 317, Oct. 1985, pp. 804-806.

Burton, R.M., Damon,W.W., and Loughridge,D.W.," The Economics of Decomposition: Resource Allocation Vs. Transfer Pricing," *Decision Science*, V.5, No.3, 1974, pp.297-310.

Ching, C., Holsapple, C., Whinston, A., " Reputation, Learning, and Coordination in Distributed Decision-Making Contexts," forthcoming in *Organization Science*, 1990.

Cohen, M.D., " Artificial Intelligence and the Dynamic Performance of Organizational Designs," in *Ambiguity and Command*, March, J. and Weissinger-Baylon, R. (Eds.), Pitman Publishing Inc., Marshfield, MA, 1986

Dantzig,G. and Wolfe, P., " The Decomposition Algorithm for Linear Programming," *Econometrica*, V.29, No.4, 1961, pp.767-778.

Davis, R. and Smith, R., " Negotiation as a metaphore for Distributed Problem Solving," *Artificial Intelligence*, V.20, 1983, pp.pp.63-109.

Decker, K.S., " Distributed Problem Solving: A Survey," *IEEE Transactions on Systems, Man, and Cybernetyics*, Vol.17, No. 5, Sept. 1987.

Durfee, E.H., Lesser, V.R. and Corkill, D.D., "Cooperation Through Communication in a Distributed Problem Solving Network", in *Distributed Artificial Intelligence*, M. Huhns (Ed.), Pitman, London, U.K., 1987, pp. 29-58.

Durfee, E.H., Lesser, V.R. and Corkill, D.D., " Cooperative Distributed Problem Solving," in the Handbook of Artificial Intelligence, Vol. IV, Barr, Cohen, and Feigenbaum(eds.), Addison-Wesley, Reading, MA, 1989, pp.83-148.

Durfee, E.,H., and Lesser, V.R., " Negotiating Task Decomposition and Allocation Using Partial Global Planning," in *Distributed Artificial Intelligence*, vol. II, L. Gasser and M. Huhns (eds), Pitman, London, 1989, pp. 229-244.

Durfee, E. H. and Montgomery, T., " A Hierarchical Protocol for Coordinating Multi-agent Behaviors: An Update," Proc. of the 10th AAAI International Workshop on Distributed Artificial Intelligence, Texas, Oct., 1990

Fox, M.S., " An Organizational View of Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11, 1981, pp.70-79.

Gasser, L. and Huhns, M., *Distributed Artificial Intelligence*, II, (eds.), Pitman, London, 1989.

Goldberg, D. *Genetic Algorithm*, Addison-Wesly, Reading, MA, 1989.

Holland, J.H., *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

Holland, J.H., *Induction: Processes of Inference, Learning and Discovery*, MIT Press, Cambridge, MA, 1986.

Kornfeld, W. and Hewitt,C., "The Scientific Community Metaphor," *IEEE Transctions on Systems, Man, and Cybernetics*, Vol. SMC-11. No. 1, January 1981.

Kraemer, K.L. and King, J., "Computer-based Systems for Cooperative Work and Group Decision Making," *Computing Survey*, Vol. 20, pp. 115-146, 1988.

Kuhn, T.S., *The Structure of Scientific Revolution*, University of Chicogo Press, 1970.

Laasri,B., Laasri,H., Lesser,V., " Negotiation and Its Role in Cooperative Distributed Problem Solving," Proc. of the 10th AAAI International Workshop on Distributed Artificial Intelligence, Texas, Oct., 1990

Lasdon,L.S., " Duality and Decompositin in Mathematical Programming," *IEEE Trans. on Man, Syst., and Cyber.*, 4, 1968, pp.86-100.

Laughlin, P.R. and Shippy, T.A., "Collective Induction," *J. of Personality and Social Psychology* (45:1), 1983, pp. 94-100.

Lesser, V.R. and Corkill, D.D., "Functionally Accurate, Cooperative Distributed Systems," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 11, no. 1, 1981, pp. 81-96.

Linstone, H.A. and Turoff, M. (eds.), *The Delphi Method: Techniques and Applications*, Addison-Wesley, Reading, MA, 1975.

Lounamaa, P. and March, J., " Adaptive Coordination of a Learning Team," *Management Science*, 33, 1987, pp. 107-123.

Malone, T.W., "Modeling Coordination in Organizations and Markets," *Management Science*, 1987, vol. 33, pp. 1317-1332.

Michalski, R.S., " A Theory and Methodology of Inductive Learning," *Artificial Intelligence*, 20, 1983, pp.111-161.

Minsky, M., *The Society of Mind*, Simon and Schuster, Inc., N.Y., 1985.

Nunamaker, J.F., Applegate, L.M. and Konsyuski, B.R., "Computer-aided Deliberation: Model Management and Group Decision Support," *Operation Research*, 1988, vol. 36, no. 6, pp. 826-847.

Quinlan, J.R., " Induction of Decision Trees," *Machine Learning*, 1, 1986, pp.81-106.

Parunak, H.V.D., " Manufacturig Experience with the Contract Net" in *Distributed Artificial Intelligence*, M. Huhns (Ed.), Pitman, London, U.K., 1987, pp. 285-310 .

Rendell, L., " A New Basis for State-Space Learning Systems and a Successful Implementation," *Artificial Intelligence*, No.2, 1983, pp.177-226.

Rendell, L., " Induction as Optimization," *IEEE Transactions on Systems, Man, Cybernetics*, V.20, No.20, 1990.

Sathi, A. and Fox, M., "Constraint-Directed Negotiation of Resource Reallocations," *Distributed Artificial Intelligence*, vol. II., L. Gasser and M. Huhns (Eds.), Pitman, London, 1989, pp. 163-194.

Shaw, M., "Applying Inductive Learning to Enhance Knowledge-based Expert Systems," *Decision Support Systems*, 1987, vol. 3, pp. 319-322.

Shaw, M., " Multi-Agent Learning and Problem-Solving Methods for Group Decision Support: A Distributed Problem Solving Framwork", Proc. of the First Internation Society of Decision Support Sysems Conference, Austin, Texas, 1990a.

Shaw, M., "Mechanisms for Cooperative Problem Solving and Multi-agent Learning in Distributed Artificial Intelligence Systems," Proc. of the 10th AAAI International Workshop on Distributed Artificial Intelligence, Texas, Oct., 1990b.

Shaw, M. and Gentry, J., " Inductive Learning for Risk Classification," *IEEE Expert*, Feb. 1990, pp. 47-53.

Shaw, M. and Whinston, A.B., "A Distributed Knowledge-based Approach to Flexible Automation: The Contract-Net Framework," *International Journal of Flexible Manufacturing Systems*, 1988, pp. 85-104.

Shaw, M. and Whinston, A.B., "Learning and Adaptation in Distributed Artificial Intelligence Systems," in *Distributed Artificial Intelligence*, vol. II, L. Gasser and M. Huhns (eds), Pitman, London, 1989, pp. 413-430.

Shaw, M. Harrow, B., and Herman, S., " NEST: A Networked Expert Systems Testbed for Cooperative Problem Solving," Proc. of the Working Conference on Cooperating Knowledge Based Systems, University of Keele, Keele, UK, 1990.

Shaw, M. Harrow, B., and Herman, S., " Distributed Artificial Intelligence for Multi-agent Problem Solving and Group Learning," forthcoming in *Proc. Hawaii International Con. of Systems Science*, IEEE Press, 1991.

Sikora, R. and Shaw, M., " A Double-Layered Learning System For Financial Classification," submitted for publication, 1990a.

Sikora, R. and Shaw, M. " A Distributed Problem Solving Approach to Inductive Learning", Technical Report No. CMU-RI-TR-90-26, Robotics Institute, Cranegie Mellon University, 1990b.

Sikora, R. and Shaw, M., " Applying DLS to Learning Rules for Controlling a Chemical Process," in preparation, 1990c.

Smith, R.G., "The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computer,* 1980, vol. 29, pp. 1104-1113.

Smith, R. G., and Davis, R., " Frameworks for Cooperation in Distributed Problem Solving," *IEEE Transctions on Systems, Man, and Cybernetics*, Vol. SMC-11. No. 1, January 1981, pp. 61-70.

Stefik, M., Foster, G., Bobrow, D.G., Kahn, K., Lanning, S., and Suchman, L., "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings," *Comm. Ass. Comput. Mach.,* 1987, vol. 30, no. 1, pp. 32-47.

Sycara, K., " Multi-Agent Compromise via Negotiation," *Distributed Artificial Intelligence (Vol. II)*, L. Gasser and M. Huhns(Eds.), Pitman, London, 1989, pp. 119-137.

Weihmayer, R., Brandau, R., Shinn, H., " Modes of Diversity: Issues in Cooperation Among Dissimilar Agents," Proc. of the 10th AAAI International Workshop on Distributed Artificial Intelligence, Texas, Oct., 1990

Woo, C.C. and Lochovsky, F.H., "Supporting Distributed Office Problem Solving in Organizations,"*ACM Trans. on Office Information Systems*, July 1986, pp. 185-204.