Self Organizational Approach for Integration of Distributed Expert Systems

Tatsuaki Itoh, Takashi Watanabe, and Takahira Yamaguchi
Faculty of Engineering, Shizuoka University
3-5-1 Johoku, Hamamatsu, JAPAN, 432
{t-itoh, watanabe, yamaguti}@cs.shizuoka.ac.jp

Abstract

In development of Expert Systems (ES), it is important to acquire knowledge from multiple human experts. Cooperative Distributed ES (CDES) is a framework which can unify multiple expertise to develop a large scaled ES. In CDES, ESs which are in the process of building through acquiring knowledge from human experts correspond to agents, and task structure is learned by cooperation among agents based on extended Contract Net. At first, the fundamental structure of CDES is provided. An experiment is made to evaluate algorithms of a Bid analyzer on a testbed of CDES. Then, self-reorganizational schemes are provided by changing the agent's scope of domain knowledge, that is granularity, by themselves, from two viewpoints - one of decomposing into fine grain, and the other of composing into coarse grain. Two experiments are made to learn the subtask structure to change the weight of cooperation among agents dynamically. Finally, we discuss the scheme which incorporates decomposition and composition of agents.

Introduction

The performance of an Expert System (ES) depends on the quantity and quality of the knowledge-base that stores expertise of the problem domain. As it is difficult to acquire a large quantity of good quality knowledge from experts, knowledge acquisition becomes a bottleneck to construct knowledge-bases. So we need multiple experts to acquire extensive knowledge quickly.

On the other hand, much research on multiagent systems has been done since 1980. There is much research that investigates the cooperation among ESs. Knowledgeable Community (Nishida, Takeda, & Iino 1994) is a framework of knowledge sharing and reusing based on a multi-agent architecture. Research towards organization self-design, in which agents with production systems observe their environment to reorganize dynamically, are proposeded by Ishida (Ishida, Gasser, & Yokoo 1992). Zhang (Zhang 1992; M. Zhang & C. Zhang 1994) proposes a strategy of unifying solutions that include uncertainty from multiple ESs that use different representation of uncertainty. But little research has been made from the viewpoint of distributed knowledge acquisition and integration of it to make a large scaled ES.

Cooperative Distributed Expert System (CDES) is a framework to unify multiple expert models. In our previous work (Itoh, Watanabe, & Yamaguchi 1994), we first constructed the fundamental structure of CDES based on extended Contract Net with Task Announcement (TA) analyzer and a Bid (BD) analyzer. In CDES, the performance of problem solving is dominated by the BD analyzer. Then, we proposed two self-reorganizational schemes (decomposition of agents and composition of agents). Each scheme was evaluated by some experiments using two algorithms of the BD analyzer - MaxP first scheme, and MinP first scheme. As a result, MaxP first scheme, which selects an agent with a higher probability of success, was evaluated to be the better of two algorithms of the BD analyzer.

In this paper, we first propose two algorithms of the BD analyzer - Random scheme, and Roulette Spin scheme, whose performance top algorithms proposed before. Secondly, self-reorganizational schemes based on the algorithm of the BD analyzer are provided by changing the agents' scope of domain knowledge, that is granularity, by themselves, from the viewpoints of decomposing into fine grain and of composing into coarse grain. Some experiments are made to learn the subtask structure to change the weight of cooperation among agents dynamically. Finally, an incorporated scheme of decomposition and composition is discussed.

Overview of CDES

In CDES, ESs are regarded as agents which are in the process of building through acquiring knowledge, as shown in Figure 1. Each expert makes his own local knowledge base (KB). KBs may include inaccurate rules, overlapped knowledge, and partial knowledge of the domain. After knowledge acquisition, agents learn a task structure which is not given a priori by some teaching problems. During the task structure learning

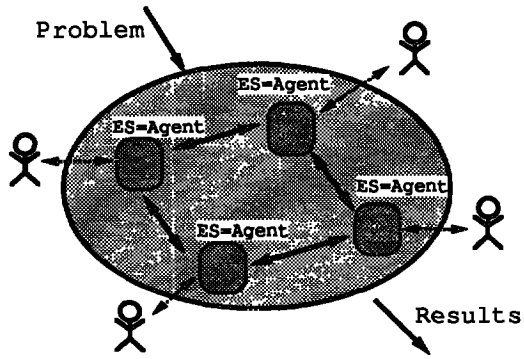

Figure 1: Overview of CDES

process, each agent infers with the local KB and produces a partial solution. Then the agent passes the solution to another agent. If the agent could not solve the problem even partially, if it fails, it sends the problem back to the agent which committed it. An internal inability due to the inaccuracy or partiality of KB causes the failure. Propagated fault of other agents may trigger another failure as well. The trace of agents, called a cooperation path, is weighted to record inference success or failure. A path with the highest weight will become a task structure of the whole CDES. But, if all cooperation paths are inaccurate, task structures cannot be acquired. In this case, agents try to learn task structures to change their organization by themselves.

Realization of CDES

CDES is realized through three development processes.

1. **Knowledge Acquisition phase:** human expert inputs and modifies domain knowledge.
2. **Learning phase:** the task structure is learned.
3. **Problem Solving phase:** CDES is applied to practical use.

First of all, human experts input and modify domain knowledge in the Knowledge Acquisition phase. Then, task structure is learned in the Learning phase, after the end of domain knowledge modification. Finally, CDES is applied to practical use. But if the knowledge of each ES is inaccurate, task structures cannot be acquired in the learning phase. Or, even if a task structure is learned for the teaching problems, it may fail problem solving in practical use. In these cases, CDES returns to the Knowledge acquisition phase to modify knowledge. Through this cycle, CDES is realized. In this paper, we focus on the learning phase to discuss a scheme to learn a task structure.

The power plant system operation planning ES

As an example, we use a power plant system operation planning ES. used practically to schedule tasks to maintain power plants. The task structure consists of 28 complicated tasks, and each task (called 'unit') has a subtask structure which consists of some subtasks. The final goal of our research is to find a framework which can learn task structure, but we will first develop a fundamental scheme to learn the subtask structure.

We focus on a simple task (unit 20). This task consists of two subtasks - generation and test-elimination of a schedule table. The generation subtask and the test-elimination subtask consist of 2 rules (rule1-2) and 11 rules (rule3-13) respectively. We assume that agent-0 has completely accurate knowledge of unit0 through 19, and agent-F also has thorough knowledge of unit21 through 28.

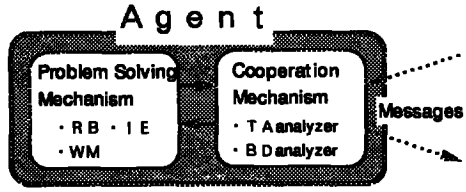

Figure 2: Structure of agent

Fundamental Structure of CDES

Structure of agent

As shown in Figure 2, each agent has two mechanisms - a problem solving mechanism, and a cooperation mechanism. The former works as a single ES, and the latter facilitates the agent's interaction with other agents.

Definition 1 a set of agents and a set of rules.

A: a set of agents.

R_D: a set of rules of the problem domain.

$R_x \subset R_D$: a set of rules that agent $x \in A$ owns.

Subtask structure learning process

We define an order of agents which have collaborated to produce partial results on the problem so far as a "cooperation path". Cooperation paths are weighted to record inference success or failure. The weight of cooperation path is modified according to the result of problem solving.

Definition 2 Cooperation path

CP: a set of potential cooperation path;$CP = \{\lambda\} \cup A \times \{\lambda\} \cup A \times \{\lambda\} \cup A \cdots \times \{\lambda\} \cup A$

$\sigma \in CP$: a cooperation path; The length of σ is less than or equal to $|A|$.

$CP_x \subset CP$: a set of cooperation path which agent x has learned.

W: a set of weight of cooperation path; $W = I \cup \{US\}$. Where

I: a set of integers.

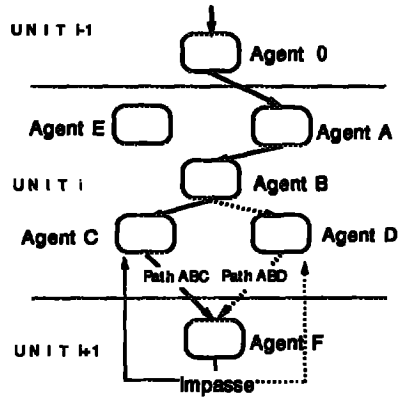

Figure 3: Subtask structure learning process

US: a tag of unused cooperation path.

w: a mapping on CP to W; $w : CP \rightarrow W$

P: a set of problems.

σ_p: current cooperation path of a problem $p \in P$.

An informal description of the algorithm is described as follows (Figure 3). Here we consider learning a subtask structure of unit20, and a case where there are 5 agents of unit20 (agent-A, B, C, D, and E) and agent0, agent-F as defined before. After agent-A receives results of the unit 0 through 19 from agent-0, it infers by itself by using all applicable rules, and agent-A interacts with agent-B, C, D and E using a cooperation mechanism. Suppose that Agent-A decides to commit the problem to agent-B, and then agent-B commits the subsequent solution to agent-C. If agent-D and E don't have any rules which can be used, agent-C sends the final result to agent-F. At this time, the cooperation path is *ABC*. After agent-F receives the results, it applies local rules to the problem. If cooperation path *ABC* is inadequate, agent-F can't solve the problem correctly. This situation is referred to as an "impasse". When an impasse occurs, agent-F informs the last agent of the cooperation path (agent-C) of the impasse. Then agent-C decreases the weight of cooperation path *ABC* and propagates the impasse to agent-B. When agent-B knows this, the agent decreases the weight of the cooperation path, as well as agent-C, but searches for another applicable path, e.g. *ABD*. If all cooperation paths which agent-B can search fail, agent-B sends a message to agent-A to find another path.

If cooperation path *DE* is made and agent-F succeeds to infer the problem, agent-F informs agent-E of the fact. Then agent-E increases the weight of the path *DE*. Agent-E also sends a message to agent-D to increase the weight of the relevant path.

Interaction among agents

In this paper, interaction among agents is based on the Contract Net (Smith 1980). There are 6 basic messages to learn the subtask structure:

Task Announcement (TA)

content sender, cooperation path, history of applied rules

function Asks whether other agents have some rules which can be used.

Bid (BD)

content sender, number of rules which can be used

function Informs that the sender has some rules which can be used.

Anti Bid (ABD)

content sender

function Informs that the sender has no rule which can be used.

Award (AWD)

content sender

function Commits problem solving.

Success (SC)

content sender

function Informs that problem solving has been successful.

Fail (FL)

content sender

function Informs that problem solving has failed.

Agents interact to solve problem as follows (Figure 4):

1. After an agent receives problems, it infers by itself by using all applicable rules.
2. It broadcasts TA with a history of applied rules.
3. Agents receiving TA analyze the announcement by Task Announcement analyzers (TA analyzer) to determine whether they have applicable rules. If any, they send BD, otherwise they send ABD.
4. After the agent receives BD or ABD from all agents, it determines an agent to commit problem solving by BD analyzer and sends AWD for the agent.
5. After agent receiving AWD knows the result of problem solving, it sends SC or FL according to the result.

BD Analyzer

TA analyzer is quite simple. If an agent has some rules which are not listed in the history, TA analyzer returns true. Whereas the search space and the performance of the system depends on the BD analyzer. We consider 4 algorithms of the BD analyzer. 1. and 2. have been already proposed (Itoh, Watanabe, & Yamaguchi 1994).

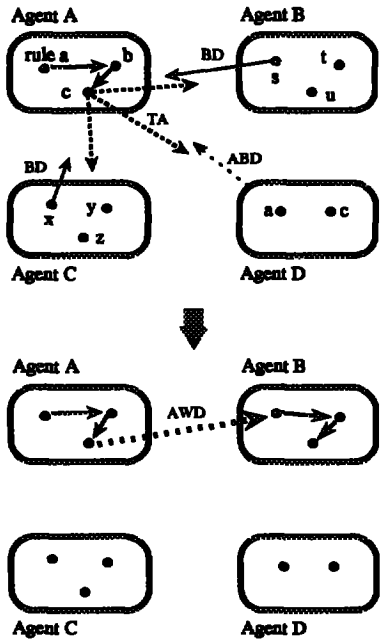

Figure 4: Interaction among agents

1. maximum weight path first (MaxP first scheme)
2. minimum weight path first (MinP first scheme)
3. Random scheme
4. Roulette Spin scheme

Experiment to evaluate the BD analyzer To evaluate these schemes, a testbed of CDES is implemented with a problem solving mechanism by AI tool ART (Inference Corp. 1991) and a cooperation mechanism by interprocess communication on a UNIX workstation. We make the following assumptions:

- Rules can be identified with other rules with interlingua like KIF (Genesereth & Fikes 1990; Ginsberg 1991).
- Structure of unit20 is unknown.
- 5 agents of unit20 exist corresponding to 3 human experts. Each agent has its own rules (Table 1).
- All rules are distributed among agents, i.e., $\cup_{a \in A} R_a = R_D$.
- The weight of cooperation path is increased by success and decreased by failure (impasse).
- 28 teaching problems are given to the system with several problems overlapped.

Evaluation of BD analyzer 4 schemes are evaluated as follows(Table 2):

MaxP As a path with the highest weight is selected first, a path with a high probability of success can be acquired. However, an acquired subtask might be a local maximum, because a nonoptimal path may be evaluated extremely, and the optimal path would hardly be applied.

MinP As a path with the lowest weight is selected first, all of the optimal paths can be acquired. But, there is little difference (maybe no difference) between the most weighted path and the second most weighted path, so it is difficult to specify an optimal path. And as a path with higher possibility of failure is applied first, this algorithm takes a lot of time to search.

Random As a path is selected at random, all of the applicable paths can be used by solving many problems. But it is difficult to specify an optimal path from among many candidates when there is little difference between the most weighted path and the second most weighted path.

Roulette Spin A path is selected at a rate of weight. As higher weighted paths are selected with a high probability, paths with a higher probability of success can be acquired.

Random scheme improves the defects of MaxP first scheme and MinP first scheme (all applicable paths might not be used, search time is too long, respectively). But it may be difficult to specify an optimal path, and it has no strategy to select paths with a higher probability of success prior to those with lower probability. The Roulette Spin scheme overcomes this weak point. And Table 2 shows that it takes comparatively little time for this scheme to search the space. So, the Roulette Spin scheme is considered to be optimal among these schemes.

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	o	o											
B			o	o	o	o	o	o	o	o	o	o	o
C	o			o		o		o	o	o			o
D		o	o		o		o				o	o	
E	o			o		o		o					

Table 1: Rules distributed among agents

algorithm	number of search	application of all paths	optimal path specification
MaxP first	88	×	o
MinP first	306	o	×
Random	172	o	△
Roulette Spin	106	o	o

Table 2: results of experiment

Self Reorganization

In some cases, the subtask structure can be learned with the above-mentioned algorithm (fixed search algorithm). However, we found many cases (e.g. Table

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	o		o	o	o	o	o						
B		o						o	o	o	o	o	o

Table 3: Rules distributed among agents

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	(o)		(o	o	o	o	o)						
B		(o)						(o	o	o	o	o	o)
C	(o)					(o	o	o	o)				

Table 4: Rules and subtask structures of agents

3) in which the fixed search algorithm can not be applied. For example, in the case of Table 3, learning of the subtask structure of unit20 would fail because the order of applied rules in any cooperation path involves the (generate rule) - (test-eliminate rule) - (generate rule) or (test-eliminate rule) - (generate rule) - (test-eliminate rule) path. So rules would not be used in the order of correct subtask structure. So, the subtask structure cannot be acquired because all cooperation paths are inaccurate. In these cases, agents try to modify the granularity of each agent by themselves. Two strategies to modify granularity can be considered.

- Strategy to decompose to fine grain (decomposition of an agent)
- Strategy to compose to coarse grain (composition of agents)

In the case of Table 3, by composing agent-A and agent-B into a single agent, it complies to all the rules of unit20. Clearly, the composed agent can solve any problem. This solution degrades the advantage of distributed agents, because if 2 agents are composed into one and if some revision on KB is necessary, 2 human experts must cooperate with each other on the unified knowledge base. They have to understand the meanings of the big KB which includes other expertise.

On the other hand, by decomposing agent-A into agent-A_1 which has rule 1 and agent-A_2 which has rule3-7, and by decomposing agent-B into agent-B_1 which has rule 2 and agent-B_2 which has rule 8-13, the optimal cooperation path (e.g. $A_1B_1A_2B_2$) can be acquired through the subtask structure learning process. Apart from the composing agents, it is easier for human experts to modify the decomposed local KBs. Note that though the decomposition into the finest granularity which is a trivial solution may enable agents to solve the problem successfully, the solution enlarges the search space. So, there will be optimal granularity in terms of the domain, the distribution of knowledge, and the inaccuracy of KBs. In this paper, we investigate how agents decide by themselves when and how they compose or decompose themselves to reorganize.

Decomposition scheme

3 strategies are proposed to decompose an agent. They are classified by timing, condition checker and objective agents to be decomposed. We formalize the strategies as follows.

Strategy 1
Checker {A}
Timing $(\forall\alpha)((A \in in(\alpha)) \wedge (weight(\alpha) < 0))$
Objective agents {A}

Strategy 2
Checker {A}
Timing $(\forall\alpha)((first(\alpha) = A) \wedge (weight(\alpha) < 0))$
Objective agents {A}

Strategy 3
Checker {A}
Timing $(\forall\alpha)((last(\alpha) = A) \wedge (weight(\alpha) < 0))$
Objective agents {A}

where

in : a mapping on CP to A:

$$in : CP \rightarrow A$$

which returns a set of agents included in cooperation path.

$first$: a mapping on CP to A:

$$first : CP \rightarrow A$$

which returns a first agent of cooperation path.

$last$: a mapping on CP to A:

$$last : CP \rightarrow A$$

which returns a last agent of cooperation path.

Costs to learn a subtask are different according to their strategies. There are costs for the search for a cooperation path, and costs for the decomposition of an agent.

Experiment to evaluate decomposition strategies

To evaluate the above-mentioned strategies, we made experiments. We make the following assumptions. Other assumptions are identical as those of previous experiments.

- 3 agents of unit20 exist corresponding to 3 human experts. Each agent has its own rules and subtask structure (Table 4). Parenthesized rules denote subtask structures which are supposed by experts.
- Granularity changes units of subtask and units of rule, in that order, per single decomposition.
- A Roulette Spin scheme is used for a BD analyzer.
- The total number of searches is taken as a measure of search cost.
- The total number of decompositions is taken as a measure of decomposition cost.

	total number of search	total number of decomposition
Strategy 1	95	1
Strategy 2	46	1
Strategy 3	73	1

Table 5: Result of the experiment

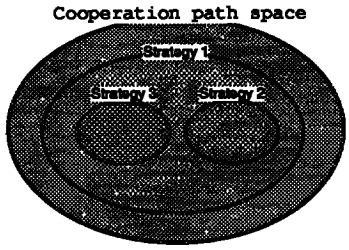

Figure 5: Cooperation path space in which an agent has to check the condition of decomposition

Result of experiment

As shown in table 5, Strategy 2 costs least. In general, if the decomposition condition is easier, the number of search for one decomposition increases while the number of decomposition decreases. Figure 5 shows the space of cooperation path that an agent has to check the condition of decomposition. The space of Strategy 2 and Strategy 3 are subsets of the space of Strategy 1. So, Strategy 1 requires a larger number of searches than Strategy 2 and 3.

The first agent of a cooperation path can use all its rules. On the other hand, the last agent of the cooperation path can use only part of its rules if other agents have used some rules which the last agent has. So, the ability of an agent is reflected in the cooperation paths whose head is the agent. So, Strategy 2 is better than Strategy 3, because Strategy 2 can decompose agents reflecting its ability.

Therefore, Strategy 2 would be optimal among the 3 strategies.

Composition scheme

Two strategies are proposed to compose agents. They are classified by timing, condition checker and objective agents to be composed.

Strategy 1
Checker {last(α_2)}
Timing
$(weight(\alpha_1 A \alpha_2 B \alpha_3) < 0) \wedge (weight(\alpha_1 B \alpha_2 A \alpha_3) < 0)$
Objective agents {A,B}

Strategy 2
Checker {last(α_2)}
Timing $(weight(\alpha_1 A B \alpha_2) < 0) \wedge (weight(\alpha_1 B A \alpha_2) < 0)$
Objective agents {A,B}

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	(o)		(o	o	o	o	o)						
B		(o)						(o	o	o	o	o	o)
C	(o)		(o	o				o	o)				
D		(o)			(o	o				o	o)		
E							(o					o	o)

Table 6: Rules distributed among agents

	total number of search	total number of decomposition
Strategy 1	63	1
Strategy 2	85	1

Table 7: Result of the experiment

Experiment to evaluate composition strategies

To evaluate the above-mentioned strategies, we make the following assumptions. Other assumptions are identical with those of former experiments.

- There are 5 agents in unit20. Each agent has its own rules and subtask structure (Table 6).
- The total number of searches is taken as measure of the search cost.
- The total number of compositions is taken as measure of composition cost.

Result of the experiment

Table 7 shows the results of the experiment. Among these strategies, Strategy 1 costs least. The condition of Strategy 2 is harder to be satisfied than Strategy 1, because the cooperation path space of Strategy 1 on which an agent checks the condition of composition is larger than that of Strategy 2, as shown in Figure 6. Therefore, this strategy takes a larger number of searches than Strategy 1. As the number of agents increases, the number of compositions of Strategy 1 becomes less than that of Strategy 2.

Thus the probability in Strategy 1 that the condition is satisfied is higher than that of Strategy 2. The condition for Strategy 1 tends to be satisfied faster than Strategy 2, so that Strategy 1 can narrow the search space. As a result, Strategy 1 would be optimal.

Incorporation of Decomposition and Composition

We consider the scheme which incorporates decomposition of agents and composition of agents. In this section, we show the scheme with Strategy 2 of decomposition and Strategy 1 of composition.

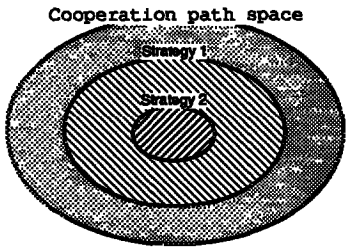

Figure 6: Cooperation path space in which an agent checks the condition of composition

Comparison between decomposition and composition

We assume that each scheme costs $COST_D$ (for decomposition) and $COST_C$ (for composition), respectively. The advantage of decomposition is that the correct cooperation path can be acquired by the decomposition of agents, because granularity of agent becomes fine. But it has the disadvantage of a large number of searches and communications. So this scheme costs much for search and communication. On the other hand, an agent with many rules can be acquired by the composing of agents. In addition, this scheme needs small number of searches and communications, so it costs less for this scheme to search cooperation paths and to communicate among agents. But it becomes harder to maintain internal consistency of knowledge-base, and to eliminate overlapped rules, when many rules are gathered into one agent. In addition, large memory is needed. So, although comparison cannot be made easily, $COST_C$ would be larger than $COST_D$ in general. So, it is better to select decomposition of agents when both condition are satisfied simultaneously.

Criteria to select a reorganizational scheme

To select a reorganizational scheme, the following costs should be considered;

- Cost for search after reorganization
- Cost for management of KB after reorganization
- Cost for reorganization

Though some equations could be made for $COST_D$ and $COST_C$, we represent them simply as the sum of each cost with coefficient.

$$\begin{aligned} COST_D &= C_{SD} \times (search) + C_{MD} \times (management) \\ &\quad + C_D \times (decomposition) \\ COST_C &= C_{SC} \times (search) + C_{MC} \times (management) \\ &\quad + C_C \times (composition) \end{aligned}$$

In this paper, a scheme which has lower costs has priority to reorganize agents. That is,

$$\begin{aligned} &if \quad COST_D < COST_C \quad \text{select decomposition} \\ &if \quad COST_D > COST_C \quad \text{select composition} \\ &if \quad COST_D = COST_C \quad \text{select randomly.} \end{aligned}$$

Assumption of the experiment

To show that the subtask structure of unit20 can be learned by this scheme, we make an experiment on the following assumptions.

- There are 5 agents in unit 20. Each agent has its own rules and subtask structure (Table 6).
- Granularity changes unit of subtask and unit of rule in that order per each decomposition.
- The Roulette Spin scheme is used for the BD analyzer.
- C_C equals 10 and other coefficients are 1, respectively. Each cost is defined as follows:

Decomposition

(Search) $= \frac{(N_a+N_{rg}-N_{a'})!}{\sum_{i=1}^{N_{a'}}(N_i!)}$

(Management) $= 0$

(Decomposition) $= N_{rg}$

Composition

(Search) $= (N_a - N_{a'} + 1)!$

(Management) $= N_r$

(Composition) $= \binom{N_r}{2}$

Variables

N_a: the number of all agents

$N_{a'}$: the number of all objective agents to reorganize

N_r: the number of rules which all objective agents to reorganize have

N_{rg}: the number of rule groups which all objective agents to reorganize have

N_i: the number of rule groups which objective agent-i to reorganize have

Search cost is represented by the number of cooperation paths after reorganization. Management cost is represented by the number of rules which are added newly to a KB. Decomposition cost, which needs to generate new agents, is represented by the number of agents which are generated after decomposition. Composition cost, which is needed to check rules which one agent owns and another one owns, is represented by the number of rules.

Result

Table 8 shows the results of the experiment. Reorganization occurs twice, and in the former both decomposition and composition occur at the same time. At this

Total number of search	75
Process of reorganization	
Objective agents	1. decomposition C, composition $(A, B), (C, E)$ 2. decomposition D
Comparison of costs ($COST_D$ vs. $COST_C$)	1. $362 < 807$ 2. ——
Results	1. decomposition (C_1, C_2), composition $((AB))$ 2. decomposition (D_1, D_2)
Learned cooperation path	D_1-(AB)

Table 8: Results of the experiment

time, $COST_D$ takes 362 and $COST_C$ takes 807. So, $COST_D$ is smaller than $COST_C$, and decomposition is selected. Finally, the cooperation path D_1-(AB) can be acquired as the subtask structure of unit20.

Equations to calculate costs which are used here are very simple. Calculation of costs is a key point to reorganize effectively, so better equations for $COST_D$ and $COST_C$ should be discussed. It might be meaningless to compose agents that have been spawned from one agent. Some mechanism to suppress the case will be necessary for effective problem solving. In some cases, even if the condition is satisfied, agents should not compose them nor decompose them but wait for the next chance for self-reorganization.

Conclusions

There are two possible approaches to construct a large ES. One approach is simply to construct a single large knowledge base gathering knowledge from multiple experts and acquire task structure by using some centralized learning algorithms (e.g. Genetic Algorithm, Artificial Neural Network, etc.). But the more knowledge increases, the more difficult it is to maintain consistency among knowledge. The other approach is to make smaller multiple ESs which correspond to experts and work together to solve a problem. As knowledge structure of human experts might be different, it is better for each expert to assert, to retract, or to modify its own knowledge separately. This approach enables human experts to maintain their own knowledge base. In CDES, using the latter approach, an expert can maintain only internal consistency of its own knowledge base. In addition, it can also avoid inconsistency among knowledge bases by the weight of cooperation paths.

We made some experiments of 4 algorithms of a BD analyzer and the Roulette Spin scheme, in which paths are selected at a rate of weight, was evaluated to be optimal among them. Next, we explained a scheme to learn the task structure, by changing granularity by themselves, from viewpoints of decomposing into fine grain and composing into coarse grain. Some decomposition strategies and composition strategies with Roulette Spin scheme for BD analyzer were evaluated. In decomposition strategies, Strategy 2, which decomposes the first agent of cooperation paths, was evaluated to be optimal among strategies. Meanwhile, in composition strategies, Strategy 1, which composes two agents whose positions can be replaced in two paths with lower weights, was evaluated to be optimal among strategies. Finally, an incorporation scheme with Strategy 2 of decomposition and Strategy 1 of composition was discussed. We showed that the subtask structure of unit20 could be learned by using this scheme.

In future works, we will develop a better scheme which incorporates a decomposition strategy and a composition strategy, and apply the scheme to acquire the whole task structure of the CDES.

References

Genesereth, M. R., and Fikes, R. 1990. Knowledge interchange format version 2.2 reference manual, Technical Report, Logic-90-4. Dept. of Computer Science, Stanford Univ.

Ginsberg, M. L. 1991a. Knowledge Interchange Format : the KIF of death. *AI Magazine*, 12(3): 57-63.

Ishida, T.; Gasser, L.; and Yokoo, M. 1992. Organization Self-Design of Distributed Production Systems. *IEEE Trans. on Knowledge and Data Engineering* 4(2): 123-134.

Itoh, T., Watanabe, T., and Yamaguchi, T. 1994b. Task Structure Acquisition Scheme with Cooperative Distributed Expert Systems by Self-reorganization. In Poster Proceedings of the 7th Australian Joint Conference on Artificial Intelligence, 57-64. Armidale, NSW., Australia:

Nishida, T., Takeda, H., and Iino, K. 1994b. Mediation in the Knowledgeable Community. In 1994 Spring Symposium Series "Software Agents" Working notes, 110-113.

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

Zhang, C. 1992. Cooperation under uncertainty in distributed expert systems. *Artificial Intelligence* 56: 21-69.

Zhang, M., and Zhang, C. 1994b. Synthesis of Solutions in Distributed Expert Systems. In Proceedings of the 7th Australian Joint Conference on Artificial Intelligence, 362-369. Armidale, NSW., Australia·

1991. *ART-IM Reference Manual.* Reading,: Inference Corp.