CHAPTER 7

DECOMPOSITION AND LEARNING

7.1 INTRODUCTION

In designing distributed AI systems, it is essential to decompose the problem into smaller and smaller parts so that it can be easily distributed to different collaborators of the system and be solved. Hence, decomposition is the first and important phase of distributed AI systems (Gasser 1992, Uma 1996b, Booch 1995). The problem is, how to formulate, describe, decompose and allocate tasks and synthesize results among a group of collaborators. Task decomposition is influenced by collaborator's capacity and capabilities, by dependency among tasks, by adjacency requirements, and by dynamically changing patterns of dependency. Hence, modularization is recognized to be a notoriously difficult problem. This chapter concentrates on decomposition method carried out by the Decomposition Agent (DA). In this work, the DA uses a new approach called theta_decomposition (Anandakumar 1998b) for decomposing the problem, to be solved by the distributed AI system.

The field of machine learning studies computational methods for acquiring new knowledge, new skills and new ways to organize existing knowledge. The bottleneck of any collaborator is lack of learning capability, that is, collaborators cannot handle many real world applications due to lack of learning capability. This is because, many real world applications are normally ill-structured and are highly dynamic in nature (Linger 1979, Booch 1995). Collaborators without learning capability, cannot optimize or minimize overall computation even when the same/similar task is encountered. Hence, in this work, each collaborator is incorporated with learning techniques, and is carried out by the Incremental Learning Agent.
7.2 DECOMPOSING AGENT (DA)

In distributed AI, many of the research work assumes that the given problem is already partitioned into smaller modules and concentrates on the other issues (Uma 1996b). This assumption is removed in this work, and the decomposition is based on many factors. In addition to considering aspects of the problem while decomposing, the work also considers personality parameters (see chapter 4), and the nature of cooperative control among collaborators.

When the collaborator encounters a problem to be solved, the Control and Analysis Agent (CAA) of that collaborator decides whether decomposition is necessary or not. If it is necessary, the CAA invokes DA for decomposition. The functions of the DA are: i) Invoking theta_decomposition procedure (Anandakumar 1998d) to decompose the given problem, ii) Attaching description to each decomposed task, and iii) Collecting all subtasks from theta_decomposition procedure and sending them to Mind Protocol Agent (MPA) for allocation (see chapter 6) among the core group members. It is assumed that, each collaborator is attached with a Knowledge-base, which consists of background knowledge and probabilistic knowledge gained by past experience or from others, or from some external source. Also, it consists of a set of rules for manipulating probabilities (Winston 1993, Heckerman 1995). The background knowledge plays an important role in theta_decomposition method. It can take, number of different forms; knowledge gained from previously solved similar problems, behavior, self-confidence, etc. The incorporation of domain knowledge can improve efficiency by narrowing the focus of decomposition and allocation (Yagnanarayana 1998, Frawley 1991). The following section explains the theta_decomposition method in detail.
7.2.1 Theta Decomposition Method

Most of the decomposition methods work with the basic assumption that the problem solving process always operates from the beginning of the problem space and that the direction of problem solving is decided and fixed a priori. For example, in distributed search method (Durfee 1991), the search always starts from either the top or the bottom of the search space. In graphics applications, Quad-tree (Dyer 1980) approach is used as a standard approach, and by default the process starts from top-left corner of the image space (see figure 7.1). In the quad-tree approach, the problem space is divided into four different subspaces, which may be then further divided into four subspace and so on. The arrow (→) mark (figure 7.1) indicates the process direction, even in nth level, the process directions are the same.

Figure 7.1 Conventional Approach
The theta_decomposition method removes the basic assumptions regarding initial position and a priori fixed direction used in the above methods. It starts from the center of the problem (i.e. eye of the problem) space or search space, and proceeds in a chosen direction. This introduces added flexibility to the decomposing mechanism by allowing both initial position and direction of decomposition to be modifiable. Hence, allowing the task decomposition to be based at spaces in the domain where there is a concentration of task processing. The choice of the center point of decomposition is based on background and/or heuristic knowledge about the application domain. This choice, ensures that the heavily loaded areas of the problem space will now be decomposed and solved distributively. The selection of eye of the problem increases the chances of encountering solutions faster. Figure 7.2 illustrates how theta_decomposition method decomposes the problem space, and the process direction. Here, the process direction is ad hoc, it means that, it is left to the collaborator to follow its own direction. The radius R is application dependent, and is drawn from the chosen center point of view to cover the entire problem solving region. The value of R may be given or may be problem_specific or may be assumed heuristically. It allows the decomposition size (theta or angle) to vary. Figure 7.3 shows theta (θ) of different size. For example, in case of graphical application, the decomposition size represents the angle, more the angle more is the area covered.

Figure 7.2 Theta decomposition Approach for n = 6

Figure 7.3 θ of Different Size
In this thesis, the features of the theta_decomposition method are highlighted through the Global Software Development (discussed in the chapter 3) and the Graphics Application (Character Recognition).

7.2.1.1 **Theta_Decomposition in Global Software Development**

Previous chapter explains how the core group is formed using the Mind Protocol Agent (MPA). Once the most promising work group is formed, the next step is to decompose the given task. In order to divide the task effectively, that is, to the taste and capacity of the group member, the DA uses the background knowledge of the core group members (discussed below). If the required knowledge is not available, then the collaborator may request for it and get it from the particular collaborator.

7.2.1.1.1 **Background Knowledge**

The collaborative software development process is gaining interest (Lijter 1996, Jerry 1999). But, the question remains: which collaborator is best suited for the modules at hand. Hence, in this work, the DA collects, Bio-graphical profiles (prepared by Data Analysis Agent), and the personality parameters (described in chapter 4) of the members. As already discussed in chapter 4, the Biographical profile consists of following details: experience, number of projects solved, type of each project, skill list etc. The personality parameters like capacity, relationship between collaborators, percentage of similarities of current task with already solved subtasks, load, willingness etc., represent the collaborator’s taste or capabilities. These information act as a background knowledge while decomposing the problem, by the DA.
The DA of the collaborator can divide the task into two different ways. The simplest way of dividing is to partition the problem space irrespective of other collaborators' personality parameters, i.e., dividing the problem space equally (by number of collaborators available or group size). But, this may not be effective in all cases. The alternative to this, is to consider collaborators' personality parameters also while decomposing. In this work, the angle represents the different properties of the development process. For example, responsibility of the entire development may be represented by the individual collaborator's responsibility, in which case, more the angle more is the responsibility. The DA uses heuristic values and the number of collaborators available for cooperation in deciding the quantum (theta or angle) of the task. It is mainly based on the relationship of a collaborator, and the similar tasks it has already tackled. In this application, the problem is divided into four modules (the environment is simulated) based on the skill (see figure 7.4) and the DA prepares the modules description.

Fig.7.4 Decomposition based on Skill
The modules details are given below

**Model 1: Catalog design (Book)**

Description: The task is to design book catalogue, compatibility with Web technology. Designed catalog will be published on the Internet.

- **Required skill**: Multimedia, and Internet technology
- **Desired skill**: Image processing, and HTML.
- **Duration**: One Month
- **Dependency**: module 2 and module 3

**Module 2: Web publishing**

Description: The task is to create and manipulate Web pages, publishing catalog on the Internet.

- **Required skill**: Web development technology
- **Desired skill**: Java/HTML/Perl.
- **Duration**: One Month
- **Dependency**: module 1 and module 3

**Module 3: Hardware Interface**

Description: The task is to develop hardware interface. The order can be placed via telephone, or fax, or e-mail.

- **Required skill**: Hardware technology, Windows '95/98/NT, Method
- **Desired skill**: C/C++, and GUI
- **Duration**: Two Months
- **Dependency**: module 2 and module 4

**Module 4: Order processing module**

Description: Order receiving, and processing,

- **Required skill**: Windows 95/NT,
- **Desired skill**: C/C++
- **Duration**: One Month
- **Dependency**: module 1,2 and module 3
7.2.1.2 THETA_DECOMPOSITION IN IMAGE PROCESSING

The image processing has many applications like pattern recognition, medical application signature analysis etc. Here, Character Recognition application is considered, and is solved distributively. That is, the problem space is divided into number of subspaces and are distributed to the different collaborators. They individually solves the problem, and finally the results are integrated. The Quad-tree (Dyer 1980) approach is used as a standard algorithm for processing images. The figure 7.5 and 7.6 compares the quad-tree approach and theta_decomposition method.

![Figure 7.5 Quad-tree Approach (Level - 1)](image1)

![Figure 7.6 theta-decomposition (n=4)](image2)

The space covered under R (given) is divided depending on the number of collaborators (n = 4 in figure 7.2) involved in solving a problem. Here, the task is divided into equal angle (θ) size subtasks. It is possible to choose different values of theta for dividing the task into subtasks of different sizes. This may depend on the collaborator's capability, load etc. If the collaborator finds the problem is too large, it intum calls for further decomposition. The sub problem description is as shown below.
Here each collaborator will move in its own direction depending on space specified by arrow (→).

7.2.1.2.1 Background Knowledge in Character Recognition

In this application, by default the problem space or search space covered under radius R is divided logically into 4 subspaces of size \( \theta = 90^\circ \) and are named as 1st_quadrant, 2nd_quadrant, 3rd_quadrant, and 4th_quadrant. The thick dashed line in figure 7.2 shows the logical division. Hence, the minimal domain knowledge that each collaborator should have is

Fact 1 : There are 4 quadrants in one complete cycle
Fact 2 : Each quadrant is of theta (\( \theta \)) = 90°
Fact 3 : The 1st_quadrant is between \( \theta = 0 \) and \( \theta = 90^\circ \)
Fact 4 : The 2nd_quadrant is between \( \theta = 90^\circ \) and \( \theta = 180^\circ \) etc.

The following knowledge may or may not be available in each collaborator.

Rule 1 : If theta belongs to 1st_quadrant then Proceed from left-bottom of the space.
Rule 2 : If theta belongs to 2nd_quadrant then proceed from right-bottom of the space etc.

In this example, the domain knowledge is, strokes or features of the alphabetical letters. For example, for letter 'A', there are three strokes/features. These features are assumed to be input or available in the local data base. The heuristics used by the DA in deciding the quantum (\( \theta \)) of task and the number collaborators available for cooperation, is based mainly on the probability of a collaborator solving a subtask, and the similarity of tasks it has henceforth tackled. As the collaborator gains
experience, the background knowledge is added with new learned knowledge and subsequently the probability values are updated using Baye's formula (Winston 1993). This manipulation plays an important role in choosing the appropriate angle for choosing the starting point of the problem space or search space more intelligently, and in identifying which collaborator is appropriate for the current sub-tasks. The knowledge gained by the DA is sent to Incremental Learning Agent.

### 7.3 INCREMENTAL LEARNING AGENT (ILA)

Learning is an essential ingredient in the intelligent behavior of any system. The reasons for interest in machine learning are: (i) A better understanding of learning process might allow the automation of the system, (ii) It helps in better way of handling new situations and lastly, (iii) modeling human learning mechanism. In the global software development, the ILA uses three different learning techniques at three different situation. The situation is as shown in figure 7.7.

#### Figure 7.7 Learning Techniques in different situations
They are, ii) Action-Research technique (Winter 1989, Anandakumar 1998a) while developing the software product, i) Probabilistic inference (Buntine, 1996) and rule-induction (Yasdi 1991, Roddich 1996, Anandakumar 1997b) at the core group formation, and iii) Linguistic summary (Yager 1991) after the problem is solved. These methods are explained one by one in the following sections.

7.3.1 Action-Research Learning Method

Unfortunately, most of the live data is imperfect, garbled and dynamically changing in nature. Due to this factor, the difficulty of distributed problem-solving task increases in four ways: a) Erroneous data or knowledge, b) Dynamically changing data, c) The number of possibilities to evaluate and d) complex procedure for ruling out possibilities. Therefore, there should be proper interaction among the collaborators (Rosenchine 1982) Learning is intellectually hard work, and that at best comes about through an incremental and iterative process. The human beings will acquire or refine their skills through experience (that is step by step or incremental). First the problems are solved on an ad hoc basis. As experience accumulates, the useful solutions are understood more systematically, this in turn enables a more sophisticated level of practice and allows problem solver to tackle harder problems (May 1989). As Vonk (1990) defines it (incremental development), "in the case of incremental development, the system as a whole is built up step by step, and each successive version consists of the previous version unchanged plus a number of new functions". Incremental methods are needed to efficiently keep pace with changes in methods and information (Frawley 1991, Durfee 1988, Millan 1996). This motivates the work of incremental learning. In this work, the collaborators interact in a Functionally Accurate, Cooperative distributed system (Lesser 1981) manner, while developing the software product. The overview of Functionally Accurate Cooperative (FA/C) distributed system is explained below.
7.3.1.1 Overview of FA/C Systems

In conventional distributed systems a collaborator (problem solver) rarely needs the assistance of another collaborator in carrying out its problem-solving functions. Such type of systems are called complete accurate, nearly autonomous (CA/NA) (Lesser 1981), because each collaborator's algorithms operate on complete and correct information ("completely accurate") and because each collaborator has in its local database i.e., the information it requires to complete its processing correctly ("nearly autonomous"). The CA/NA approach, however, is not suitable for distributed system applications in which algorithms and control structures cannot be replicated or partitioned effectively so as to match the natural distribution of data in the network. In this situation, a CA/NA system is expensive to implement because of the high communication and synchronization cost required to guarantee completeness and consistency of the local databases.

Lesser et al. (1991), in their approach, the distributed system is structured so that each node (collaborator) can perform useful process, using incomplete input data while simultaneously exchanging the intermediate results of its processing with other collaborators to construct cooperatively a complete solution. Such systems are called Functionally Accurate (FA), because it exhibits acceptable system input or output behavior but is distinct from Completely Accurate, in which all intermediate results shared among subtasks are required to be correct and consistent. The cooperation among the problem-solvers is needed to eliminate incorrect intermediate results and to converge to a complete and consistent solution. Therefore, such systems are called Functionally Accurate and cooperative (FA/C). The FA/C model is further improved by the authors. He incorporated more intelligence; i) Distributed searching (Durfee 1991), ii) Distributed planning (Durfee 1986, Durfee 1987a, Appligate 1990) and iii) Partial global planning (Durfee 1987), to deal with uncertainty and error in input data and knowledge.
7.3.1.2 Incremental Action-Research

In this work, the performance of the FC/A system is improved by incorporating incremental Action-Research learning method (Anandakumar 1998). Each collaborator communicates each other in FA/C manner to find single global solution. An incremental Action-Research learning method introduced here, helps in improving the quality of results and is a continuously developing sequence. When a collaborator generates or receives a subgoal (partial results), it is compared with the locally available subgoals. The ILA gains knowledge after every communication. The ILA improves the performance of the system through phase 1: similarity study (by Similarity Analysis Agent) and phase 2: Accumulation and updating the learned knowledge. While developing the software product, quite often the collaborator exchange their information (partial results/ subgoal/ reliability values). Whenever, the information is received, the ILA invokes the Similarity Analysis Agent (SAA), for similarity analysis.

In similarity study, the SAA does a comparative study of current problem with previously encountered problems in order to tackle the current problems based on similarity aspects. Similarity analysis considers the set of already solved problems and gathers new information including what type of problems the collaborator has already solved, abstracts, other collaborators interaction etc. The ILA receives these outputs for further process. The knowledge gathered in similarity study is accumulated and modified in the phase 2. The Action-Research learning method uses the past experience stored in the knowledge base and will help the ILA to take better decision regarding the type of action to perform (send/request for data/results), the appropriate collaborators to contact for help, plan future activities and modify/monitor the actions taken currently. The ILA gives high priority for frequently solved problem during search. This is true in case of human beings.
also, i.e. while assigning a task to a particular worker, most of the time same/similar tasks are assigned to the same worker (assuming that the workers details are known) in order to obtain faster solutions.

The Action-Research method unifies goal-oriented and data-oriented approaches (Corkill 1985). Apart from these two approaches, when a Collaborator generates the same or similar subgoal or receives the same or similar subgoal instead of solving it again, the collaborator will send only the pre computed results of that subgoal to other collaborators (requested) or substitute with the pre computed values (if it is local). In order to control the various activities and to make the response faster the ILA makes use of set of heuristic variables called progressive heuristics. This iterative refinement helps in making powerful decisions in the group and reduces the communication cost. This is the significant difference of the proposed method over FA/C system.

7.3.1.3 Accumulation and Updating of Knowledge

As and when the collaborator learn new facts or knowledge, it will be added into the knowledge base incrementally. This method accumulates and updates the knowledge, as and when ILA receives the information at each collaborator. This learned information helps to monitor, revise and interleave the activities in exchanging and computing of partial tentative sub goals or results. The accumulation and update procedure is given below:

ALGORITHM B : Procedure for knowledge accumulation and update at collaborator Ai
INPUT : Learned knowledge set NL
OUTPUT : Refined knowledge
Method

Let KB be the local knowledge base and lk is the fact

Step 1  :  For all lk∈NL do the following
Step 2  :  if (lk is in KB) then discard lk as it is already present.
Step 3  :  if (lk partially similar to knowledge kj (KB) then update the knowledge kj and put back to the knowledge base KB
Step 4  :  if (lk not in KB) then append KB = KB ∪ lk
Step 5  :  end for
Step 6  :  Procedure ends

The learned knowledge and goal processing then work in tandem to improve problem-solving performance, as it pursues, monitors and receives learned knowledge. The collaborator is now capable of identifying where the improvement can be made in computing and propagating partial results and selectively contacting other collaborator. This accumulated knowledge helps in removing basic assumptions or constraints step-by-step or incrementally. This gain in new knowledge and experience helps the collaborator to tackle the similar or repeated problem faster, to handle new situation and make good decisions on the choice of other collaborators.

7.3.2 Rule-Induction Method and Probabilistic Inference

The ILA uses rule induction based on functional dependencies. Let F be a set of functional dependencies (Roddick 1996, Anandakumar 1997b), in which the following inference rules are true.

i) If X is a set of attributes and Y ⊆ X then X → Y
ii) If X → Y and W is a set of attributes then WX → WY
iii) If X → Y and Y → Z then X → Z
iv) If X → Y and X → Z then X → YZ
v) If X → YZ then X → Y and X → Z and
vi) IF X → Y and WY → Z then XW → Z
The following examples are with respect to the application considered in this thesis.

**Example 1:**

C1 is 'CLOSEFRIEND' of C2 and C2 is 'CLOSEFRIEND' of C3.

Then, C1 is 'CLOSEFRIEND' of C3.

**Example 2:**

C1's Behavior is GOOD and ALWAYS helps. Then C1's Behavior is 'GOOD' and C1 'ALWAYS' helps etc.

Further, probabilistic inferences are drawn from the trust factors. The relationship graph (chapter 6) can be treated as a probabilistic network. ILA uses Baye's rule (Winston 1993, Dagum 1993) to infer the dependency relation among the collaborators. For example, from figure 6.1, a relationship link as C₁→C₂→C₃→C₄. Now, from the rule of probability we can find p(C/C₄).

\[
p(C/C₄) = \frac{p(C/C₂)}{p(C₄)} = \frac{p(C₁, C₂, C₃, C₄)}{p(C₁, C₂, C₃, C₄)} \tag{7.1}
\]

where p(C₁, C₂, C₃, C₄) is the joint distribution determined from the Bayesian network (Heckearman 1995). The above equation is rewritten as

\[
p(C/C₄) = \frac{p(C₁, C₂, C₃, C₄)}{p(C₁, C₂, C₃, C₄)} = \frac{p(C₁) \Sigma p(C/C₁) p(C₂/C₃) p(C₄/C₃) p(C/C₄)}{p(C₁) \Sigma p(C₁) p(C₂/C₁) \Sigma p(C₃/C₂) p(C₄/C₂) p(C/C₄)} \tag{7.2}
\]
This gives the dependency relation. This probabilistic inference is used to find the behavior of unknown collaborators. Further, the Baye's theorem is used while making decisions like sending requested information, deciding to help, to compute willingness etc., and to update the probabilistic information. The learned knowledge is stored in the knowledge base for future use.

7.3.3 Linguistic Summaries

The ILA, after receiving all the learned knowledge, it produces the impressions. It infers information from gained information and prepares summaries of it. These summaries are known as impressions. A new approach for producing summaries, is suggested by Yager (1991). It summarizes the data in terms of three values: a summarizer (S), a quantity agreement (Q), and a truth value (T). The concept is explained with an example, let LTFV = {0.8, 0.4, 0.6, 0.9, 0.6, 0.7, 0.8, 0.65, 0.8} and it can hypothesis the summary: S = about 0.7, And Q = most. Then T, obtained by a procedure determined by Yager (1991), indicate the truth of the statement, "most LTF in D are about 0.8". Then, T = TRUE. In the application, summary impressions are produced, and are listed below.

7.4 CONCLUSION

In this chapter, the properties of theta_decomposition, and Action-Research, rule-induction, and probabilistic inference learning techniques are highlighted. The chapter shows the importance of background knowledge while decomposing the task. It also highlight the effect of personality parameters in dividing the task. Finally, impressions are formed, that is, linguistic summaries are produced.
BEHAVIOUR (Linguistic Summary)

C₁ Behavior is always POOR with probability = 0.333
C₁ Behavior is always AVERAGE with probability = 0.167
C₁ Behavior is always GOOD with probability = 0.500
C₁ Behavior is always VGOOD with probability = 0.667

C₂ Behavior is always POOR with probability = 0.667
C₂ Behavior is always AVERAGE with probability = 0.500
C₂ Behavior is always GOOD with probability = 0.333
C₂ Behavior is always VGOOD with probability = 0.167

C₃ Behavior is always POOR with probability = 0.667
C₃ Behavior is always AVERAGE with probability = 0.500
C₃ Behavior is always GOOD with probability = 0.333
C₃ Behavior is always VGOOD with probability = 0.167

C₄ Behavior is always POOR with probability = 0.500
C₄ Behavior is always AVERAGE with probability = 0.333
C₄ Behavior is always GOOD with probability = 0.667
C₄ Behavior is always VGOOD with probability = 0.167 etc.,

LOCAL TRUST FACTOR (Linguistic Summary)

C₁ self-confidence is always POOR with probability = 0.167
C₁ self-confidence is always GOOD with probability = 0.833
C₁ self-confidence is always VGOOD with probability = 0.500
C₁ self-confidence is always EXCELLENT with probability = 0.767

C₂ self-confidence is always VPOOR with probability = 0.167
C₂ self-confidence is always POOR with probability = 0.167
C₂ self-confidence is always AVERAGE with probability = 0.333
C₂ self-confidence is always GOOD with probability = 0.333
C₂ self-confidence is always VGOOD with probability = 0.500
C₂ self-confidence is always EXCELLENT with probability = 0.767 etc.,