Studies on Communication Strategies in Cooperative Search

楢崎, 修二

https://doi.org/10.11501/3111011

出版情報：九州大学, 1995, 博士（工学）, 論文博士
バージョン：
権利関係：

Studies on Communication Strategies in Cooperative Search

植　崎　修　二

Studies on Communication Strategies in Cooperative Search

植　崎　修　二

Studies on Communication Strategies in Cooperative Search

Shuji Narazaki

1996

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The speed of compters has been increasing constantly. However, even now, it is not sufficient for building Artificial Intelligence (AI) systems, especially for a real-time self-controlled robot.

The brain of a human being that consists of about 18 giga cells working in parallel, is more intelligent than the artificial brain on any computer, though a cell is thought to have less computational power than the current VLSI chip. Thus, a bunch of processors overcomes a single, complex processor. This idea is well known and is used as parallel/distributed processing. The purposes of parallel/distributed processing can be summarized as follows:

functional distribution to solve a comlex problem

load distribution to execute a program efficiently

geometrical distribution to solve problems that are distributed over a wide area.

Since, in a lot of areas of application, computational power cannot be used satisfactorily, the computer hardware, operating systems and languages for parallel processing have been researched and developed.

The same may be said in AI. The introduction of parallel processing into AI was made in the latter half of 1970's. *Distributed Hearsay-II* [LE80] was the first speech-recognition system that used parallel architecture in order to achieve a high-performace. The recognition process is divided into some stages and each stage consists of some processes that run in parallel. It elucidates the difficulty of paralleled AI processing; since the flow of a job depends strongly on

incoming data, it is difficult to determine the order of a job and to decide the flow of information before its execution. In fact, the first implementation of Hearsay-II, that was centralized, used heuristic-based schedulers to control the order of process activation. Since distributed Hearsay-II uses physically partitioned communication media in order to avoid bottle-necks at a single communication medium, each process has to work with local information under the limitation of the communication bandwidth. The scheduling problem became the problem of selecting information to be sent. Localization of communication and cooperation between tasks were required.

Distributed Hearsay-II is a system with a single input flow, multiple processing units, and a single output flow. On the other hand, a distributed monitoring system that was an application of AI was a system with multiple input flows. It merges local information from sensors that are distributed physically, and generates a global, rational view of the sensed regions. This application requires distributed processing, since the inputs are inherently distributed, and since the merging task is also distributed for reducing communication cost.

The most important issues in these applications are:

1. making sure that the information in each processing unit is sufficient in quality and quantity to solve the problem locally, and.

2. addressing the fact that the exchange of all local information between processing units is imposible because of the existence of communication cost.

Namely, each processing unit that is called an *agent* has *bounded rationality*. The term 'bounded rationality' means an individual should make a decision rationally with only incomplete pieces of information and with limited time to select his action. It is used in economics and in organization-theory to determine the origin of the forms of organizations.

A realistic answer for this problem is building dynamic relations among agents by exchanging only some pieces of local information among each other, as long as it leads to achieving a better answer. In other words, making an appropriate organization of agents is required. This is the root of *Distributed Artificial Intelligence* (DAI).

In the middle of the 1980's, DAI has been thought of as consisting of two sub-research areas:

Distributed Problem Solving (DPS) and *Multi-Agent Systems* (MAS). In DPS, agents have a shared goal. Each of them execute subgoals simultaneously. They exchange their partial results in order to maintain consistency or achieve better performances. The distributed constraint satisfication problem, that contains distributed search problem, is a classical sample problem of DPS.

On the other hand, MAS consists of autonomous agents. Each has its own goal. To complete their goals, they should cooperate with each other and should have social rationality. For example, managing shared resources or planning their own behaviors is required. In both systems, agents should change the relations between agents dynamically. Since the goal of MAS is the *executability* rather than performance, which is the goal of DPS, we will discuss DPS systems mainly in this thesis.

In both areas, *cooperation* means to select an appropriate organization dynamically by exchanging a piece of local information with each other. Thus cooperation is a kind of meta-level computation that controls problem-level computation done by agents. It controls the number of agents, roles of each agent, communication topologies and frequencies, messages among agents, and so on. The optimal form of computation changes dynamically, therefore agents must choose an adequate form of cooperation during execution in order to achieve the best performance. The selected form would be better that other ones. Classes of these forms even include sequential or centralized computation forms, i.e. smart agents choose the centralized computation if they find that it is better than the decentralized.

Therefore, cooperation requires acquiring information on other agents. Usually that is done by communication[1]. A property of this kind of communication is that it is *optional*: the necessity of it depends on its situation. Its goal is making a model of an agent's environment to determine the form of organization. The agent can determine the frequency or the number of receivers, as long as he can build a sufficient model. Thus even if an agent requires broadcast for an optional communication in a situation, he would need only one-to-one communication for same optional communication in another situation. Optional communication for cooperation should be controlled by a strategy that uses a local model of its environment.

[1] An example of the exceptions is agreement mechanism based on game-theoretic approach [RGG86], in particular the focal-point approach [FKR95], which does not require explicit communication.

We think that the existance and the neccesity of a local model of an agent's environment is the most difference between DAI/MAS and usual parallel/distributed processing. Some researchers have defined an agent as "the system that requires the terms like belief, utility, and orientation in order to avoid over-complexity in describing its behaviors" [Ish95]. A part of its complexity comes from the sensitivity of agents against changes in their environments. Agents collect information concerning their environments in order to behave rationally. But it is not a trivial task.

Our interest could be described as to make a general communication strategy for DPS, especially for the systems that consists of identical agents. In DPS, more amount of knowledge about the circumstances leads to better behaviors of the agents, but sharing knowledge causes more communication costs. Therefore, it is crucial to consider the *utility* of communication to exchange the local knowledge of each agent. The term "utility" comes from economics and the game theory, to describe the *quality* of the result of an action done by an agent. Using utility of communication will provide a different approach to that used in the distributed Hearsay-II. Since the localizing method used in it was based on the heterogeneity of processing elements, we can not use their idea for homogeneous agents. Utility is a good measure for describing an agent's environment, even if agents are homogeneous.

If communication costs could be negligible, sharing all information among all agents would be the best for cooperation. But as communication costs are much higher than computation costs in real distributed systems, none of the agents can have a global view any longer. Instead, each agent has its onw local view only, and there is just partial consistency of information shared. How local the view of an agent is depends on the utility and the cost of communication among agents. But because its view is local, it can not find the optimal form of communication.

We consider cooperative search, such as the distributed vehicle monitoring, as a typical example of distributed problem solving. Agents running in parallel perform a search procedure. They can reduce the amount of their own tasks, or improve the quality of search results by exchanging some information about models of themselves, the circumstance, and/or intermediate results. But in the distributed situation where communication costs are not negligible, the wider and the more frequent communication is, the worse the whole performance is. Therefore

deciding when and with whom to communicate is crucial.

Finding an adequate structure of agent groups is an important issue in DAI [GRHL89, DLC87b, IGY92, Gas92a], and there has been some researches in this matter [BG88, Huh87, AG92]. For example, agents can change the utility of communication by changing the *connectivity* of them, i.e. the abstractness or quantity of information, the number of receivers, or the frequency of communication. They determine the social structure of agents. Most research has addressed the effects of the abstractness. But few strategies that change the structure dynamically have been proposed.

In this thesis, we describe some new communication strategies for cooperative search [NYY94]. Under the assumption of the homogeneity of agents that means every agent has the same computational power, our strategies estimate the real amount of information based on the history of the local revision of information. Agents control communication costs by changing the number of the receiver of multicasts, or by changing the frequency of the multicast. Consequently, programmers need not select an efficient communication pattern at programming time. Furthermore, we propose a way to describe a strategy and a problem separately. Our method uses meta-level computation for it. Its implementation is described later in this thesis.

The structure of this thesis is as follows.

In Chapter 2, we summarize researches on cooperation strategies that changes the quality of communication. We mainly disscuss Distributed Vehicle Monitoring Testbed, strategies for the pursuit problem, field-based communication languages, and studies on cooperative search. From our point of view, most studies fall into three categories: abstractness control, communication-range control, and frequency control. But the last two control methods to which our proposal strategies belong, have not been well researched.

Chapter 3 describes our three strategies for cooperative search. The first two strategies change the spatial connectivity among agents. The last one changes the frequency of multicasting among agents.

Mathematical analysis shows the quality of proposed strategies. First, we investigate the existence of a peak in the performance if changes in shared information are distributed uniformly. It leads to the need of communication strategies. Second, we show that the phase-transition

between flat relations and hierarchal relation on spatial structures are required in some cases. Thus we provide a protocol for changing the relations.

Here, we also mention the means for modeling other agents in our strategies. An agent uses its own execution history as a model. This means that we think of parallel searches on agents as a kind of ergodic process. Using the local history, an agent forecasts the utility of communication before an interaction. In a cooperative search, an execution history can be built from the history of the renewal of some hint for search, for example, search-space threshold. Thus, building an execution model in our strategies is not expensive. A simple game shows the effectiveness of this method. Agents change the number of receivers (range of communication). It fits well even if each history can not hold the exact model since the length of the history is bounded.

In Chapter 4, we evaluate the two strategies through the Traveling Salesman Problem simulations. The Traveling Salesman Problem is one of the typical examples of cooperative search. Agents exchange thresholds with each other. The strategies control the way to exchange them. The result shows the strategies brought good performance. The quality merely depends on communication costs and the initial relations. Agents' connectivities are controlled rationally by the strategies.

We also discuss the adaptability of the strategies. We claim they can be applied to the A* search well. And we expect that our method is not only useful for cooperative search problem but also for the other domains. We mention the management of a shared object with strong consistency on the distributed systems and self-organization in genetic algorithm systems. History-based cooperation mechanism is a good way to build a model of environments.

Section 5 states another issue of this thesis that is an implementation that makes the reuse of strategies easy. The communication strategy should be independent of application programs. We found that the metalevel computation mechanisms in some object-oriented programming languages (OOPL) are useful for code separation. Using it, we can separate communication strategies from problem-level programs easily. Thus programmers' task is divided into two independent subtasks. Library designers provide metalevel libraries for communication strategies, while application programmers focus their attentions only on problem-level programming.

and declare communication strategies as emta-class objects. This approach is a new way to use the power of the metalevel computational in the DAI domain.

Our current implementation on the Common Lisp Object System (CLOS) Meta-Object Protocol (MOP) provides a general framework for writing cooperation methods based on a particular variable's history. We also give sample codes for TSP, shared objects with strong consistency, and discuss the merits and limits of this approach.

Chapter 6 is the conclusion and future directions of the research. We discuss the form of cooperation on parallel systems on which we can ignore the communication costs. After showing the need of cooperation on parallel systems, we will give a means to modify our strategies for the requirements of such systems.

Chapter 2

Communication in Distributed Problem Solving

Since one of the goals of DAI is building an efficient organization of autonomous processing units (agents) on distributed environments [BG88, DM91a], communication among agents is a crucial issue of DAI. In this thesis, we focus our attention on distributed search. Search is a fundamental problem in Artificial Intelligence, and covers a wide area of the issues in DAI system [Les90, Yok95]. The processing styles of distributed search fall into two categories. In the first category, the search space is divided into some sub-spaces that are managed by dedicated heterogeneous agents respectively [SRSF91, KG94, YDIK92, CL88, HB91]. Cooperation among agents is crucial to solving the problem. In the other case, each agent is autonomous: it has the ability to solve the problem without the others' cooperation. Since our interest is in the organization of uniform members, we will discuss the latter.

Now we will give a summary of the previous works, especially in DPS, in the next section . After that, we will categorize them into three types and introduce a measure of the rationality of communication.

2.1 Previous Works

Communication is crucial for agents in oder to decide their behaviors in dynamic environments. It is a primary way of knowing an agent's environment. Since communication costs are not negligible in real computational environments, we must give a smart strategy for efficient

communication for agents. Some DAI researchers stood on the same position. Though these are not proposals for a realistic strategy, it would be useful to make a survey.

2.1.1 distributed vehicle monitoring testbed : flat structures

Distributed monitoring or distributed interpretation is a classic example of DAI. Durfee et al. have used it as a platform for testing the quality of communication strategy [DLC87b, DM89, Dur88]. Their framework is called *Distributed Vehicle Monitoring Testbed* (DVMT). The system consists of nodes. Each node has a sensor whose range is bounded. The search space, the area to be searched, is divided by the ranges of sensors that would overlap each other. The search space and the ranges are pre-determined before the execution. The goal of the system is to build the paths of moving vehicles in the search space (Figure 2.1). Since each vehicle passes through some of the sensors' ranges, and the sensor's inputs would contain errors

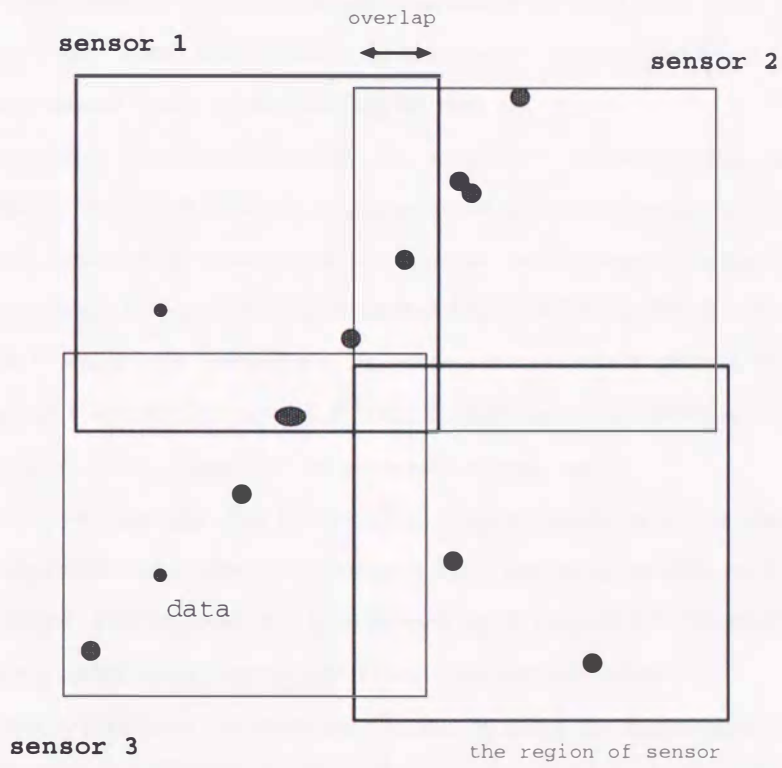


Figure 2.1: DVMT environment

like ghosts or noises, each node should exchange information each other in order to make more rational outputs.

For this reason, communication between nodes affects the quality of answers. However, due to the existence of noises, complete knowledge, itself, does not lead to the correct answer. Even if an agent could be isolated, it could still return an answer. Thus we can say that the system is built in a *functionally accurate, cooperative* (FA/C) manner [LC88, Les91].

The job of each node can be divided into the following stages:

1. getting output from a local sensor,

2. building a candidate path from the sequence of outputs, and

3. combining reliable paths in order to build overall paths and delete unreliable outputs.

Nodes repeat these stages on demand. Information sent from other nodes affects local decision making in each stage. Nodes should select the information to send and select the path to be combined that is urgent based on the information that has arrived.

On such a setting, they have measured the number of communications while changing information that is exchanged. The important result of these experiments is that the selection (or reduction) of information to be sent achieves better performance than the exchange of *all* information among the nodes. Table 2.1 is quoted from [DLC87b]. Three environments were tested; Normal, Overlap, and Twice-Data. And three communication policies were used; Send-all, Loc-comp, and F-and-L. The caption of Table 2.1 describes these meanings. The differences between them affect search plans that are generated by each agent.

The result can be summarized as follows. First, higher-abstracted information about paths of vehicles is desirable, for it has more certainty than raw data, as long as a node sends it in a timely manner. Second, sending a plan of each agent propels role-differentiation, namely reduction of the number of duplicated executions of an identical task.

But regrettably, they have not proposed the way to select the appropriate abstractness of information dynamically. And the number of sensors was fixed to 4. Therefore, we can not discuss a large scale sensor system on which we can not ignore communication cost or delay.

Table 2.1: A Summary of Communication Performance in DVMT

Environment	Communication Policy	Solution Time	Total Transmitted Hyps
Normal	Send-all	49	25
	Loc-comp	45	10
	F-and-L	49	20
Overlap	Send-all	52	16
	Loc-comp	39	11
	F-and-L	46	16
Twice-Data	Send-all	107	74
	Loc-comp	70	17
	F-and-L	99	29

Legend

Environment:	The simulated environment
Solution Time:	Earliest time at which a solution was found
Total Transmitted Hyps:	The total number of hypotheses transmitted

Send-all sends all hypotheses (candidates for paths). Loc-comp (locally-complete) communication policy transmits only hypotheses which it can not improve upon. F-and-L (first-and-last) is essentially the Loc-Comp policy with the added stipulation that if a hypothesis that is eligible for transmission does not incorporate any previously communicated hypotheses, then transmit it without delay. The overlap environment has larger overlapped areas. Twice-Data environment has the same overlap as the Normal, but twice as much data.

Finally, the roles of nodes are homogeneous: the effects of a hierarchical structure were not measured.

2.1.2 pursuit problem

The *Pursuit Problem* is also a classical example that requires cooperation among agents to solve the problem. In this problem, there are four pursuing agents (*pursuers*) and an escaping agent (*an escaper*). They are located in a two-dimensional space. The goal of the pursuers is to locate themselves at the four neighbors of the escaper. See Figure 2.2. The hatched circle in the center of the left figure is the escaper, and the other four circles are the pursuers. This configuration is known as Benda's scenario 1 ([BJD85]). Every agent moves one block (eight-neighbors) at every step. The right figure is the goal state.

Since this goal is not decomposed to four independent subgoals that are to place itself in *a* neighbor of the escaper, all pursuers should cooperate with each other, as all of them place themselves at appropriate neighbors of the escaper. And since the escaper moves randomly or smartly, four agents continue to cooperate during the pursuit.

Research directions are summarized as follows:

1. Make an appropriate organization like an order-obedience hierarchy or one for information flow [BJD85, SM89, LR92, RB89] . Measurements are mainly the completeness of

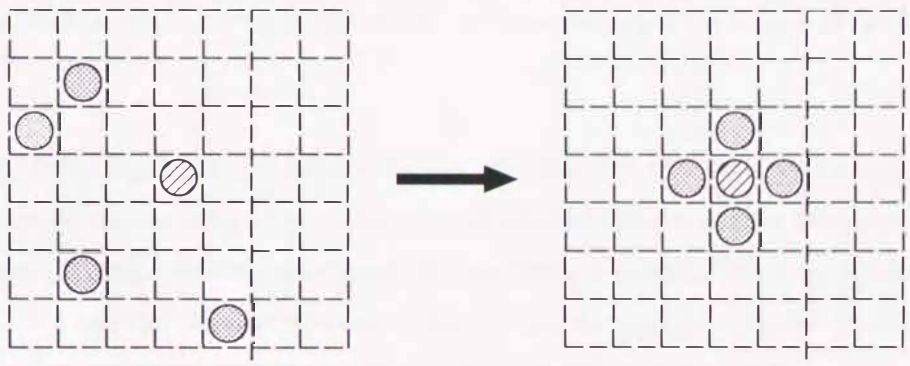


Figure 2.2: Pursuit Problem
The left figure shows an initial state that is known as Benda's scenario 1. The right one shows its goal state.

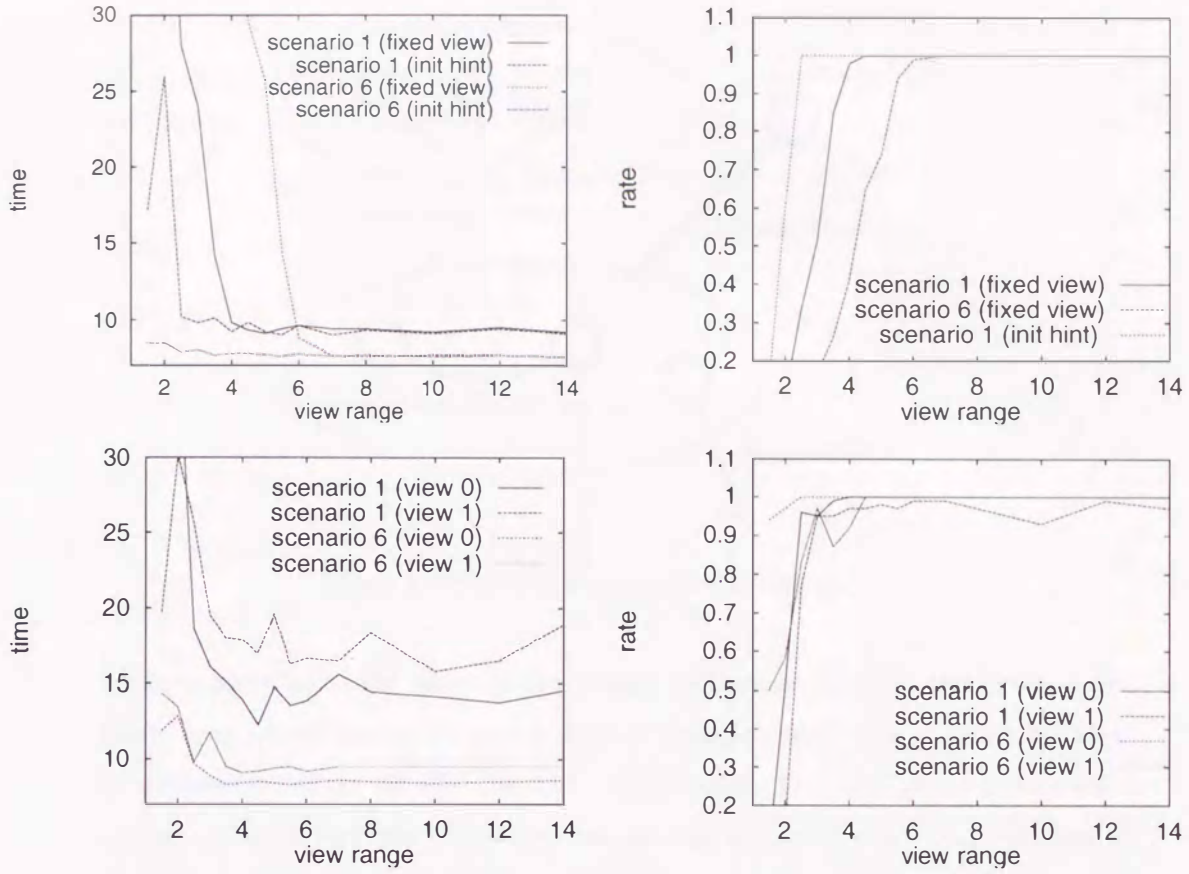


Figure 2.3: Effects of View Ranges in Pursuit Problem

Experiments are with Benda's scenario 1 and 6. The curves labeled "init hint" shows the result of pursuit by agents that know the initial position of the escaper at the first step.

jobs or the time needed to achieve the goal. To get a high completeness, flat structures in which decision-making about actions are distributed are better than hierarchical structures [BJD85]. On the other hand, the hierarchical ordering structure is better than the flat and the negotiation-based structures for the purpose of speed [SM89]. Their experiments were done under environments in which agents have infinite views.

2. Investigate the effects of view ranges of pursuers [Osa93, Nar95]: how much information is required for building a coherent organization for pursuit?

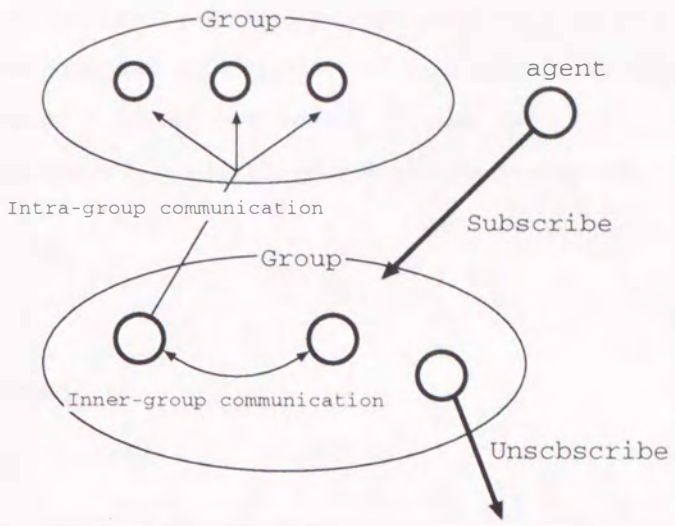


Figure 2.4: Group communication in Cellula

We have investigated the effects of view ranges of pursuers [Nar95]. Our environment differs from Osawa's paper in that it has no communication costs. Thus, this ideal environment shows the net effects of sharing information. The result shows that problem solving can be divided into some stages and that an appropriate view range changes at each stage. As shown in Figure 2.3, the effects of view ranges are very precipitous. It is important to hold a view range larger than $2 \sim 4$. At the same time, this means that agents can avoid broadcasting or infinite view ranges to achieve rational behaviors. This examination also lacks any proposal of dynamic strategies.

Very few researchers have proposed dynamic and flexible structures.

For example, Osawa described a two phase strategy that changed the flat structure into a hierarchical one, when the pursuers reached a pre-fixed distance from the escaper [Osa93]. The value of the threshold was determined by pre-experiences. But it inherently depended on communication cost. In this sense, his approach was rather a static strategy.

2.1.3 languages with field-based communication

Some researchers on distributed processing have proposed languages with group-based communication facilities. Sometimes a group is referred to as a *field* in the context of DAI. A field is used as a mechanism for implicit communication among agents that make a group (Figure 2.4). Proposed languages have at least the following operators on a field:

1. making a new field
2. deleting a field
3. joining to a field
4. leaving a field
5. communicating with the members of a field.

And the objectives of the fields are summarized as follows:

1. to localize communication. Agents can select the receivers that would be interested in the information to be sent. An example is a partitioned blackboard system in [FL77, LE80].
2. to share some properties among members. For example, an agent in a field can modify the behaviors of the member of the field easily [Num92]. This is a kind of group reflection [MWY91].

   It is also viewed as a commitment mechanism. By storing a commitment among agents in a field where the agents belong, it becomes common knowledge[HM84, HM90]. It affects the behaviors of the members as a constraint.
3. to achieve large parallelism by building a field from identical workers [CG89, CG90, Gel85, Chi93]. A message to a field is received by a member that is idle. If the field has a lot of workers, another message can be received by another worker.

For example, our cooperative computation model *Cellula* [YN90, YN91] can make a group dynamically. Since each computational unit in Cellula has its own field, making a new agent means making a new field that is used as a blackboard [LE80, EM88] for agents that have a common interest. Communication through a blackboard uses pattern matching; it can emulate

Table 2.2: Effects of temporary caching method

(1) the number of communication in TSP

number of processors	send immediately	temporary caching
2	1065	3219
3	9534	4301
4	13968	4441

(2) that in number assingment problem (DONALD + GERALD = ROBERT)

number of processors	send immediately	temporary caching
2	112	113
3	473	231
4	625	367

These numbers are quoted from [Nar90]. Though we omit results from the elapsed time, they changed more drastically. Since tuple managements are done in a centralized manner, it becomes a bottle neck easily. As a result, the performance of the system without temporary caching becomes worse as the number of processors increases. On the other hand, a system with temporary caching achieves almost a linear speedup.

both *one*-to-*one* and *one*-to-*n* communication. Though Cellula provides flexible communication mechanisms, making and deleting a field, and managing members of a field is a programmer's task. Therefore building a smart communication strategy should be done by each programmer. Cellula itself has no built-in strategy for general purposes. As long as a specific communication pattern, we have proposed a method to reduce the number of inter-node communications for a Cellula implementation [YN91]. It uses the property of the communication of Cellula. In Cellula, a data unit to be exchanged is called a *tuple*. It has no address (left-hand value); the designation is based on the pattern-matching of their contents. Since the order of pattern-matching is not specified in its semantics, if a tuple matches a query on the local host, no inter-node communication is needed. Therefore, if a node generates tuples and queries that match each other frequently, waiting some periods for local matching could reduce the broadcast. More exactly, in this case, the tuple waiting for broadcast or to be sent back to a tuple-server does not exist in the system, and can only be visible on the generating node. The semantics of the tuple operation on Cellula allows for this temporal inconsistency. We call this method *temporary caching*. Since this mechanism would worsen the system performance, we add to the languuage a declaration that says to use the mechanism or not to use it. In some experiments, it

achieved good result as shown in Table 2.2. But the waiting time is fixed, and not controllable. This static mechanism also has an inherit restriction.

2.1.4 works on cooperative search

Search is not only a fundamental problem of Computer Science but also an important problem of Artificial Intelligence. Studies in AI established some smart sequential algorithms like A* search or beam search and a few parallel algorithms like bidirectional search.

Comparing with other problems, we can parallelize the search algorithms rather easily, because a search process has high-parallelism. Generally, there is no relation between searching agents. But cooperation in searching agents is useful in both distributed and parallel systems. Because the search process can be thought of as a process of finding new knowledge about the problem, exchanging it will be helpful to reduce the size of the job or to achieve a better qualified answer. The importance of cooperation can be explained by following other situations. When we can not use a divide-and-conquer method because decomposed sub-problems have some relations (constraints) with each other, the searching processes should communicate with each other to satisfy global constraints. Sometimes it is called a distributed constraint satisfication. We call both of them *cooperative search* [HW93, Kni93]. In a sense, the distributed monitoring problem mentioned in chapter 2.1.1 can be interpreted as a problem of cooperative search on an erroneous domain, and the pursuit problem mentioned in 2.1.2 is another example that searches for an effective organization and a plan on it.

An important paper was published by Huberman [Hub90, Hub92, HH87]. He showed the effect of exchanging hint information in cooperative search problems. The performance was plotted as log-tail curve against the amount of shared information. He claimed smart cooperation is important for achieving better performance.

Hogg et al. measured the effect of communication between two different search algorithm processes in a hard graph problem (coloring problem) [HW93]. They used an iterated-modifying incomplete search algorithm and an exhaustive and complete search algorithm, simultaneously. They exchange a hint that is a partial solution as we have just described. But communication is done through a central blackboard of limited size. The communication cost is fixed and an

agent makes a decision when it puts a hint into the blackboard without a model of itself or its environment. They reported only the effect of communication density and the effect of coexistence of two different search algorithms, with only 10 agents. We can not extrapolate behaviors in large-scale computational environments from their paper. Kitamura et al. [KTTO93, KC93] also discussed the effect of the diversity of decision making in cooperative searches. They were also concerned with the effect of communication costs. Their algorithm uses the state of a communication channel, but since it does not make a model of agents, the performace is not optimal when the communication cost becomes large.

Since distributed decision making is a kind of distributed search for an agreeable plan, we describe them here. Sycara et al. studied the usage of hints [SRSF91]. Their approach is based on a combination of heuristic optimization and distributed heuristic search. Searching processes exchange *textures*, which are a general form of hints. Their experiment shows that exchanging textures as a model of an environment is useful. Billard and Pasquale studied on the efficiency of shared resource selection on delayed communication networks [BP95]. They analyzed it using a game theoretic approach. Each agent changes its gain matrix. Due to the communication delay and overhead, agents can not select the best combination of behaviors with simple algorithms. They found that a mediate frequency to exchange the gain matrix achieved the maximal net productivity. And they proposed a learning mechanism to reduce the effect of delay. Their approach would be useful, unless the resource was the communication network itself. And since their model used broadcast, any spatial locality of communication was not investigated.

[IGY92] proposed a method of load balancing for a distributed expert system. In this system, the policy of distribution of sub-problems to processors is decided by loads of each processors, and expected communication cost.

Most of them mentioned above did not consider the cost of communication very well. A reason is that their assumed environments were parallel rather than distributed. Since a lot of problems of DAI, however, are classified as search, jobs should be done on physically distributed devices. We must consider more about the cost of communication. We will describe our motivation again in Section 2.3.

2.1.5 other works

Representing organization is necessary to infer it. Some papers have discussed organization. But most works lack the reasoning mechanism for building an appropriate organization dynamically; it would be considered another task by an application programmer on a proposed description framework. The following is a brief survey.

[Num92] proposed a notation of organization that was used for describing a protocol like the contract net protocol. Organizations were built by a programmer.

[MIT90] proposed an approach to build agents based on Object-oriented concurrent programming languages. The system had the notation of a group, which was used for multicast. They assumed that the boundary of a group denoted the boundary of the coherency of information. From our point of view, it was too strong a requirement to finish a job. Weekly shared objects or multi-version objects can not exist in their system.

As an analogy, relations between DAI systems for management sciences or organization theories were captured in [GRHL89, Fox88, MW92]. Both centralized/decentralized computer systems and human organizations have only bounded rationalities. Costs of communication and jobs in human organizations is measured for transaction analysis. But they did not concern themselves with communication cost in large-scaled distributed systems that ware rarely found in human organizations. Fox classified human/computer organizations into some categories [Fox88]. Most primitive structures are a uniform and a hierarchical. But we can think there are intermediate structures between the two structures. Since their differences affect the performance, structures should be characterized by a quantitative measure of communication density.

At Page 58 of [GRHL89], Gasser et al. claim that:

> In our view, an organization is *a particular set of settled and unsettled questions about belief and action through which agents view other agents*. Said another way, we believe that an organization should *not* be conceived as a structural relationship among a collection of agents or as a set of externally-defined limitations to their activities. Instead, ... the way to define organization is to locate the concept of organization in their beliefs, expectations, and commitments of agents themselves.

From our point of view, the unsettled problem that occurred most frequently emerged from the uncertainty about the other agents' knowledge or status, as Gasser also said in the same paper. Thus structures among them can not be static, even if their relationship does not change. And solving the (local) uncertainty is required at any time: organization changes every time. We believe this perspective is more natural than Gasser's interpretation in order to capture its performance.

[HG93] investigates the effect of performance in flat/hierarchical structures. They illustrated the multilevel client-server model.

2.2 Utility of Communication

This section describes the role of communication and the measure of necessity of communication in the context.

In distributed systems in which communication costs can not be ignored, agents or programmers should select a communication structure. But there is uncertainty about the state of other agents. This is caused by both a limited view of agents and the inherent properties of problems. Lesser called it cooperative control uncertainties in [Les91]. Most distributed programs do not suffer from it, because a communication structure is fixed by a programmer to assure the correctness of execution. As a result, communication is requisite and its pattern is fixed. A DAI system, however, has another kind of communication, namely optional communication. The purpose of the optional communication is to share knowledge that is acquired during execution, but is also not required for finishing a job. Thus all optical communications do not require multicast. The reasons are the following:

- Some pieces of knowledge are found in some agents simultaneously. To avoid duplicated communication, the range of communication should be controlled to an appropriate size.
- In some case, changing the range of communication does not affect the correctness of the program. It changes only the performance of the program. Even if the range of communication changes the quality of the answer, it does not mean that broadcast is the best communication method since there would be two agents that are independent

of each other.

- New knowledge would make another one out of date. If new knowledge is found frequently, waiting for new knowledge makes another pieces of information obsolete and reduces the communication cost without performance deterioration.

Therefore, if it can be allowed, reducing the communication size to an appropriate size is desirable on distributed systems. This is a kind of distributed consistency maintenance [FL77, GBH87].

Now it is required to balance the desire to share new knowledge and to keep communication costs small, in other words, to maximize the satisfiability of communication. We can achieve it by introducing the *utility of communication*. Here the utility of communication is defined as the relative benefit of communication against it cost. Figure 2.5 illustrates the relation between communication density and the utility. The dotted line shows the amount of the information that each agent has after communication. By communicating frequently or widely, agents can hold a more coherent and global view of other agents. But we can assume the amount of information is bounded in the domain of distributed problem solving. The solid line shows the cost of communication, which is monotonically increasing. Thus the utility decreases. To achieve the maximum communication effect, agents should be in the shaded region. The best balanced point is the point that maximizes the utility. A reason that Kitamuras' algorithm does not work well with a large cost might be that they did not introduce the idea of utility (see section 2.1.4).

Of course, with this definition, calculating the utility of communication is difficult, because it depends on the global state of the system. Agents on distributed systems can not get the current status of the environment; they can only make an incomplete, partial model of execution. Thus the utility of communication that determines how agents exchange information should be calculated from a model of local computation. The local model is a history of a part of the system that an agent can investigate.

Now we give such a model for cooperative search. In a cooperative search, agents use and exchange some pieces of information about the problem in order to reduce the search space.

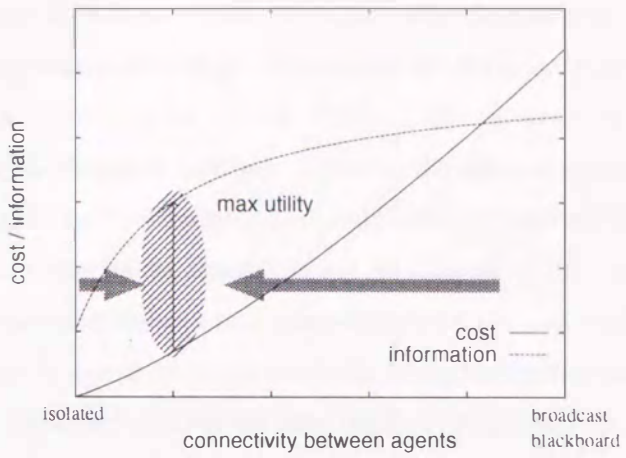


Figure 2.5: balancing utility and cost

Since they are acquired dynamically, an agent can not forecast the current value of information[1] that other agents have without communication. But under the assumption of homogeneity of the agents and the continuity of local computation, we can consider the distribution of the renewal of information of an agent, which is calculated from its history, *is* that of another agent. Thus the current values of information that other agents have can be estimated from its history locally. This simplification makes building a model very easy. In this case, both the communication costs and the amount of reduced tasks due to the acquired information determine the utility. Therefore in order to decide how to exchange information or unsolved sub-problems between agents, an agent can use its partial history as a model of execution.

Precise knowledge with a poor cooperative strategy sometimes leads to selfish behavior. Huberman *et al.* described such a phenomenon on shared object management in [HH88]. For example, let us consider about a group of searchers. If agents share complete knowledge, since they have the same knowledge, we can not expect super-linear speed up that emerges from the diversity of heterogeneous agents. *NK-model* [Kau93] as a self evolutional system might be another example. In the NK-model, agents are automata. At each time, every agent changes its state corresponding to the input (its environment) that is the states of the fixed neighbor

[1] What we call information here is a object Star called 'ideal object' or 'map' in [Sta89].

agents. The change of its state affects the neighbors at the next tick. Thus iteractions among agents make complex feed-back loops. This is a model of the interaction among animals. And the quality of the system is determined by the sum of each agents' adaptability. If agents are very sensitive to the change of their environments, the cascade of changing behaviors can not stop. The system in a positive feedback loop could not be adapted to the situation. On the other hand, the group of very robust agents can not be adapted to the important changes in their environments. Kauffman claimed that most evolutional systems would select an appropriate connectivity between agents by evolution itself. Since the evolutionality of the NK-model is corresponding to the adaptability of agents in distributed environments, we think that the same result would be held in designing the organization of problem-solving agents. Therefore the net utility of communication should not be measured by the amount of information to send, but the amount of contribution to solving the problem of the information. A problem-dependent measure will be required.

2.3 Utility-Based Communication Strategy

In this section we introduce our communication strategy based on the results of the previous discussion. First, we categorize axes of communication that can change the utility. Next we describe our idea and the way to model the agents' environment.

2.3.1 three axes of communication

Now we categorize management methods of communication among agents. Methods for changing communication costs can be put into the three categories shown in Table 2.3. Most approaches in previous works are included in these categories.

Methods in the first category change communication utility by changing the number of

Table 2.3: ways to change communication costs

axis	control parameter
space	number of receiver
time	frequency of communication
information	abstractness and quantity of information to send

receivers of optional communication. Usually the reduciotion of number of traffic leads to the reduction of communication cost. In this case, by reducing the receiver, we can expect the reduction of traffic. Methods in the second category change the frequency of non-mandatory communication. This also reduces the traffic. This approach affects the distribution of decision-making procedures or knowledge. The less agents communicate, the more autonomous they should be.

Ones in the third category change the quantity or quality of information to send by changing the level of its abstractness. Highly abstract information require less communication cost to send than raw data. Complete or too-detailed information may lead to an erroneous path if the information includes errors. On the other hand, agents can not built an exact model of their environments from information that is too abstract. Sometimes these methods are required as the result of a dynamic change of agents' roles. Most studies in MAS belong to this category. Usually agents in MAS can change their roles through distributed planning. In MAS, the computation costs of an organization is of more concern than the communication costs of the organization. Sometimes communication costs are ignored in the context of MAS.

Though the third category has been researched by a number of researchers, and studies have shown the importance of selecting information to exchange, for example [DLC87a, SD92, LC88], few strategies that change the form of cooperation dynamically have been proposed. One reason is the difficulty in the generalization of methods; ways to change the abstractness depend on the problem that the program solves.

Note that these three axes can also be effective ways to resolve conflicts in planning. For example, [DM91b] discussed them in a robot planning context.

2.3.2 controlling communication based on local model

The strategies proposed in this thesis belong to the first two categories. The reason we select these categories is that they lead to communication strategies that are independent of problems to solve.

If we want to select an appropriate connectivity dynamically, as we described in the previous section, a measure of communication utility is required. The result of computation after

each communication should be used to determine the utility. With it, we can calibrate the connectivity between agents. Furthermore, a measure of computation is also required. Our idea is:

1. Assuming that every agent has identical computational power and knowledge to finish the job, we can consider that the knowledge is uniformly distributed. Thus, if the distribution of knowledge is known, we can estimate the amount (effect) of information that is gathered by communication.

2. Now the problem is reduced to how to know the distribution of knowledge. Our idea is to use a history of local computation. If agents are identical, a local history in an agent has common attributes in all the agents' histories. This means cooperative computation is an *ergodic process*, which means the the time-average of a point is the same as the space-average at a time in physics. This is just a hypothesis. We will not give its proof but experiments will show its applicability.

Though this self-control method can be applicable to the third category, we decided to use it in the first two categories. This selection is caused by our motivation. We focus on the high performance execution of programs on distributed environments. In order to achieve good performance, high parallelism is required. Distributed problem solving with autonomous, identical agents is one approach that is available on distributed environments. Since agents can perform problem-solving algorithms by themselves, we can expect linear speed-up from a group of autonomous agents. As the result, the number of agents will be very large. Recent VLSI technologies support this vision. But the speed and bandwidth of communication circuits, especially with non-deterministic communication patterns, will not be sufficient. Reducing communication costs would be the most useful approach. It can be achieved by localizing communication (changing spatial connectivity) or by sending some pieces of information at once (changing temporal connectivity). This is the reason we investigate the first two categories at first.

Of course we can use same categories simultaneously. But here we omit the combination of the strategies to make the explanation simple.

We describe more details of this approach in the next section. We select cooperative search as a framework.

2.3.3 comparison to communication in Multi-Agent System

Now we will describe the difference between DPS and MAS from the communication point of view. It will explain why we choose DPS problems for the application of our strategies.

In DPS, motivation of communication comes out as a result of the execution of operators (actions). Thus each optional communication has a corresponding operator. But the utility of utterance is decided dynamically. It is the reason that the communication strategy is required. On the other hand, in general, the motivation of communication in MAS emerges before execution of a plan or during execution. Since utterance itself is an operator for building a social plan or for solving a sub-problem, it is a more dynamic emergent property. And it is not optional communication but compulsory communication.

Of course, we do not claim that MAS has no optional communication, since DPS is included in the problem-solving process of MAS [Wer92]. After building a social plan and sharing a common goal, agents exchange new information with each other guided by the utility of communication. Especially, if an agent consists of fine grained internal processes, the agent in a MAS will be a group that is consisted of agents that run in a DPS system. However there is no study on a planner or logic that takes account of the utility of communication. In this paper, we focus on the strategy itself. But by embedding it into a logic for cooperation, our proposing strategy can be combined into multi-agent planners naturally.

2.4 Summary

The following is a summary of this chapter.

- In DAI, some communications among agents are optional.
- For getting better performance, we must maximize the utility of communication.
- There are three ways to change the total utility of communications.

- Communication strategies based on changing the number of receivers and changing the frequency, and expecially how to select them dynamically, has not been proposed in previous works.
- We propose an estimation method of communication utility. It uses a history of local computation.

Chapter 3

Communication Strategies for Cooperative Search

In this chapter, we propose some communication strategies for cooperative search. As mentioned in the previous section, cooperative search is a fundamental problem of distributed problem solving. First we build a model of cooperative search. Second, we propose some expectation methods based on the utility of communication. They belong to spatial and temporal categories presented in the previous chapter. Finally we build some concrete communication strategies.

3.1 A Model of Cooperative Search

In this section, we build a model of performance of cooperative search on distributed environments. It will show the importance of optional communication in distributed environments.

First, we must define bases for our discussion. The problem we use here is a cooperative search based on autonomous agents with the branch-and-bound method on distributed environments. The branch-and-bound methods is the well-known effecient method of search. It is also suitable for parallization. With it, each agent searches the best answer that satisfies predefined constraints in a search space simultaneously. Each agent can finish this task by itself; the task need not be decomposed into some distributed subtasks. In other words, communications between agents are optional. Since the search space expands from an initial point during the execution, we can not forecast the total search time or total number of expansion. And, though

this is not a crucial condition, agents execute an abouidentical program. Since they acquire knowledge about an answer during execution, identical program does not mean that they have the same knowledge and processing power.

We assume the information to send is a kind of threshold, which is an one-dimensional measure of goodness of partial answers. Every agent can understand and use it. The goodness of information is totally ordered. Two values t_1 and t_2 can be combined into one, choosing $\min(t_1, t_2)$(or $\max(t_1, T_2)$) and discarding the other. Thus the value descreses (increases) monotonously during its execution.

Since all agents know the goal of system and it is not conflict with each agent's goal; they are identical.

3.1.1 performance of cooperative search

First, we build an execution model of cooperative search with a *fixed communication strategy* or fixed strategy in short.

By above assumptions, each agent expands the nodes or *subproblems* in a state graph and cut some amount of the search space according exchanged threshold. They iterate the job until no unexpanded node remains (see Figure 3.1). Thus the size of unsearched space S_t at step t is described as:

$$S_t = (1 - k(n))(S_{t-1} - Np),$$

where N is the number of agents, p is the searched space by an agent at the step, $k(n)$ is the space cut with the best hint that is exchanged by n agents and the whole search space at the initial state: S_0 is normalized to 1. Because we assume agents are homogeneous concerning average speed, p is identical for all agents.

Exhaustive search terminates at step t^* such that $S_{t^*} = 0$. Thus t^* becomes:

$$\frac{1}{\log(1 - k(n))} \log\left(\frac{(1 - k(n))Np}{1 - (1 - Np)(1 - k(n))}\right).$$

Since the time to execute one step is the sum of the time of a single computation and the time to communicate n members $T(n)$, total execution time: T will be:

$$T = (1 + T(n))t^* = \frac{1 + T(n)}{\log(1 - k(n))} \log\left(\frac{(1 - k(n))Np}{1 - (1 - Np)(1 - k(n))}\right), \tag{3.1}$$

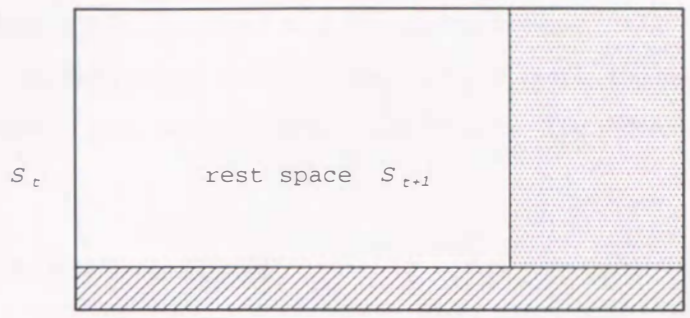


Figure 3.1: reduction of search space

The hatched region shows expanded nodes. The grayed region is cut space. The rest space becomes whole space in the next time step: $t + 1$. We assume the grayed region is larger than hatched region.

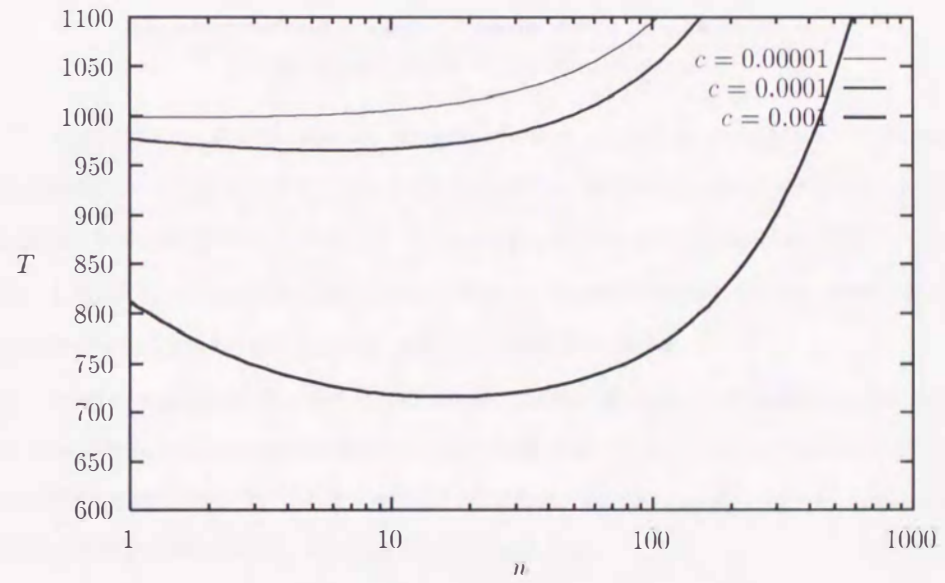


Figure 3.2: effects of cooperation

The number of agents is 1000. p is 0.000001. Cooperation effect: $Tc(n)$ is $cn/(n+1)$ in this graph. The larger cooperation effect becomes, the larger the changes of total time is.

where we use the computation time as the unit of time.

Figure 3.2, 3.3 and 3.4 show expected total time against communication cost function. The larger the order or the factor of the communication cost, $O(T_c(n))$ is, the more sharp the peak is and the narrower the good communication range becomes. The difference of time is only 1%.

3.1.2 estimated search speed

We must now give agents a decision function to select communication costs based on a history of updating the information. It uses a simple idea. We think each step of a distributed problem solving task consists of (local) computation and inter-node communication. They take some periods. If we can estimate their utility, we get the following measurement of computational speed on distributed systems:

$$\frac{\text{goodness-of-computation} + \text{goodness-of-communication}}{\text{computation-time} + \text{communication-time}}. \tag{3.2}$$

The optimal execution of a distributed program is the one which maximizes the above term. Thus if an agent can determine performance properties of the execution environment and the utilities above with their local history, it can manage its communication cost. And by changing the form of communication, the expected utility of communication will be maximized. The selected form would not be optimal but probabilistically optimal.

In cooperative search, exchanged information consists of hints, subproblems, the estimated value of a heuristic evaluation function or anything that reduces the amount of the job and is found during execution. In the following evaluation, we use the strategy to exchange hints themselves. In this case, we can simplify Equation 3.2 as:

$$\frac{\text{amount-of-expansion} + \text{amount-of-cutted-space}}{\text{computation-time} + \text{communication-time}}. \tag{3.3}$$

Finally, by regulating, it is reduced as:

$$\frac{1 + \text{amount-of-branching}}{1 + \text{communication-time}} \sim \frac{1 + O(\text{value-of-threshold})}{1 + \text{communication-time}}. \tag{3.4}$$

The strategy of exchanging subproblems in a distributed A* search will be described later.

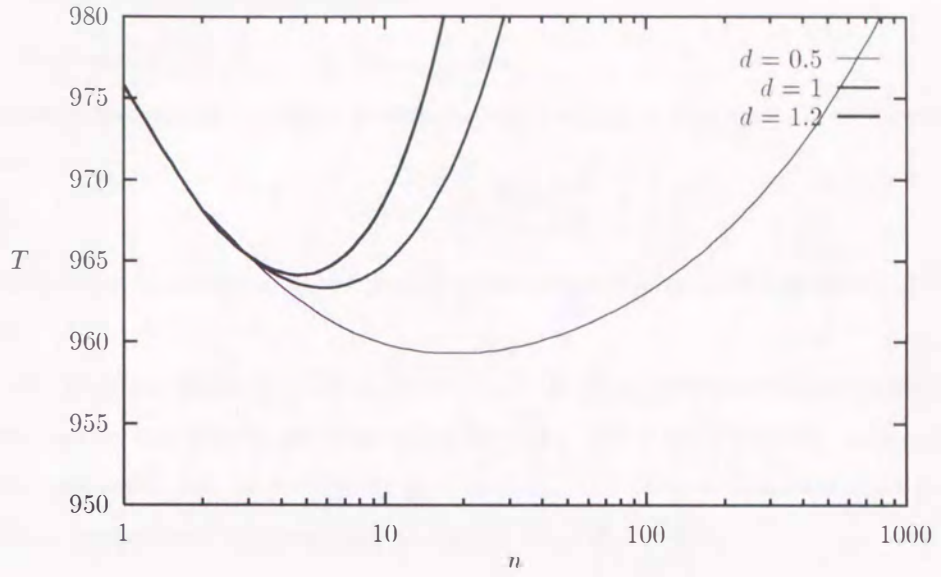


Figure 3.3: effect of communication function(1)

In this picture, communication cost function is n^d, where n is the number of agent (1000). The value of d changes the peak point. Cooperation effect is fixed to $0.0001n/(n+1)$ in this picture. The value of l_1 in Figure 3.4 is ten times as large as one in Figure 3.3.

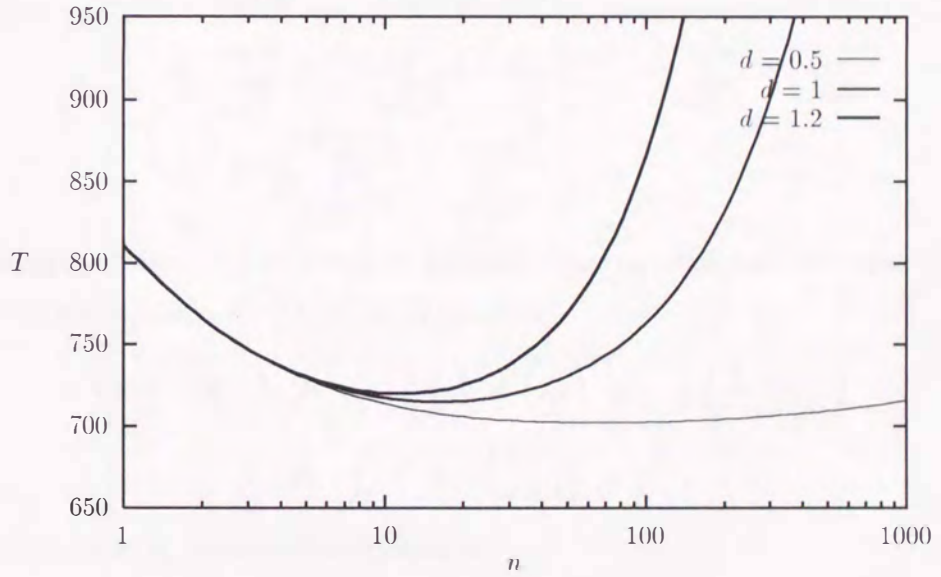


Figure 3.4: effect of communication function(2)

Cooperation effect is $0.001n/(n+1)$. Other parameters are same as Figure 3.3.

3.1.3 quality of the estimation

Now, we give the quality of this estimation method.

Assuming $Np, k(n) \ll 1$, which means the search takes a long time, T of Equation 3.1 becomes:

$$T \approx \frac{1 + T(n)}{Np + k(n)}. \tag{3.5}$$

This is a first-order estimation of the optimal execution that maximizes the utility of communication.

We now analysis the peak point of Equation (3.5). First, we reduce $k(n)$ to $\alpha n/(n+1)$. This term can be acquired by the following calculation. Let $x \in [1/N...N/N]$ be the possible value of information, and its probability $p(x)$ be $1/N$. The effect of communication among n agents can be considered as getting the maximum value after n trials:

$$p_n(x) = x^n - (x - 1/N)^n$$

Thus the expectation of $p_n(x)$, $E[p(n)]$ becomes:

$$\begin{aligned} E[p(n)] &= \sum_{x=1/N}^{1} x p_n(x) = \sum_{x=1/N}^{1} x(x^n - (x-1/N)^n) \\ &= \sum_{x=1/N}^{1} x^{n+1} - \sum_{x=1/N}^{1} (x-1/N)^{n+1} - \frac{1}{N} \sum_{x=1/N}^{1} (x-1/N)^n \\ &= 1 - \frac{1}{N} \sum_{x=1/N}^{(N-1)/N} x^n \end{aligned}$$

Now we assume N is very large ($N \to \infty$). This means the approximation of the above equation can be written with an integral form as the following:

$$\begin{aligned} S \equiv 1 - E[p(x)] &= \frac{1}{N}\left(\left(\frac{1}{N}\right)^n + \left(\frac{2}{N}\right)^n + \cdots + \left(\frac{N-1}{N}\right)^n\right) \\ &\leq \int_0^1 x^n dx = \left[\frac{x^{n+1}}{n+1}\right]_0^1 = \frac{1}{n+1}. \end{aligned}$$

At the same time, we get the following equation:

$$S + \frac{1}{N}\left(\frac{N}{N}\right)^n \geq \int_0^1 x^n dx = \left[\frac{x^{n+1}}{n+1}\right]_0^1 = \frac{1}{n+1}.$$

Combining these two equations, we get:

$$\frac{1}{n+1} - \frac{1}{N} \leq S \leq \frac{1}{n+1}.$$

With $N \rightarrow \infty$, finally we get $S = 1/(n+1)$ and the following result:

$$E[p_n(x)] = 1 - S = 1 - \frac{1}{n+1} = \frac{n}{n+1}.$$

In this case, $E[p_1(x)]$ is $1/2$. Thus we get the following formula:

$$E[p_n(x)] \frac{2n}{n+1} E[p_1(x)]. \tag{3.6}$$

Second, we assume $Np + k(n) \sim k(n) = \alpha n/(n+1)$. Thus we get the following equation:

$$T = \frac{(n+1)}{n}(1 + T_c(n)). \tag{3.7}$$

By differentiating, we get the following equation:

$$T' = \frac{-1}{n^2}(1 + T_c(n)) + \frac{n+1}{n} T_c'(n).$$

Therefore the peak point locates at n such that:

$$\begin{aligned} \frac{-1}{n^2}(1 + T_c(n)) + \frac{n+1}{n} T_c'(n) &= \frac{-1}{n}(1 + T_c(n)) + (n+1) T_c'(n) \\ &= 0 \end{aligned}$$

This means that n must satisfy the following equation:

$$\frac{T_c'(n)}{1 + T_c(n)} = \frac{1}{n(n+1)}.$$

On the other hand, Equation 3.2 can be rewritten as:

$$\begin{aligned} \frac{1 + \alpha k_c(n)}{1 + T_c(n)} &= \frac{1 + \alpha n/(n+1)}{1 + T_c(n)} \\ &\sim \frac{\alpha n/(n+1)}{1 + T_c(n)}. \end{aligned}$$

where we use the previous assumption: $1 < \alpha$. Then peak point becomes:

$$\frac{\alpha}{(n+1)^2}(1 + T_c(n)) - \frac{\alpha n}{n+1} T_c'(n) = 0.$$

By rewriting it, we get:

$$\frac{T_c'(n)}{1 + T_c(n)} = \frac{1}{n(n+1)}.$$

Therefore these equations have the peak point at the same n. We can use the term 3.2 as a good measure of computation utility.

3.2 Range Control Strategy

Based on the result of the previous section, we now give some strategies. The first strategy, *the range control strategy*, manages the cost of communication by changing the number of receivers. It changes spatial connectivity of agents. This strategy assumes changing the number of receivers affects the communication cost. And we use function $n/(n+1)$ as the expected value of information gathered from n agents. This means that the quality of the renewal of information is uniformly distributed. Since this function has an upper boundary, the speed of an agent has a peak if the communication cost function is at least a monotonously increasing function.

Figure 3.5 shows the control flow of this strategy.

Execute the followings after every unit computation:

1. Evaluate the renewal rate of the last value of information x and the current value,
2. store it into the agent's history (vector),
3. Evaluate $E[p_n(x)]$ and the speed (term 3.4) based on the history. Select n that maximize the speed.
4. If $1 < n$ and the information is updated in the this step, send it to n neighbors.

Figure 3.5: control flow of range control strategy

3.2.1 two spatial structures: flat and hierarchical

This strategy is applicable to two topologies: flat structures and hierarchical ones, as shown in Figure 3.6.

In hierarchical structures, agents make some *clusters*. An agent searches in a fixed period, then collects some pieces of information with one-to-one communication from a cluster and sends back the combined information to the member of the cluster. The agents collecting information are shown as shaded nodes in Figure 3.6. Solid arrows in the figure show exchanges

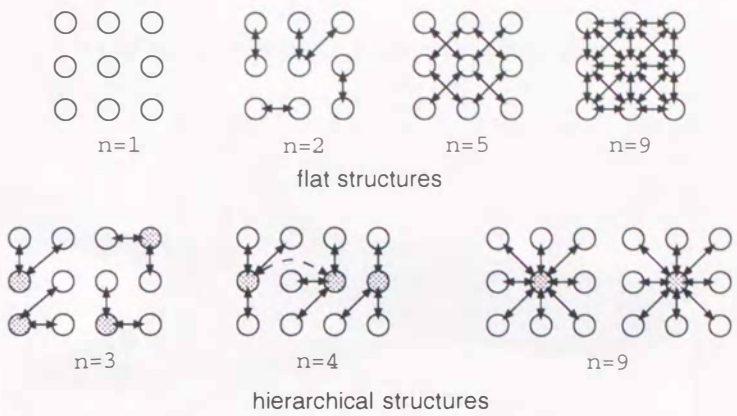


Figure 3.6: two spatial structures

of information in a cluster, and dashed arrows shows possible inter-cluster communications among information collectors. The other members in a cluster communicate only with the collector. If the size of communication range n becomes larger, clusters should grow and the number of them decreases. Using hierarchical communication structures will reduce the number of communication from $O(n^2)$ on flat structures to $O(n)$. In this case, hierarchical structures will be better than flat ones.

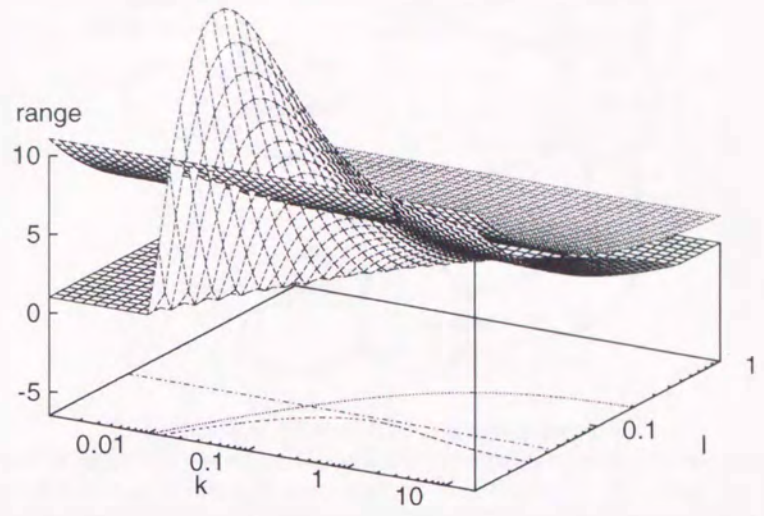
3.2.2 phase transition between two spatial structures

The following mathematical analysis shows the existence of phase transition between flat structures and heterogeneous hierarchical ones. For simplicity we assume agents use one-to-one communication; the cost of multicast between n agents is $O(n)$. The processing speed of both structures communicating n agents becomes:

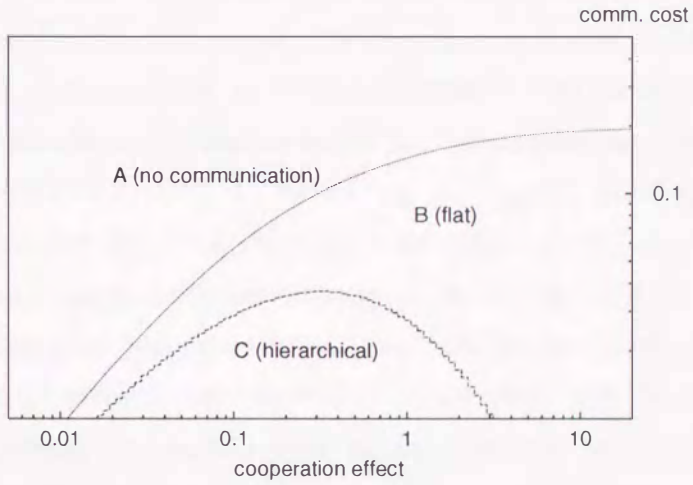
$$\frac{1+k(n)}{1+l(n-1)} \qquad \frac{n}{n+1}\frac{1+k(n)}{1+l},$$

where k or the *cooperation effect* is the factor of the quality of cooperation between n agents $k(n) = kn/(n+1)$, and l is the communication cost at one-to-one communication. The cooperation effect depends on the distribution of the value of new information. The phase transition at which two strategies have the same speed occurs at the following n_1:

$$n_1 = 1 + \sqrt{2 + \frac{1}{l}} > 2. \tag{3.8}$$



(a)



(b)

Figure 3.7: phase transition between strategies

And by differentiating the term $(1 + k(n))/(1 + l(n-1))$, the max speed of flat structures is at the following n_2:

$$n_2 = \frac{-l + \sqrt{l^2 - (k+1)l(-k+l+kl)}}{(k+1)l}. \tag{3.9}$$

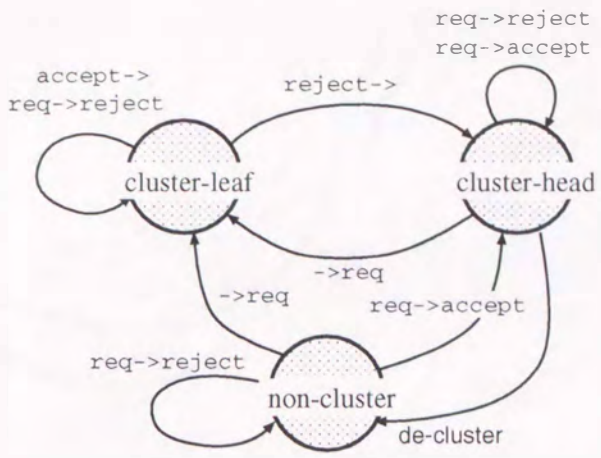


Figure 3.8: protocol for changing structure

Each agent has three states(cluster-leaf, cluster-head, and non-cluster). And there are three kind of messages to change structure(req, reject, accept). For example, if an agent at non-cluster state receives reg message, the agent sends accept message back and moves to cluster-head state.

Figure 3.7 shows the result. In Figure 3.7 (a), Equation (3.8) draws a smooth surface, and Equation (3.9) forms a surface with a peak. The cross lines of surfaces in Figure 3.7 (a) are shown in Figure 3.7 (b) from another view; the intersection of both surfaces forms a dotted line. Figure 3.7 (b) also shows the boundary in which communication has a positive utility as a solid line. These surfaces divide the k-l surface into three regions. In the figure, Region A requires no communication, B ($n_1 > 1$) should use a flat structure and C ($n_2 > n_1$) requires a hierarchical structure. Thus both communication structures are required if the communication cost or the distribution of the information varies. Usually the cooperation effect changes during the execution as the job proceeds. Furthermore the communication cost also changes in some cases, like problem solving with mobile robots. Agents must select an appropriate structure and their roles during the execution.

3.2.3 structure changing protocol

We can extend the range control strategy to a *communication structure selecting strategy*. It selects a proper structure based on expected communication costs at every step. If agents find that a hierarchical structure is better than a flat one, some of them become information

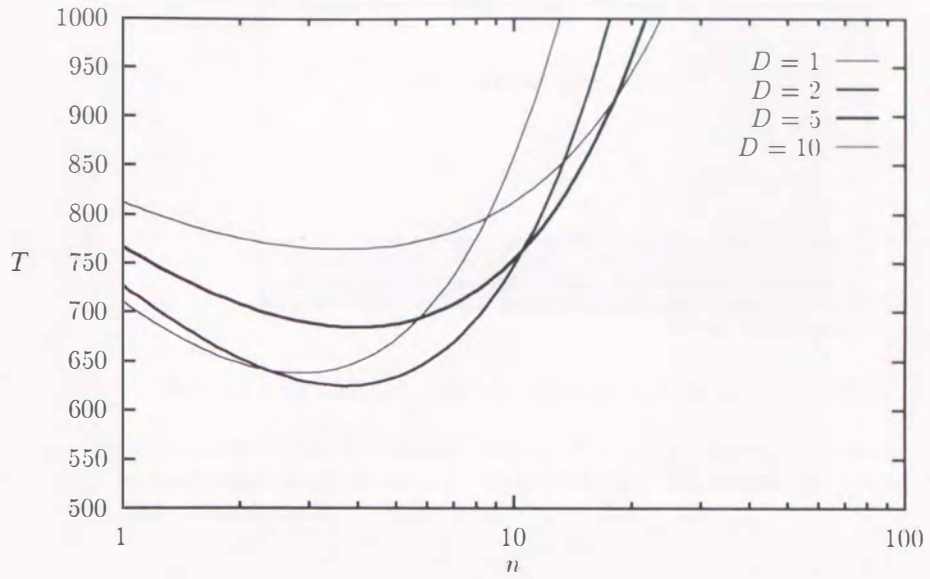


Figure 3.9: effect of broadcast frequency
Broadcast is done after D time local computations.

collectors, and others communicate with one of the collectors.

An information collector manages the size of a group or a cluster after clustering. This strategy creates hierarchical structures autonomously. If the reorganizing cost is low, the structure selecting strategy is better than the range control strategy.

Furthermore, if the information collection becomes a bottle neck once more because of the increase of the size of the cluster, making second level clusters might be required. Though we have not implemented the strategy on multi-cluster structures, a multi level hierarchical system seems rational if the system is confronted to a problem with large uncertainty[Sim81]. Complex social organization should emerge out of necessity.

3.3 Frequency Control Strategy

As well as the range, the frequency of multicast affects the performance of the system. *Frequency control strategy* changes the rate of the update of information (see Figure 3.9). Namely you should communicate in the frequecy f that maximizes the following utility:

Figure 3.10: reduction of search space with frequency control

The hatched region shows expanded nodes. The grayed region is cut space. The rest space becomes whole space in the next time step: $t+1$. We assume the grayed region is larger than hatched region.

$$\frac{f \times \text{amount-of-computation} + \text{effect-of-communication}(f)}{f \times \text{computation-cost} + \text{communication-cost}} \tag{3.10}$$

If the information increases monotonously, we can defer the communication until accumulated updates are worth exchanging. On the other hand, some amounts of the amount-of-computation is useless effect if an agent has the best information at the time. The amount of the useless computation increases monotonically. We can decide the real amount of computation by a history. In this strategy, in order to calculate the utility of communication, we can use a similar model of computation to the one in the range control strategy presented above. While this strategy extrapolates the value of the best information found in n agents, the frequency control strategy extrapolates the value in the whole system or a cluster after f local computation steps. Using a similar model used above, we have evaluated the quality of this strategy. It gives a good estimation of the frequency when the search takes a long time.

While agents in the range control implementations are homogeneous and each of them decides the range of multicast by itself, in the current implementation of the frequency control strategy, a unique agent decides the timing of broadcasting of all agents in the system. It is possible to implement another strategy in which each agent selects its frequency.

Now we make more exact model of the effect of communication frequency in cooperative

search (see Figure 3.10). The purpose of broadcast to exchange latest thresholds with each other is reducing the search space by better threshold value. Therefore we should distinguish the reducing factor by threshold that is found in local unit search from the one that is got from broadcast. The following equation that is used in the range control strategy: $S_{n+1} = (1 - k_c(m))(S_n - mNp)$ should be modified as

$$S_{n+1}^m = (1 - k_b(m))R_X^m((1 - k_c)X - mNp, S_n^m), \tag{3.11}$$

where $k_b(m)$ is the effect of broadcast after m local search period and R represents a recursive relation that is defined as the following:

$$\begin{aligned} R_x^0(f(x), x_0) &\equiv f(x_0) \\ R_x^n(f(x), x_0) &\equiv R^{n-1}(f(f(x)), x_0)). \end{aligned}$$

By solving R_X^m, we get the following:

$$\begin{aligned} S_{n+1}^m &= (1 - k_b(m))\left\{(1 - k_c(1))^m \left(S_n^m + \frac{mNp}{k_c(1)}\right) - \frac{mNp}{k_c(1)}\right\} \\ &= (1 - k_b(m))(1 - k_c(1))^m S_n^m + (1 - k_b(m))\left((1 - k_c(1))^m - 1\right)\frac{mNp}{k_c(1)}. \end{aligned} \tag{3.12}$$

By solving about S_n, we get the following equation:

$$S_n^m = X^n\left(1 + \frac{B}{A-1}\right) - \frac{B}{A-1},$$

where the auxiliary variables are:

$$\begin{aligned} X &= (1 - k_b(m))(1 - k_c(1))^m, \\ Y &= (1 - k_b(m))\left((1 - k_c(1))^m - 1\right)\frac{mNp}{k_c(1)}. \end{aligned}$$

And the number of iteration C such that $S_c^m = 0$ is:

$$C = \frac{1}{\log X}\log\left(\frac{Y}{X + Y - 1}\right).$$

Thus the consumed time T_m is:

$$T_m = (m + T_c(N))C = (m + T_c(N))\frac{1}{\log X}\log\frac{Y}{X + Y - 1}, \tag{3.13}$$

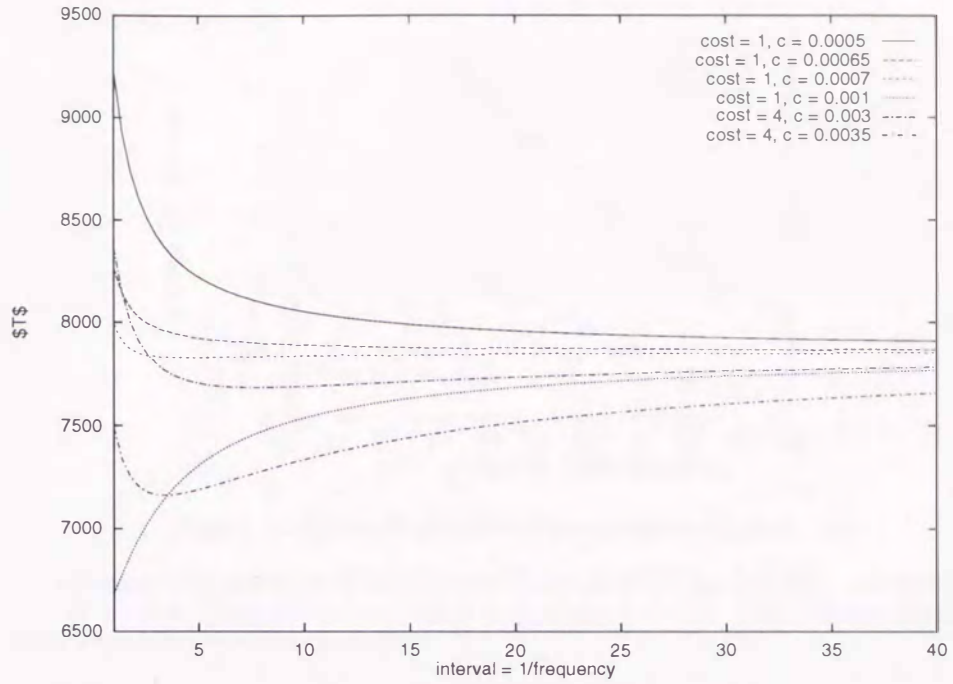


Figure 3.11: effect of multicast frequency

where $T_c(N)$ is time to broadcast among N agents.

We can now estimate the total consumed time against the frequency of communication. First, We expect $k_b(m) > k_c(1)$, since the distribution after m time local computations is larger than one after m time computation-and-communication steps. For example, we can use $k_c(m) = mk_c(1)$ as $k_b(m)$. And we assume that $k_c(1) > p$ again. With these estimations of N and $T_c(m)$, agents can estimate the time against broadcasting frequency.

Figure 3.11 shows the graph of Equation 3.13 with some parameters. In any cases, expected time is converges to a fixed time, when they are isolated, namely 1/frequency $\rightarrow \infty$.

The best frequency of multicast at which the consumed time is minimized changes from one to infinity. The change of consumed time is also very large. Therefore fixing the frequency is not rational. Dynamic balancing is required.

Since the parameters like $k_b(m)$ and $k_c(m)$ change during execution, it is difficult to forecast the exact time to finish search by using this model. This estimation, however, is useful for building estimation methods.

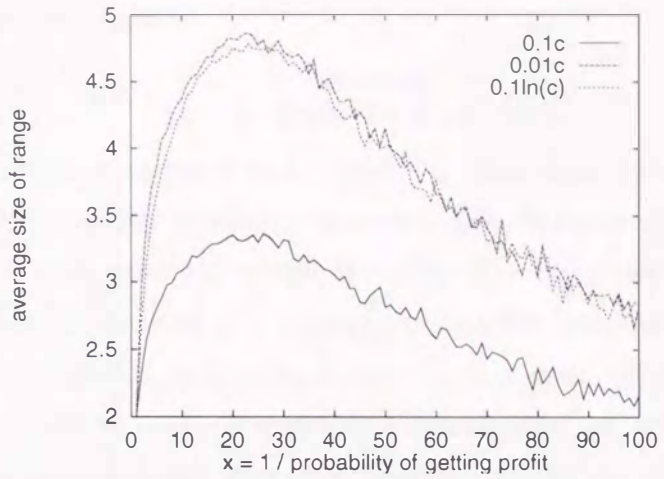


Figure 3.12: the effect of the heterogeneity of agents

Each agent is assigned 100 if random(x) = 0, otherwise 0. Communication cost is $0.1c$, $0.01c$ or $0.1\ln(c)$, where c is the number of packets sent in a step. The sizes are average values after 20 steps of 100 simulations.

3.4 History-Based Estimation of Parameters

In this section, we give an estimation method of some parameters for building a model of an environment. As described in the previous section, we use an update history of a local variable. This approach has the following merits.

1. The process of maximizing the utility of communication consists of two steps: collecting information and estimating the utility with the collected information. Making a history is a collecting method that requires only cheap computation cost.

2. No additional communication is required. This locality in making a model is desired in distributed systems. Furthermore, if mandatory communication is required for performing the job, agents can exchange their histories as additional information. In this way, agents can escape from complete isolation and can deal with heterogeneous environments.

Now we show the adaptability of this strategy using a simple game. In this game, each agent gets either a fixed positive number (100) as a profit or zero as no result at every step

according to the following function:

$$\text{value} = \begin{cases} 100 & \text{if } \text{random}(x) = 0 \\ 0 & \text{if } \text{random}(x) \in [1...x-1], \end{cases}$$

where random(x) returns a random integer in $[0, x-1]$. Each agent stores the result in its history memory and exchanges it with others in order to share the best profit. If the possibility of getting a profit is large, namely x is small, the utility of communication becomes low. At the same time, if agents merely get a positive profit, the utility also becomes low. In these two cases, the heterogeneity of distribution of information is small, which means the range of communication should be small. If x and the communication cost are fixed, the size of the communication range converges after some iteration. And we can change the utility of communication by changing n. Figure 3.12 shows the result. Here, communication cost is $0.1c$, $0.01c$, or $0.1\ln(c)$, where c is the number of packets sent in a step. The sizes are average values after 20 steps of 100 simulations. We can find a peak size at $x = 20 \sim 30$. The size at the peak is affected by communication costs. This result shows that the strategy selects a proper connectivity of agents.

We can check its quality of the size of the range by the following analysis. The peak point should maximize the following term:

$$\frac{1-(1-1/N)^n}{T_p + T_c} \tag{3.14}$$

By differentiating by n, we will get the curve in the graph.

$$\frac{\partial}{\partial n} \max_n \frac{1-(1-1/N)^n}{T_p + T_c} = 0 \tag{3.15}$$

Now we assume the communication cost is linear. This assumption is already used in the previous simulations. Then we get the following equation:

$$T_p + T_c = \alpha + n. \tag{3.16}$$

$$\begin{aligned} n &= \frac{N}{2}\{A \pm \sqrt{A + \frac{4\alpha}{N}}, \\ &\text{where } A \equiv (\frac{\alpha}{N\log(1-1/N)})^2. \end{aligned} \tag{3.17}$$

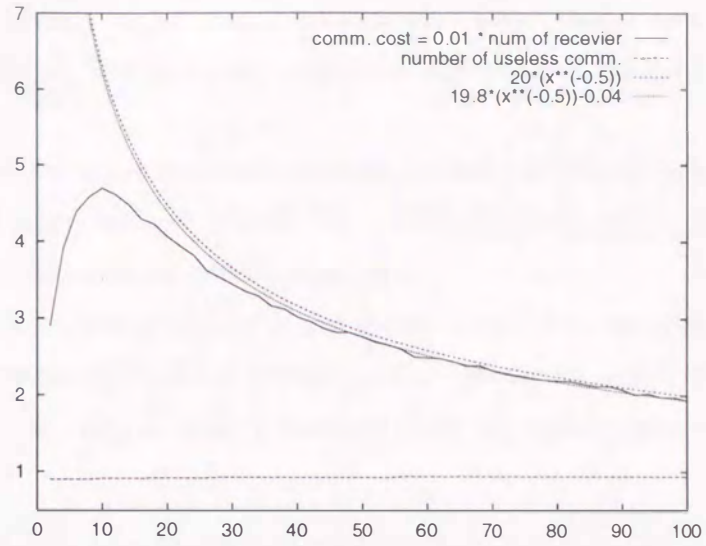


Figure 3.13: estimated curve

By using the relation: $\log(1+x) \sim x$ if x becomes large,

$$A \sim (1-\alpha)^2 \tag{3.18}$$

thus, the optimal n becomes:

$$n = \frac{N}{2}\{(1-\alpha)^2 \pm \sqrt{(1-\alpha)^2 + \frac{4\alpha}{N}} \tag{3.19}$$

Thus the graph should have the following curve:

$$\begin{aligned} \frac{\partial n}{\partial p} &= O(\sqrt{\frac{1}{n}}) + O(n)O(n^{-\frac{3}{2}}) \\ &= O(n^{-\frac{1}{2}}) + O(n^{-\frac{1}{2}}) \\ &= O(n^{-\frac{1}{2}}). \end{aligned} \tag{3.20}$$

Figure 3.13 shows an average curve with a communication cost function $0.001x$. The size at probability 50 and 100 is 2.76 and 1.94 respectively. If we assume an estimating function should be $Ax^{-0.5} + B$, A and B become 19.8 and -0.04. It matches for experimental data well. Remember the length of each agent's history is 10.

This figure also shows the average number of useless communication per node. Here 'useless communication' means receiving a number not bigger than its own profit. Through whole

probabilities, it is slightly smaller than 1. The number bigger than 1 means oversending. Likewise a too small value lets agents not cooperative. Agents expect the distribution of profit well.

We can conclude this history-based estimation method can build a model of the environment (the probability of information occurrence) even if the length of the history memory of each agent is short for representing the probability exactly.

Here we should note the applicability of this history-based self modeling method. In principle, each agent requires the model of the surrounding environment in DAI/MAS. But since they shared some properties, an agent's behaviors reflect the other agents' ones. They are summarized as:

1. The Goal is shared among all agents. It is given by a programmer or as a result of analysis of the specification of a given problem.

2. An information is renewed during execution. Even if agents are identical at the start time, its uniformity breaks down after some local computations.

3. Every agent knows that the goal is shared by all agents. This is a main motivation to exchange the information acquired dynamically.

4. Agents believe that the others have the ame rationality or intelligence as themselves. This assumption reduces the complexity of the world. And without it, agent can not estimate the utility of his behavior because of its uncertainty.

Communication strategy will be adapted to problems that have these requirements. These requirements remember the required rationality in non zero-sum games[RGG86, RB89]. Actually, we can think that history-based estimation makes a game matrix where a possible actions are one of communicating nobody, communicating one, communicating two, and so on (see Figure 3.14). Homogeneity helps estimating the gains of other agents.

3.5 Structure of an Agent

In this section, we describe the structure of agents that in based on our estimation model.

gain matrix

agent1 \ agent2	state1, action1	...	...	...
state1, action1	gain111/gain112	...		
state1, action2	gain121/gain122			
...	...			
state1, actionN				
state2, action1				
...				
...				
state3, action1				
...				
...				

homogenity estimation

agent1 \ agent2	action1	...	...	...
action1	gain11/gain11	...		
action2	gain12/gain12			
...	...			
actionN				

folding by symetry

communication	profit
smallest	...
...	...
...	...
largest	...

Figure 3.14: making game-matrix from local history

Usually a gain matrix shows the profits of N agents when they do (action1, action2, ... action N), where the matrix is N-dimensional. In this case, since we can not expect the current states of other agents, profits should be assigned to tuples of their state and their action. Our estimation method based on local computation history makes it as a symetric matrix, since we assume all agents have the same value-distribution. As a result we can ignore the state of each agent. Furthermore, we assume every agent acts in the same manner, N dimensional matrix is reduced to a vector against their connectivity. A local goal does not conflict with the global or other's goal.

In this framework, an agent consists of the following three components.

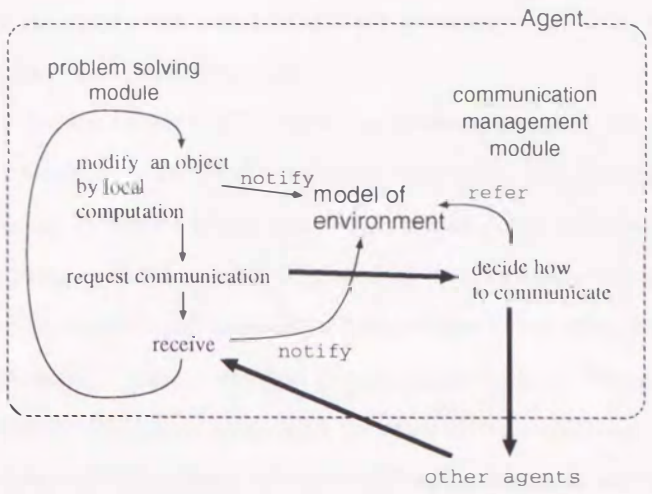


Figure 3.15: structure of searching agent

problem solving module

Description of algorithm for solving a problem. This is given by problem-solving programmer.

model of environments

A data as a model of agent's environment. It depends on the strategy for cooperation in the agent.

communication management module

Code for changing communication utility. It consists of communication strategy and a mechanism to change the way of communication.

To separate each other and from our point of view, problem solving module merely include codes for cooperation among agents. This is invoked from the problem solving module.

Figure 3.15 shows the relation between modules. Communication management modules are independent of algorithms of problem solving modules. Two modules in an agent interact through the model. At each step of the execution of the problem solving module and receiving information from others, current information changes the model. With the model and the

communication cost function, the communication management module maximize the term (3.2) under the assumption of their homogeneity.

We want to keep the modularity in its implementation as possible. By separating modules and making general interface between the modules, communication management module and model could be reusable. It will be a help for both DAI researchers who make new cooperative strategies and usual programmer who want to make fast programs on a distributed environment.

From the view of software engineering, above separation is not sufficient. Communication management module included both parts that required modifications for each strategy and not. The former part should be treat as a component for reuse. If we recall it as a framework, agents now consist of a framework (the general structure of communication management module), a strategy that can be selected by a programmer freely, and a problem solving module. We will describe such an implementation in Section 6.

3.6 Summary

Here is the summary of this chapter.

1. A model of cooperative search shows the necessity of a communication strategy. A programmer can not decide the best communication topology before the execution of a program.

2. We proposed two strategies. One is changing the number of receiver. The other changes frequency of communication. By changing communication cost, they maximize the utility of communication.

3. Mathematical analysis shows the necessity of a phase transition between flat structures and hierarchical structures.

4. Both strategies achieve good estimation if the search space is huge.

5. We use a history based model of execution. It can hold renewal rate of information. This method requires only small computation cost and small memory under the assumption of homogeneity of agents.

6. From our view point,an agent consists of three components. By making a clear interface between them, we can built an agent with better modularity.