Strategies for Selecting Communication Structures in Cooperative Search

Shuji Narazaki, Hiroomi Yamamura, Norihiko Yoshida
Kyushu University
{narazaki,yamamura,yoshida}@csce.kyushu-u.ac.jp

Abstract

Modeling environment is essential for agents to cooperate with each other in a distributed system. In this paper, we propose two strategies for selecting agents' communication structures in cooperative search using their local histories as a model of their computational environment. Using a history of local computation, an agent can select a proper communication structure, and the utility of communication always matches its cost. Simulations using the traveling salesman problem shows the strategies get good performance. We also describe an extension of these strategies to other areas and a way to separate them from application programs using meta-object programming in Object-Oriented Programming Languages (OOPL).

1 Introduction

Communication cost is one of important properties of distributed computational environments. If we can ignore the communication cost, sharing information among all processes or *agents* is the most rational for cooperating with each other. But since communication cost is high in a distributed system, we give up achieving any global view. This means each agent has only a local view and weak consistency of shared objects. For effective execution, locality of agents' view should be determined by the communication cost in the execution environment and stages of problem solving. But it is difficult for an agent to decide the optimal communication structure because of the limitation of view.

For example, consider an execution of cooperative search, which is a typical problem of distributed problem solving. Each agent gets new information or constraint some times during the execution, and communicates with others in order to exchange it for better performance. But when the communication cost is not negligible compared with computation cost, exchanging information would make the performance worse. Thus deciding to whom and when an agent sends new information is an important issue in massively distributed computational environments with limited bandwidth networks.

Though making proper structures of agents for problems with uncertainty in distributed environments has been researched in the area of *Distributed Artificial Intelligence* (DAI) [1,2,14] and this is an important issue of DAI [4,8,10,15], few realistic strategies have been proposed.

In this paper, we propose communication strategies to select an appropriate structure dynamically. With them, each agent manages its communication cost using an expected value of collected information derived from its local history concerning a renewal rate of the information. Consequently programmers need not select an efficient communication pattern at programming time. We have applied the strategies to cooperative search. But they can be used in many applications.

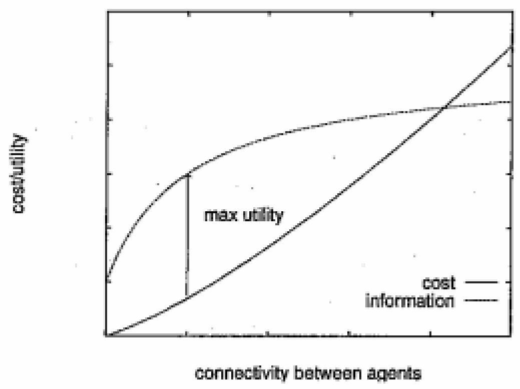


Figure 1: balancing utility and cost

The structure of this paper is as follows. Section 2 describes the strategies. We evaluate them by simulation in section 3, and discuss the result in section 4. The last section 5 is the conclusion.

2 Communication strategies using partial models of execution

Uncertainty about both execution environments and inherent properties of problems requires cooperation of agents. To cooperate with each other, an agent requires a model of the others or its *environment*. Since agents in distributed environments can not get the current status of environments, each agent can only make an incomplete, partial model of execution. Thus, to know the current status, agents should exchange their local models and extrapolate the current status by merging them. If agents can communicate without any communication cost, broadcasting to everybody anytime is the best communication strategy. But it could never happen especially in a massively distributed environment. On the other hand, the complete model is not required necessarily. Thus the *utility* of communication should determine how agents exchange information (see Figure 1). Here utility is the term describing goodness acquired from an action and is derived from economics or game theory. In this case, both the communication cost and the amount of reduced tasks due to the acquired information determine the utility.

In cooperative search, agents use and exchange some peaces of the information on the problem in order to reduce the search space. Since they are acquired dynamically, an agent can not forecast the current values of other agents without communication. But under the assumption of the homogeneity of agents, we can think the history of an agent *is* the history of another agent. The current values in other agents can be estimated from the history of itself. In this case, exchanging models is not required. Therefore in order to decide how to exchanging information or unsolved subproblems between agents, an agent can use the local history as a model of execution.

Table 1: ways to change communication cost

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

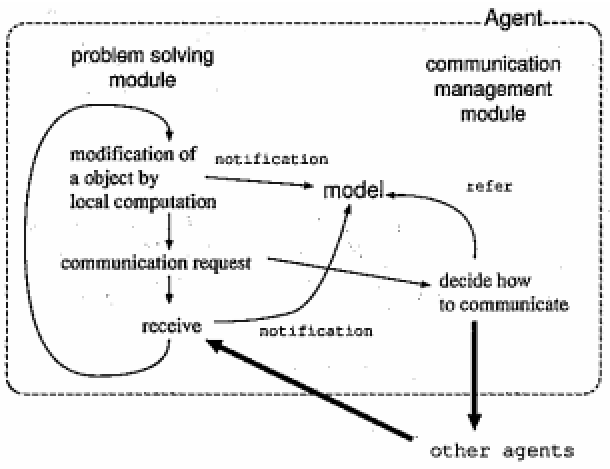

Agent
problem solving module
communication management module
modification of a object by local computation
notification
model
refer
communication request
decide how to communicate
receive
notification
other agents

Figure 2: structure of searching agent

Methods for changing communication costs can be put into the three categories shown in Table (1). Though third category has been researched by a number of researchers, for example [3, 18, 23], but almost no researches were proposals of a strategy but evaluations of the importance of selecting communication structure. The strategies proposed here belong to the first two categories. The reason we select the categories is that they would lead to communication strategies that are independent to the problem to solve. Here we omit about mixed strategies for making the explanation simple.

Now we must give agents a decision function to select communication cost based on a history of updating the information. We use a simple idea. An optimal execution of distributed program would maximize the following factor:

$$\frac{\text{utility-of-computation} + \text{utility-of-communication}}{\text{computation-time} + \text{communication-time}}. \tag{1}$$

Thus if an agent can determine performance properties of the execution environment and the utilities above with its local history, it can manage its communication cost. And the system can get the optimal computation structure.

In this framework, an agent consists of a *problem solving module*, a *communication management module* and a *model* of execution. Figure 2 shows the structure. 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 problem solving module and receiving information from others, current information changes the model. With the model and the communication cost function, communication management module maximize the factor (1) under the assumption of the homogeneity of them.

In cooperative search, information exchanged is hint or subproblems and so on. In the following evaluation, we use the strategy to exchanging hint itself. The strategy to exchange subproblem will be discussed in section 4.

2.1 controlling spatial connectivity of agents

The first strategy, *range control strategy* manages the cost by changing the number of receivers. It changes spatial connectivity of agents. This strategy assumes changing the number of receivers affects

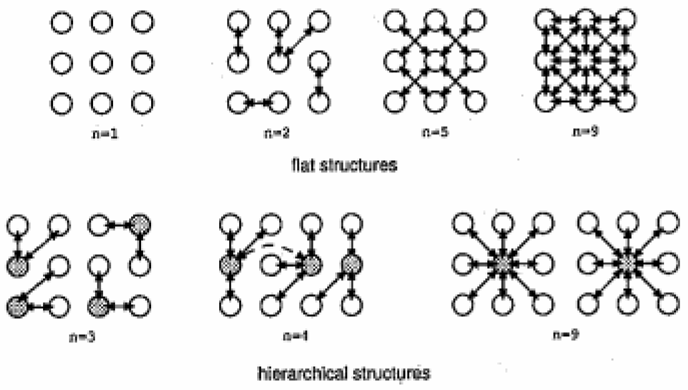


Figure 3: two spatial structures

the communication cost. And we use function $n/(n+1)$ as the utility of communication among n agents. It gives right values if the renewal is uniformly distributed.

In a simple experiment, the strategy shows a self-organization of agents. In this simulation, a random number is assigned to each agent. Each agent stores the given number into its history memory and exchange it with each other in order to know the largest one. If agents are homogeneous, the utility of communication becomes low. This means the range of communication becomes small. Fixing the assignment function and communication cost, the size of communication range would converge after some iteration. By changing the assignment function, we can change the utility of communication. Figure 4 shows the result. We can find a peak at some heterogeneity of assignment. This means the strategy selects proper connectivity of agents.

Here we analyze the goodness of the strategy in cooperative search. At first we build a simple model of cooperative search. In cooperative search, each agent expand the nodes (subproblems) in state graph and communicate hint if needed. They iterate the job until no unexpanded node remains. Thus the size of unsearched space S_t at step t is described as:

$$S_t = (1 - k(n))(S_{t-1} - Np) \tag{2}$$

where N is the number of agents, p is a searched space by an agent at a step, $k(n)$ is the space cut with the best hint that is exchanged by n agents and the whole search space S_0 is 1. And we assume all agents work in a synchronous way. Since the time to execute a step is the sum of the time of single computation and the time to communicate n members $T(n)$, total execution time T will be:

$$T = \frac{1 + T(n)}{\log(1 - k(n))} \log\left(\frac{(1 - k(n))Np}{1 - (1 - Np)(1 - k(n))}\right), \tag{3}$$

where we use the computation time as the unit of time. Assuming $Np, k(n) \ll 1$, which means the search takes a long time, T becomes:

$$T \approx \frac{1 + T(n)}{Np + k(n)}. \tag{4}$$

Thus strategies maximizing factor (1) give the first-order estimation of the optimal execution.

This strategy is applicable to two topologies; flat structure and hierarchical one, as shown in Figure 3. In hierarchical structures, agents make some *clusters*. An agent collects some peaces of information with one-to-one communication from a cluster and send back the combined information to the member

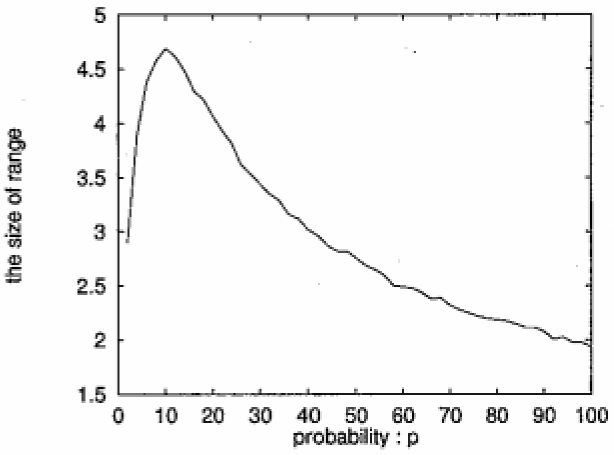


Figure 4: the effect of the heterogeneity of agents

Each agent is assigned 100 if random$(p) == 0$, otherwise 0. Communication cost is $0.01+10^{-5}x+4\times10^{-8}x^2$, where x is the number of packets sent in a step. The sizes are average values after 20 steps of 100 simulations.

of the cluster. The members communicate only with the collector. If the size of communication range n become larger, using hierarchical communication structures will reduce the cost to $O(n)$ from $O(n^2)$. Hierarchical structures will be better than flat ones.

Furthermore, *role differentiation* might occur in the group of identical agents. If communication range becomes very large, collecting tasks becomes a bottle neck. Then the agent collecting information should stop the computation and devote to manage communication among the cluster. Heterogeneous agents will emerge from homogeneous agents.

Mathematical analysis shows the existence of a phase-transition between flat structures and heterogeneous hierarchical ones. The processing speed of both structures communicating n agents becomes:

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

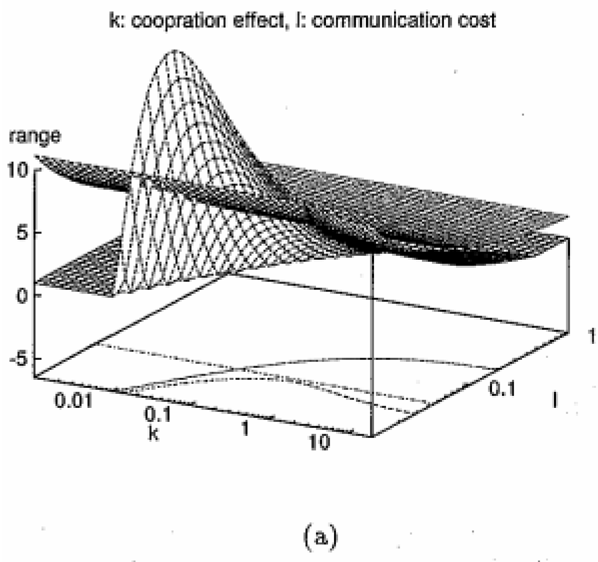
where k is the factor of the goodness of cooperation between n agent $k(n)=kn/(n+1)$, and l is communication cost to one to one communication. The phase transition occurs at the following n:

$$n = 1+\sqrt{2+\frac{1}{l}} > 2. \tag{5}$$

And the max speed of flat structures is at the following n:

$$n = \frac{-l+\sqrt{l^2-(k+1)l(-k+l+kl)}}{(k+1)l}. \tag{6}$$

Figure 5 shows the result. Equation (5) draws a smooth plane, and Equation (6) forms a plane with a peak. These planes divide k-l plane into three regions. In the figure, Region A requires no communication, B should use a flat structure and C require a hierarchical structure. Both communication structures are required if communication cost or the distribution of the information vary. Agents must select an appropriate structure and their roles during the execution.



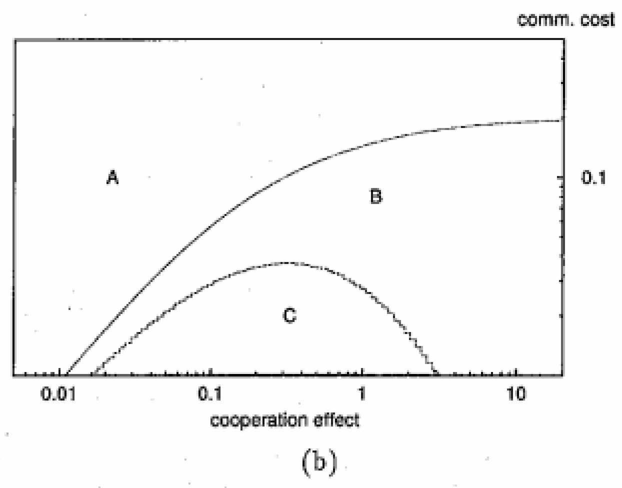


Figure 5: phase transition between strategies

Therefore we can extend the range control strategy to a *communication structure selecting strategy*. It selects a proper structure based on expected communication cost at every step. If agents find that a hierarchical structure is better than a flat one, some of them become information collectors and others communicate with one of collectors. An information collector manages the size of a group or cluster after clustering. This strategy emerges hierarchical structures autonomously. If the reorganizing cost is low, 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 confronts a problem with large uncertainty[22]. Social organization should emerge by need.

2.2 controlling frequency of communication

As well as the range, the frequency of multicast affects the performance of the system. *Frequency control strategy* changes the rate of update of information. If the information increases monotonically, we can defer the communication until accumulated updates is worthwhile exchanging. In this strategy, to calculate the utility of communication, we use a similar model of computation to the one in range controlling strategies above. While range control strategy extrapolates the value of the best threshold found in n agents, frequency control strategy extrapolates the value in the whole system or a cluster after m local computation steps. Using a similar model used above, we have evaluated the goodness 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 will be possible to implement another strategy in which each agent selects its frequency.

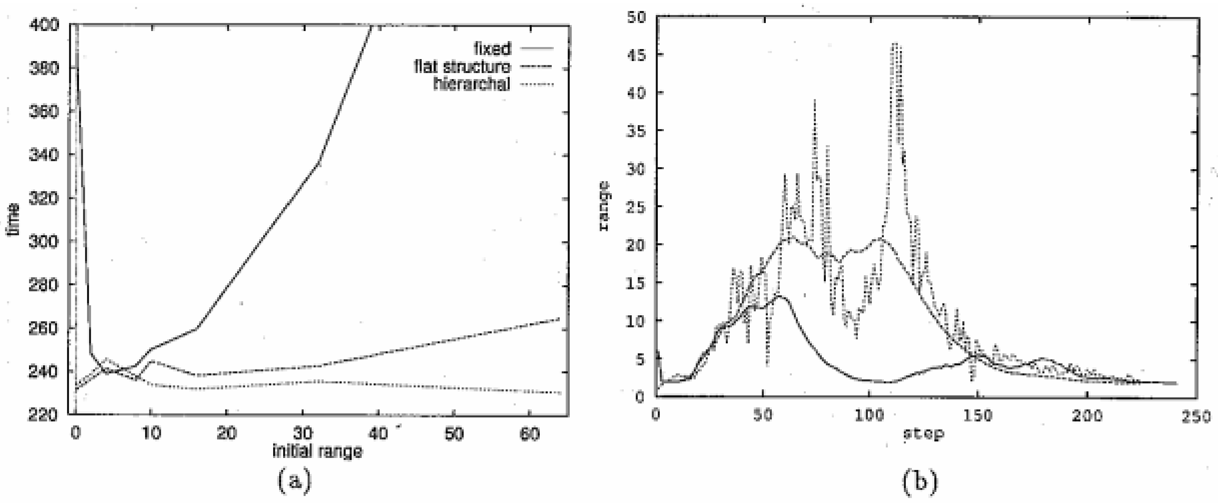


Figure 6: the result of spatial strategies

The communication cost is $0.01 + 10^{-5}x + 10^{-9}x^2$,where x is the number of all communication (packet) in the step. In (b), the solid line shows range control strategy on flat structure, the dashed line shows the optimal size given by control structure on hierarchical structure, and the dotted line is the real size of clusters on hierarchical structure. Communication cost used in (b) is different from the one in (a). The cost function hardly affect the tendency of range change. All of points are average values of 10 $\sim$ 100 data.

3 Evaluation of the strategies

The goodness of the strategies has been measured with simulations of a traveling salesman problem (TSP). We implemented range control strategies on both spatial structures, frequency control strategy and fixed strategy for comparison. TSP is the problem of searching the shortest path of given cities. A number of agents can search the shortest path in parallel. We used a *branch-and-bound* method in the problem solving module that exchanges the cost of current best path as a threshold. Since the goodness of threshold increases monotonously, merging some pieces of information (threshold) means just selecting the best one among them. It relieves the expectation cost. The pseudo-coded algorithm is following:

```
do in parallel {
  **while** ( global bag is empty ) {
    pick up a candidate from global bag
    expand it
    **if** ( a new node is better than threshold ) update local threshold
    **if** ( local threshold is updated ) multicast it
  }
}
```

In this simulation, the number of cities is 10 and the length of histories is 10. The system consists of 100 agents. In this case, the threshold updates over 200 times. After each expanding a node, the difference between the value of current threshold and the one before the expansion is stored in the

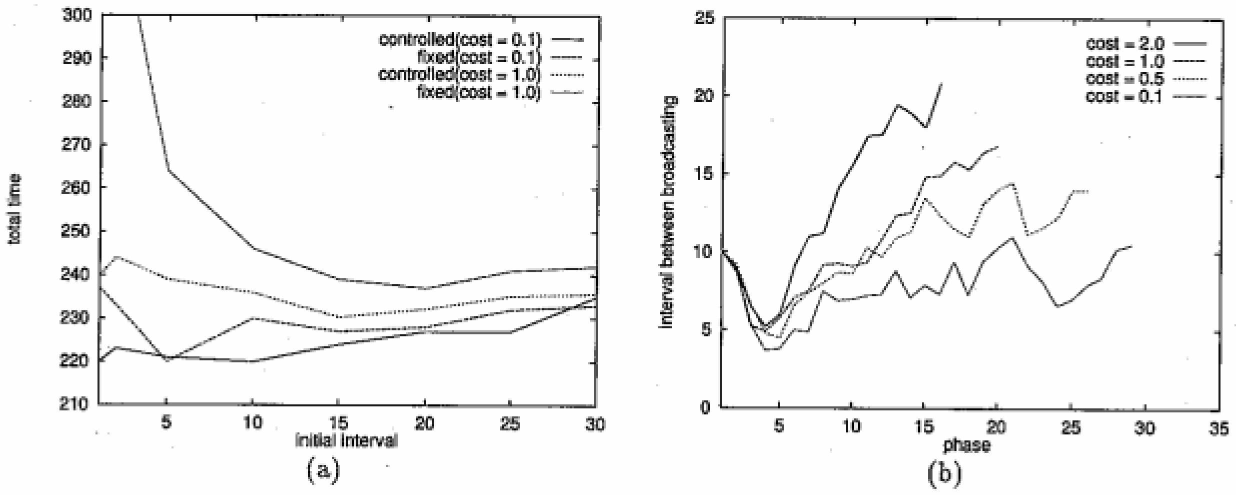


Figure 7: the result of frequency strategy

The cost of communication is the ratio compared with the computation cost. 'Phase' is the local computation steps between two broadcasts.

history memory of an agent. For the history can simulate the probability of updating threshold, a Monte Carlo simulation gives the agent the expected best value among n agents. At every step agents select a communication structure. Since the simulator models asynchronous networks, range control strategy on hierarchical structures requires some steps of communication in order to change cluster size.

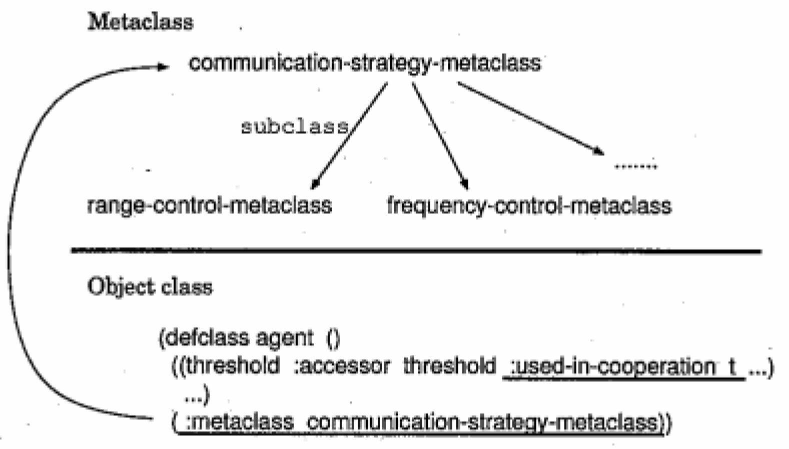
The result shows that the strategies proposed get good performance against affected communication costs. A result is shown in Figure 6, 7. The properties we can find are summarized as:

1. Changing the range and the frequency during the execution is useful. As Figure 6 (b) shows, there are three stages from the view of the size of ranges. In first and last stage, the communication range is small. This result is derived from the nature of searching process: at the initial stage, no agent find a new threshold; after finding and broadcasting the best answer in the middle stage, updating the threshold never occur and no more communication needs. Conclusively, The more frequently information are renewed, the more frequently or more widely identical agents communicate. But too frequent updates decrease the number of communication. The initial range hardly affect the performance.

2. In frequency control strategy, the frequency of communication is affected by the communication (broadcast) cost. Similarly, the size of communication range decreased if its cost is high where the utility of communication becomes low.

3. There is no clear difference between flat structures and hierarchical structures. The reason is the peak size of ranges is not so large in this simulation and changing cluster size requires some other communications. This reduces the merit of hierarchical structures.

The properties above are held in other communication cost functions from $O(\log(n))$ to $O(\exp(n))$. They hardly depend on cost function if it increases monotonously. Conclusively programmers need not care about the communication topology by using the strategies.



Metaclass
communication-strategy-metaclass
subclass
....
range-control-metaclass
frequency-control-metaclass
Object class
(defclass agent ()
((threshold :accessor threshold :used-in-cooperation t ...)
...)
(:metaclass communication-strategy-metaclass))

Figure 8: description with metaclass

4 Discussion

In this section, we discuss the applicability of these strategies to other fields and a way of implementations to use strategies easily. Related work is also described.

4.1 applicability of the strategies

The current shortest path as a threshold in TSP is a form of hint acquired during execution. The search algorithm used here does not uses any estimation heuristics. In general case, we can use an estimated value as the measure for modeling execution status. Considering A* search, the goodness of a node (subproblem) is measured by an evaluation function. Thus we can use the strategies to exchanging subproblems with the following relations.

application	our TSP implementation	A* search
measure	update rate of threshold	update rate of best estimated value
exchanged data	threshold	good unexpanded subproblem

Furthermore, the application of the strategies is not only cooperative search. The strategies can be thought as algorithms for managing the consistency of hint that is shared *weakly* between agents. As well as it, the idea using a local history as a model of environment could manage shared objects with strong consistency. Usually strongly consistent objects are used much more than weak consistent objects. Thus programmers' burden to keep coherency effectively will be relieved in a number of application area.

In this case, the measure of execution is ratio of local access and remote access. Exchanging value among processors is a shared object itself. If the current cost for remote access is higher than the expected cost for managing coherency, shared object should be duplicated and distributed. On the other hand, if it is low, the objects should be merged to reduce the cost in writing. If agents (copies of a shared object) know the access ratio sufficiently with the access history, the strategies work well.

4.2 separation of communication strategy form problem solving

As mentioned above, communication strategies which collect the information as a history are useful. We think deciding communication structure should be a kind of computational reflection [19]. This implies writing codes for communication strategies can be omitted from programmers' task, having an interface to a meta object. It is important to separate communication strategies as a meta-problem solving from the algorithm for solving a problem. It would make reuse of communication strategies easy.

From this point of view, we are re-implementing the strategies as meta objects in CLOS (Common Lisp Object System)[24] and use MOP(Meta-Object Protocol)[16, 17, 21]. Communication strategies is implemented as a meta object of application classes. Figure 8 shows the relation of classes. As shown in the figure, metaclass `communication-strategy-metaclass` is an abstract class for communication strategies. The strategies we mentioned here are subclasses of this class. One of them is attached to an application class corresponding an agent or a process with keyword parameter `:metaclass` in its definition. Programmers also add a keyword parameter `:used-in-cooperation` to the slot that is the measure of execution. The metalevel class adds methods for managing its history and invokes the method for communication strategy automatically when the agent wants to send information. A communication cost function is given in creating its instances.

4.3 related work

Huberman described that cooperation between searching agent make performance better as a result of theoretical analysis[12, 13]. Hogg also researched cooperation in graph problem [11]. But they did not mention how to communicate each other and did not consider communication cost. Their work is a kind of cooperation in parallel activities. Need for cooperation is not emerged from the constraint of execution environments but the property of problems; the communication structure for cooperation depends on the problem itself.

Relation between OOPL or meta object and Multi Agent System (MAS) have been researched [5, 6, 9]. We think that the separation of communication policy using metaobject is a step toward multiagent oriented programming language from OOPL.

5 Conclusion

In this paper, we have mentioned communication strategies based on self histories for modeling the environment to select efficient communication structures dynamically. The result of experiments shows the strategies give good communication structures to autonomous agents. We expect that quantitative models of execution are useful for MAS.

Though there are some restrictions that the implemented strategies assume homogeneity of agents, the simplicity of combining some pieces of information and the knowledge of the communication cost function, we think that these strategies can be used in many systems that acquire new knowledge in execution time. They would contain distributed database systems that replicate data. The history-based expectation mechanism could apply to not only homogeneous environments but also heterogeneous ones.

In current implementations, the history of the environment is not separated from its own history. But in heterogeneous network this would become an important problem. The implementation and the evaluation is a future plan. It will also work on heterogeneous agents. Research on strategies combined range control and frequency control is another one.

References

[1] Nicholas M. Avoris and Les Gasser, editors. *Distributed Ariticial Intelligence: Theory and Praxis.* Kluwer Academic Publishers, 1992.

[2] Alan H. Bond and Les Gasser, editors. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, 1988.

[3] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent Cooperation Among Communicating Problem Solvers. *IEEE Transactions on Computers C-36*, pages 1275–1291, 1987. Reprinted in [2], pp.268–284.

[4] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Cooperation Through Communication in a Distributed Problem Solving Network. In Huhns [14], chapter 2, pages 29–58.

[5] Jacques Ferber and Jean-Pierre Briot. Design of a Concurrent Language for Distributed Artificial Intelligence. In *Proceedings of the International Conference of Fifth Generation Computer Systems*, pages 755–762, 1988.

[6] Jacques Ferber and P. Carle. Actors and Agents as Reflective Concurrent Object:a Mering-IV Perspective. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1420–1436, November/Deceember 1991.

[7] Stephanie Forrest, editor. *Emergent Computation.* The MIT Press, 1991.

[8] Les Gasser. Boundaries, identity, and aggregation: Plurality issues in multiagent systems. In Eric Werner and Yves Demazeau, editors, *Decentralized A.I. 3*, pages 199–213. Elsevier Science Publishers B.V., 1992.

[9] Les Gasser. Object-Based Concurrent Programming and Distributed Artificial Intelligence. In Avoris and Gasser [1], pages 81–107.

[10] Les Gasser, Nicholas F. Rouquette, Randall W. Hill, and John Lieb. Representing and using organizational knowledge in distributed AI systems. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence, Vol.2*, chapter 3, pages 55–78. Pitman/Morgan Kaufmann, London, 1989.

[11] Tad Hogg and Colin P. Williams. Solving the Really Hard Problmes with Cooperative Search. In *AAAI-93*, pages 231–236. AAAI, AAAI Press/The MIT Press, 1993.

[12] Bernardo A. Huberman. The performance of cooperative processes. *Physica D*, 42:38–47, 1990. Reprinted in [7].

[13] Bernardo A. Huberman. The value of cooperation. In Michael Masuch and Massimo Warglien, editors, *Artificial Intelligence in Organization and Management Theory*, chapter 10, pages 235–243. Elsevier Science Publishers B.V., 1992.

[14] Michael N. Huhns, editor. *Distributed Artificial Intelligence.* Morgan Kaufmann, 1987.

[15] Toru Ishida, Les Gasser, and Makot Yokoo. Organization Self-Design of Distributed Production Systems. *IEEE Transactions on Data and Knowledge Engineering*, 4(2):123–134, 1992.

[16] Gregor Kiczales, J. Michael Ashley, and Luis H. Rodriguez Jr. Metaobject Protocols: Why We Want Them and What Else They Can Do. In Paepcke [20], chapter 4, pages 101–131.

[17] Gregor Kiczales, Jim des Rivières, and Daniel G. Bobrow. *The Art of the Metaobject Protocol.* The MIT Press, 1991.

[18] Victor Lesser and D. Corkill. Functionally accurate, cooperative distributed systems. In Bond and Gasser [2], chapter 4.3.1, pages 295–310.

[19] Pattie Maes and Daniele Nardi, editors. *Meta-Level Architectures and Reflection*. North-Holland, 1988.

[20] Andreas Paepcke, editor. *Object-Oriented Programming The CLOS Perspective*. The MIT Press, 1993.

[21] Andreas Paepcke. User-Level Language Crafting: Introducing the CLOS Metaobject Protocol. In *Object-Oriented Programming The CLOS Perspective* [20], chapter 3, pages 65–99.

[22] Herbert A. Simon. *The Sciences of the Artificial*. The MIT Press, Boston, second edition, 1981.

[23] Young-Pa So and Edmund H. Durfee. A Distributed Problem-solving Infrastructure for Computer Network Management. *International Journal of Intelligent & Cooperative Information systems*, 1(2):363–392, June 1992.

[24] Guy L. Steel Jr. et al. *Common Lisp the Language*. DEC press, second edition, 1990.