

STRATEGIES FOR SELECTING COMMUNICATION STRUCTURES IN COOPERATIVE SEARCH

SHUJI NARAZAKI, HIROOMI YAMAMURA, and NORIHIKO YOSHIDA
Department of Computer Science, Kyushu University
Fukuoka, 812-81, Japan

Received

Revised

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 a cooperative search using their local histories as a model of their computational environment. Under the assumption of homogeneity of agents, an agent can select a proper communication structure by using a history of local computation, and the utility of communication always matches its cost. Simulations using the traveling salesman problem show that strategies produce high performance. We also describe an extension of these strategies to other areas and the means to separate them from application programs using meta-object programming in Object-Oriented Programming Languages (OOPL).

Keywords: Distributed problem solving, cooperative search, self-organization, computational reflection

1. Introduction

Cooperation is a kind of meta-level computation which controls problem-level computation of autonomous processing units or agents. It controls the number of agents, rolls of each agent, communication topologies and frequencies, messages among agents, and so on.

Forms of cooperation affect the performance of computation but not the semantics of computation. 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. Classes of these forms even include sequential or centralized computation forms. Agents are advised to choose centralized computation if they find it is better than decentralized (usually referred to as *cooperative*) computation.

In distributed problem solving, greater knowledge of the circumstance or conditions leads to better behavior of agents, but sharing them causes more communication costs. Therefore it is crucial to consider the *utility* of communication to exchange local knowledge of each agent. The term "utility" stems from economics and the game theory, and it is used to describe the *quality* of a result of an action.

If communication costs were negligible, sharing information among agents would be the best for cooperation. But communication costs are much higher than computation costs in real distributed systems. Agents can no longer have a global view. Instead, each agent has its local view only, and there is just partial consistency of shared information. How local the view of an agent is depends on the utility and the cost of communication among the agents, but because its view is local, the agent can not find the optimal form of communication.

Cooperative search, such as distributed vehicle monitoring, is a typical example of distributed problem solving. Agents running in parallel perform some search procedure. They can reduce the magnitude of their own task or improve the quality of search results by exchanging information about models of themselves, the circumstances, 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 entire performance is. Therefore deciding when and with whom to communicate is crucial.

Finding an adequate structure of agent groups is an important issue of Distributed Artificial Intelligence (DAI) [4, 8, 10, 16], and there has been some research completed: [1, 2, 15], but few communication strategies have been proposed.

In this paper, we propose communication strategies for cooperative search. Under the assumption of homogeneity of agents, which means all agents have the same computational power, our strategies estimate the real amount of information based on the history of local revision of information. Agents control communication cost by changing the number of agents that exchange information with each other or by changing the frequency of communication. Consequently, programmers need not select an efficient communication pattern at programming time.

The structure of this paper is as follows. Section 2 describes the strategies. We evaluate them through traveling salesman problem simulations in Section 3. Finally we discuss the results in Section 4. Section 5 is the conclusion.

2. Communication strategies using partial models of execution

2.1. a model of communication

On distributed systems in which communication cost can not be ignored, agents 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. This makes selecting a proper structure difficult. Thus to avoid useless communication, it is required that agents estimate the current global state of other agents or their *environments*. Agents in distributed systems can not get the current status of the environment; they can only make an incomplete, partial model of execution. However, making the complete execution model by exchanging local models is not required necessarily, since the exchanging cost becomes very high.¹ Thus the *utility* of communication that determines how agents exchange

¹The completeness of the model might not be required. Precise knowledge with a poor cooperative strategy sometimes leads to selfish behavior. Huberman *et al.* described such a phenomenon on shared object management in [14].

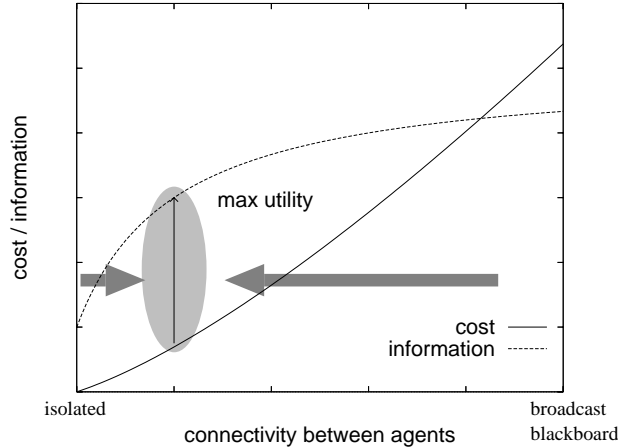


Figure 1: balancing utility and cost

information should be calculated from a local model.

Figure 1 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, 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. It is monotonically increasing. Thus the utility as the difference decreases. To achieve maximum communication effect, agents should be in the shaded region.

In a cooperative search, agents use and exchange some pieces of information about the problem in order to reduce the search space. Since they are acquired dynamically, an agent can not forecast the current value of information that other agents have without communication. But under the assumption of homogeneity of agents and the continuity of efficiency of 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 subproblems between agents, an agent can use its partial history as a model of execution.

Methods for changing communication costs can be put into the three categories shown in Table 1. Though the third category has been researched by a number of researchers, and studies has shown the importance of selecting information to exchange, for example [3, 19, 22], few strategies that change the form of cooperation dynamically have been proposed. The strategies proposed here belong to the first two categories. The reason we select these categories is that they would lead to

Table 1: ways to change communication costs

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

communication strategies that are independent of problems to solve. Here we omit the combination of the strategies for making the explanation simple.

Now we must 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:

$$\frac{\text{utility-of-computation} + \text{utility-of-communication}}{\text{computation-time} + \text{communication-time}}. \quad (1)$$

The optimal execution of a distributed program would maximize 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 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 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 (1) under the assumption of their homogeneity.

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. The strategy of exchanging subproblems in a distributed A* search will be described in section 4.

2.2. controlling spatial connectivity of agents

The first strategy, *the 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 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. This term can be acquired by the following calculation. Let $x \in [1/N \dots N/N]$ 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

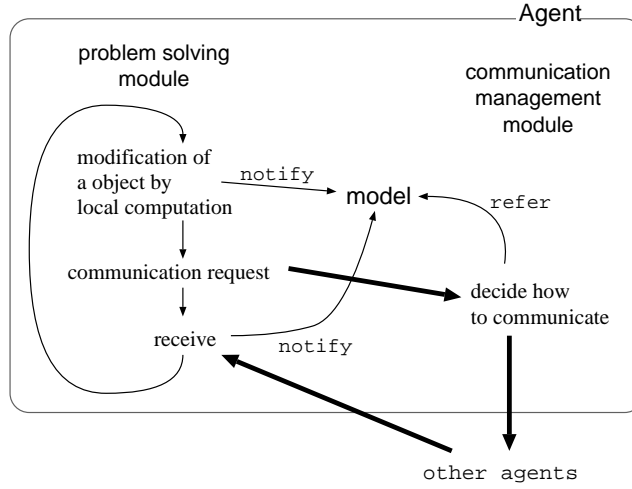


Figure 2: structure of searching agent

after n trials: $p_n(x) \equiv (xN)^n - \{(x - 1/N)N\}^n$. Thus the expectation of $p_n(x)$ becomes:

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

By $N \rightarrow \infty$, we get the result:

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

Since this function has an upper boundary, the speed of an agent has a peak if the communication cost is at least a monotone increasing function.

Now we show the adaptability of this strategy in 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 $\text{random}(x)$ returns a random number 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

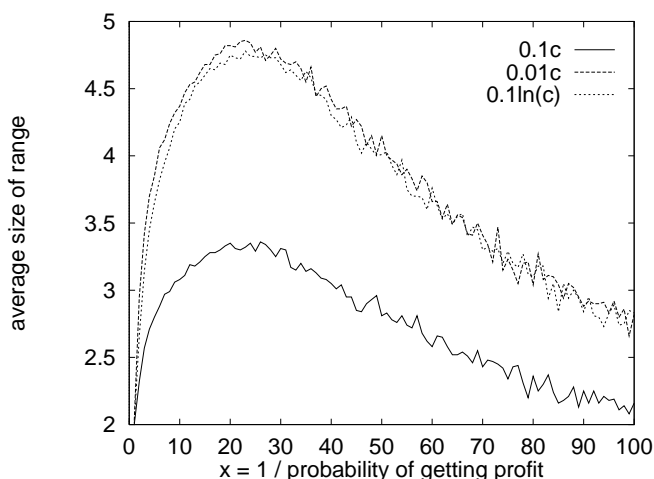


Figure 3: the effect of the heterogeneity of agents

Each agent is assigned 100 if $\text{random}(x) = 0$, 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.

of communication becomes low. At the same time, if agents merely get 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 communication cost are fixed, the size of communication range would converge after some iteration. And we can change the utility of communication by changing n . Figure 3 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.

Now we analyze the quality of the strategy in a cooperative search. First we build a simple model of a cooperative search. In a cooperative search, each agent expands the nodes or *subproblems* in a state graph and exchanges hints if needed. We assume that agents use a branch-and-bound method. They iterate the job until no unexpanded node remains. Thus the size of unsearched space S_t at step t is described as:

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

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 as the initial state is 1. Because we assume agents are homogeneous concerning average speed, p is identical for all agents. Exhaustive search terminates

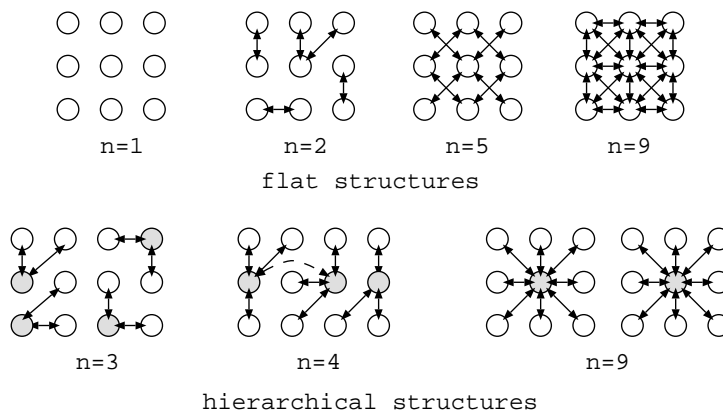


Figure 4: two spatial structures

at step t^* such that $S_{t^*} = 0$. Here 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:

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

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:

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

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

This strategy is applicable to two topologies; flat structures and hierarchical ones, as shown in Figure 4.

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

Furthermore, *role differentiation* might occur in the group of identical agents. If the communication range becomes very large, collecting tasks becomes a bottle neck.

Then the agent collecting information should stop the computation and devote itself to managing communication in the cluster. Therefore we can develop a method emerging heterogeneous group from homogeneous agents. In this method, every agent checks the size of its communication range. If it reaches a threshold size, an agent becomes an information collector or a member of a cluster that already exists. We can decide the threshold size using the result of the following paragraph. Since this change is a probabilistic process, clusters grow gradually.

Mathematical analysis shows the existence of a 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:

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

where k or *cooperation effect* is the factor of the quality of cooperation between n agent $k(n) = kn/(n + 1)$, and l is the communication cost at one-to-one communication. 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 :

$$n_1 = 1 + \sqrt{2 + \frac{1}{l}} > 2. \quad (2)$$

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

$$n_2 = \frac{-l + \sqrt{l^2 - (k + 1)l(-k + l + kl)}}{(k + 1)l}. \quad (3)$$

Figure 5 shows the result. In Figure 5 (a), Equation (2) draws a smooth plane, and Equation (3) forms a plane with a peak. The cross lines of planes in Figure 5 (a) are shown in Figure 5 (b) from another view: the intersection of both planes forms a dotted line. Figure 5 (b) also shows the boundary in which communication has positive utility as a solid line. These planes divide k - l plane 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. Both communication structures are required if communication cost or the distribution of the information varies. Agents must select an appropriate structure and their roles during the execution.

Therefore 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 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, structure selecting strategy is better than the range control strategy.

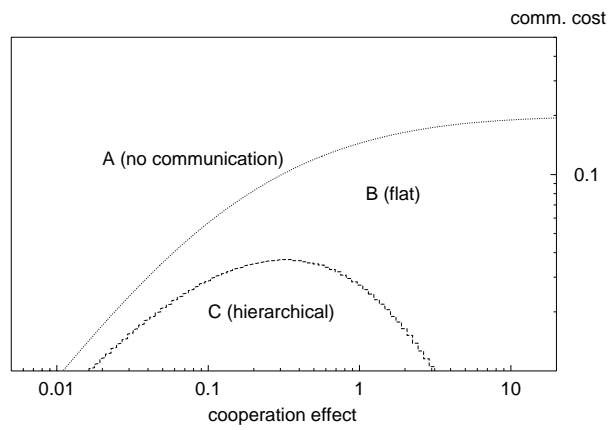
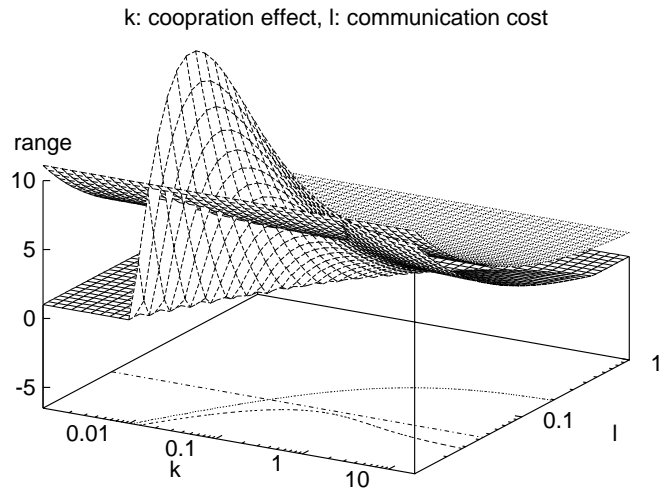


Figure 5: phase transition between strategies

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[21]. Social organization should emerge out of necessity.

2.3. 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 the update of information. If the information increases monotonically, we can defer the communication until accumulated updates are worth exchanging. In this strategy, to calculate the utility of communication, we can use a similar model of computation to the one in range control strategies above. While range control strategy extrapolates the value of the best information 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 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 will be possible to implement another strategy in which each agent selects its frequency.

3. Evaluation of the strategies

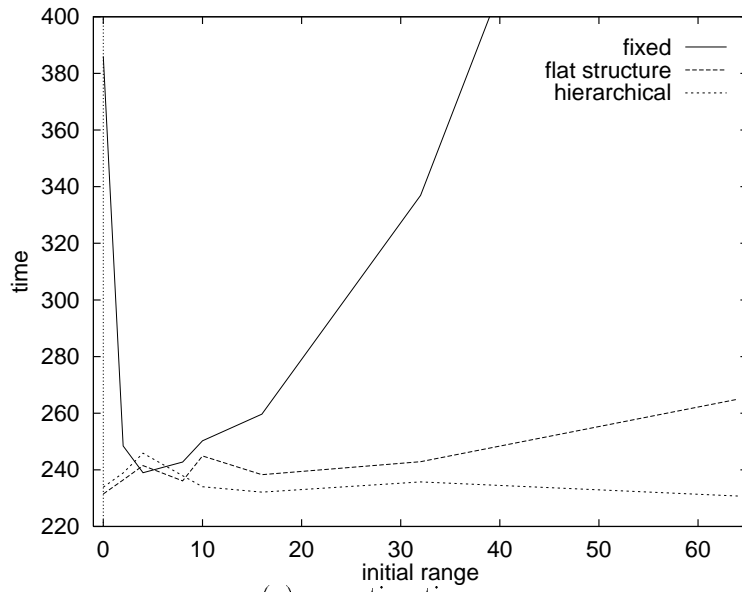
The quality of the strategies has been measured with simulations of the Traveling Salesman Problem (TSP). TSP was defined by A. J. Hoffman and P. Wolfe in p. 2 of [18] as:

The TSP for a graph with specified edge lengths is the problem of finding a Hamiltonian cycle of shortest length.

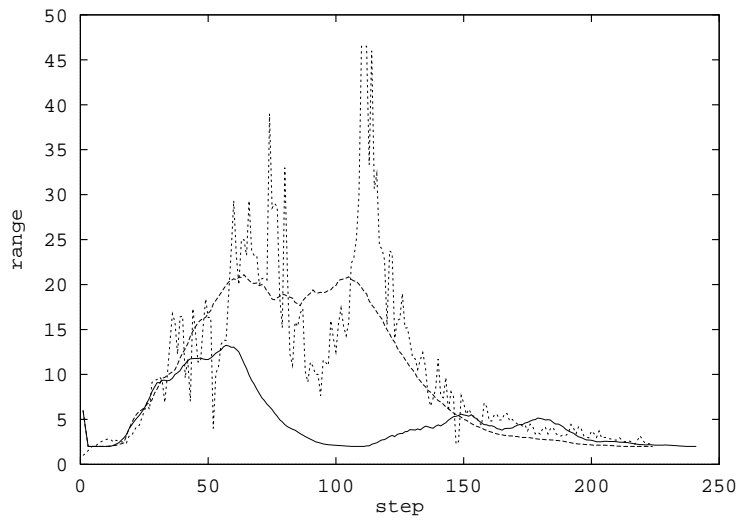
A Hamiltonian cycle is a cycle that contains all the vertices of the graph exactly once. We implemented range control strategies on both flat spatial structures and hierarchical ones, frequency control strategy and fixed strategy for comparison. A number of agents can search the shortest path in parallel. And 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 quality of threshold increases monotonously, merging some pieces of information (threshold) means only selecting the best one among them. It relieves the expectation cost. The pseudo-coded algorithm is the following:

```
do in parallel {
  while ( global bag is not empty ) {
    pick up a candidate from global bag
    expand it
    if ( a new node is better than threshold )
      update local threshold and 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 node, the difference between the value of current threshold



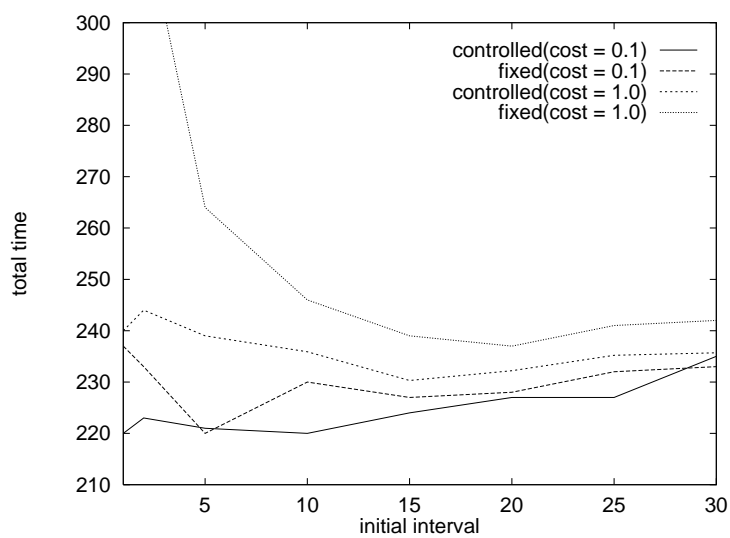
(a) execution time



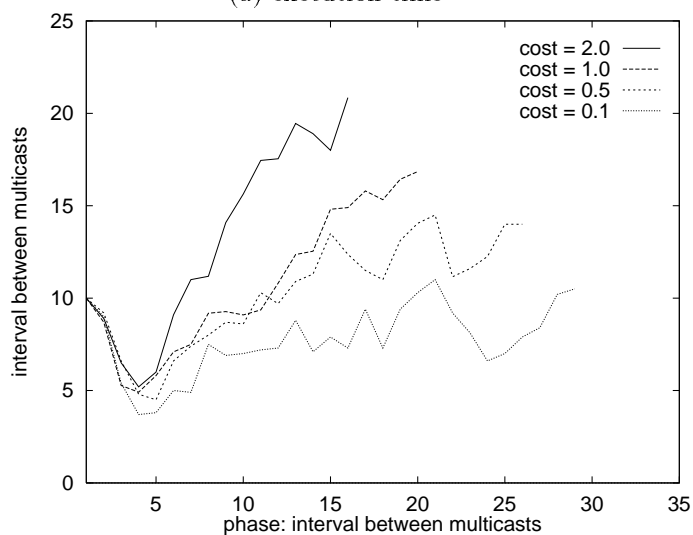
(b) changes of the communication range

Figure 6: the result of spatial strategies

and the one before the expansion is stored in the history memory of an agent. Since the history holds the revision of information, we can calculate the expected value of information that other agents get most recently. Thus a Monte Carlo simulation on the history gives the agent the expected best value among n agents. At every step, agents select a communication structure. Since the simulator models asynchronous



(a) execution time



(b) changes of the broadcast frequency

Figure 7: the result of frequency strategy

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

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 achieve good performance against affected communication costs. A result is shown in Figure 6 and 7. The communication cost used in Figure 6 (a) is $0.01 + 10^{-5}x + 10^{-9}x^2$, where x is the number of

all communications (packets) in the step. We select it rather than a linear function, because it shows non linearity if the number becomes large. This non-linearity is introduced to imitate the congestion on network. And we ignore the cost of getting subproblems from the global bag. In Figure 6 (b), the solid line shows range control strategy on flat structures, the dashed line shows the optimal size given by control structure on hierarchical structures, and the dotted line is the real size of clusters on hierarchical structures. Strictly speaking, the communication cost used in Figure 6 (b) is different from the one in Figure 6 (a). The cost function hardly affects the tendency of range change. All points are average values of $10 \sim 100$ data.

The properties we can find are summarized as:

- (i) Changing the range and the frequency during the execution is useful. The initial range hardly affects the performance in both structures. As Figure 6 (b) shows, there are three stages from the view of the size of ranges on both structures. In the first stage (at $0 \sim 20$ step) and the last stage (over about 150 step) the communication range is relatively small. This result is derived from the nature of the searching process. At the initial stage, for no agent find a new threshold, the strategy reduces the communication range. In the middle stage (at $20 \sim 150$ step) where the distribution of agents' information becomes large, the range becomes very large. In particular, the strategy on hierarchical structures changes sharply, because an information collector of a cluster controls the size of the cluster. The best answer was often found at $80 \sim 150$ step. At the last stage after finding and broadcasting the best answer, updating the threshold never occurs and no more communication is needed. In conclusion, the more frequently information is renewed, the more frequently or more widely identical agents communicate. But excessively frequent updates decrease the number of communication.
- (ii) 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.
- (iii) 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 on hierarchical strategies. This reduces the merit of hierarchical structures. However experiments with other cost functions showed large communication costs make the difference clear.

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

4. Discussion

In this section, we discuss the applicability of these strategies to other fields and ways of implementing these strategies. Related works are also described.

4.1. applicability of the strategies

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

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

Furthermore, the strategies can be applied not only to cooperative search. They can be thought of as algorithms for managing the consistency of an object that is shared *weakly* between agents. In addition, the idea of 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 on distributed systems. Thus the programmers' burden of keeping coherency effectively will be relieved in a number of application areas.

In this case, the measure of execution is the ratio of local access and remote access. Exchanging values 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 would work well.

Here we summarized the limitations of a proposed scheme:

- (i) Communication cost must be known before execution. It would be difficult on the system that has hierarchical topology or a large-scale open system.
- (ii) The strategies assume homogeneity of agents. We describe a solution to it in the next section.

4.2. separation of communication strategy from problem-level program

Communication strategy changes the behavior of an agent by using a history that is a measure almost independent of a problem-level program. Thus we think deciding communication structures should be a kind of computational reflection [20]. This means communication strategy could be separated from problem-level programs and would be a metal-level description of program solving agents.

This separation has two merits:

- (i) Programming is divided into two independent subtasks: building communication strategy and writing problem-solving codes. As a result, writing codes for communication strategies can be omitted from problem-solving programmers' task. It also makes reusing the communication strategy easy. Communication strategy can be used as a library.

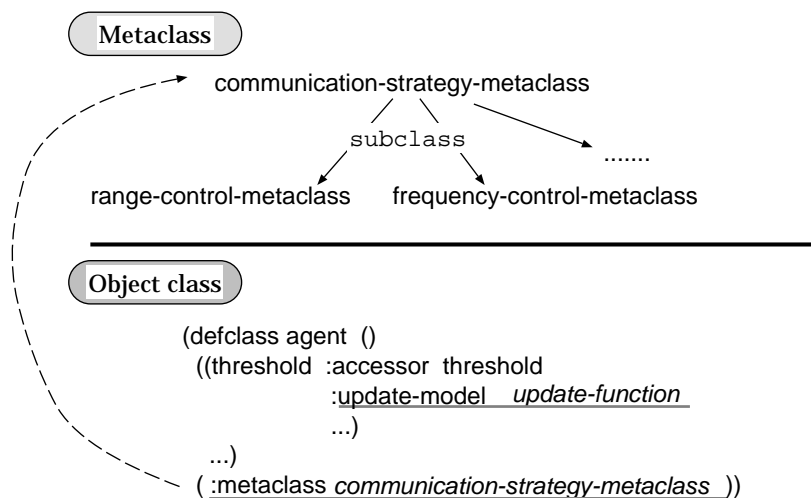


Figure 8: description with metaclass

- (ii) Since communication strategy is included in the semantics of language, the strategy could be invoked automatically when needed. For example, if a communication strategy is defined as metalevel computation of the assignment to a slot that holds information to exchange, all modification of the slot invokes the strategy. We can omit explicit invocation codes from problem-level program.

Some object-oriented-programming languages have meta objects that represent behavior of objects. In particular CLOS (Common Lisp Object System)[23] has flexible interface to meta level programming, which is known as CLOS MOP (Meta Object Protocol) [17]. Thus we have re-implemented our TSP simulator in CLOS.

Figure 8 shows the relation of classes in our system. 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. A programmer declares one of these strategies as the metaclass of his agent class. He should also add a keyword parameter **:used-in-cooperation** to the slot that is the measure of execution. The metalevel class adds some methods for managing its history into the agent class at the definition time. And it invokes the method for communication strategy automatically when the agent wants to send information. A communication cost function is given in creating its instances by users.

When an assignment to the slot that holds information to exchange are done, communication strategy as a part of new semantics of the assignment is invoked automatically. Deciding communication structure and sending information is done in meta-level.

4.3. related work

Huberman stated that cooperation between searching agents improves performance through theoretical analysis[12,13]. Hogg also researched cooperation in a graph problem[11]. But they did not mention how to communicate with each other and did not consider communication costs. Their works are a kind of cooperation in parallel activities. Need for cooperation is not emerged from the constraint of execution environments but from the property of problems; the communication structure for cooperation depends on the problem itself.

Sycara *et al.* researched distributed search [24]. They have proposed a method of selecting a piece of information to send among agents. The structure of job assignment to agents is fixed during the execution.

Durfee *et al.* proposed a cooperation method in a distributed vehicle monitoring system [4]. In their model, the search space is divided by view ranges of agents. Information exchanged between agents is row data, the path of a detected vehicle or plan. The abstract level of information affects the performance. This model has two points different from our TSP example. First, tasks for agents are divided in a fixed way before execution. It is result-sharing rather than task-sharing. And they did not propose a way of selecting communication structure dynamically. Second, they changed the form of cooperation by changing the abstract level of exchanged information.

The relations between OOPL or meta object and Multi Agent System (MAS) has been discussed in [5, 6, 9]. But separation of communication policy using metaobject from a problem-level program has not been proposed. We think that the separation is a step toward multiagent oriented programming language from OOPL.

5. Conclusion

In this paper, we have mentioned communication strategies based on partial histories of agents for modeling their environment to select efficient communication structures dynamically. The results of experiments show that strategies give good communication structures to autonomous agents. We expect that local history as a qualitative model of revision of information, which is a measure of execution, is useful for MAS.

Though there are some restrictions such as the implemented strategies assuming homogeneity of agents, the simplicity of combining some pieces of information, and the information of the communication cost function, we think that these strategies can be used in many applications that acquire new information in execution time, if we can find information which value increases monotonously and can build a communication cost function. These would contain distributed database systems that replicate data.

The history-based expectation mechanism could apply to not only homogeneous environments but also heterogeneous ones. The history of the revision of information at other agents in current implementations is not separated from its own history. Agent exchange the information with each other and update its history with the received information. But in heterogeneous network, this dispersion would become an important problem. The implementation and the evaluation of strategies based

on separated histories are a future plan. It will also work on heterogeneous agents.

In this paper, we proposed a cooperation scheme on distributed systems. However cooperative processing is not only useful on distributed systems but also on concurrent systems. One example that requires cooperation, even if the communication cost can be ignored, is a parallel search based on genetic algorithm. A search process would fall in a local minimum position by over-distribution of the best code at each step. Thus broadcasting the best code pattern does not necessarily lead to the optimal form of computation. The same discussion is held on heuristic-base parallel search. In these examples, cooperation changes the way of sharing information between agents. Thus we can define cooperation as methods for selecting an appropriate structure of processing elements. Its application is not restricted to distributed systems. We think that local-history based methods like ours would be useful for building schemes on such systems.

References

- [1] Nicholas M. Avoris and Les Gasser, editors. *Distributed Artificial 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 [15], 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/December 1991.
- [7] Stephanie Forrest, editor. *Emergent Computation*. The MIT Press, 1991.
- [8] Les Gasser. Boundaries, identity, and aggregation: Plurality issues in multi-agent 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 Problems 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] Bernardo A. Huberman and Tad Hogg. The behavior of computational ecologies. In Bernardo A. Huberman, editor, *The Ecology of Computation*, pages 77–115. Elsevier Science Publishers B.V.(North-Holland), Amsterdam, 1988.
- [15] Michael N. Huhns, editor. *Distributed Artificial Intelligence*. Morgan Kaufmann, 1987.
- [16] Toru Ishida, Les Gasser, and Makoto Yokoo. Organization Self-Design of Distributed Production Systems. *IEEE Transactions on Data and Knowledge Engineering*, 4(2):123–134, 1992.
- [17] Gregor Kiczales, Jim des Rivières, and Daniel G. Bobrow. *The Art of the Metaobject Protocol*. The MIT Press, 1991.
- [18] E.L. Lawler, J.K. Lenstra, D.B. A.H.G. Ronnoony Kan, and Shmoys, editors. *The Traveling Salesman Problem*. Addison Wesley, 1985.
- [19] Victor Lesser and D. Corkill. Functionally accurate, cooperative distributed systems. In Bond and Gasser [2], chapter 4.3.1, pages 295–310.
- [20] Pattie Maes and Daniele Nardi, editors. *Meta-Level Architectures and Reflection*. North-Holland, 1988.
- [21] Herbert A. Simon. *The Sciences of the Artificial*. The MIT Press, Boston, second edition, 1981.
- [22] 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.
- [23] Guy L. Steele Jr. et al. *Common Lisp the Language*. DEC press, second edition, 1990.
- [24] Katia P. Sycara, Steven F. Roth, Norman Sadeh, and Mark S. Fox. Distributed constrained heuristic search. *IEEE Trans. Syst. Man Cybern.(Special Issue on Distributed AI)*, 21(6):1446–1461, 1991.