TM-1063

Group Problem Solving by Autonomous Agents for Task Allocations Problem

by
S. Shindo & M. Sumida (Mitsubishi)

June, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Group Problem Solving by Autonomous Agents for Task Allocation Problem

Seiichi SHINDO Masato SUMIDA

Central Research Laboratory
Mitsubishi Electric Corporation
8 1-1, Tsukaguchi-Honmachi. Amagasaki, Hyogo, 661 JAPAN

Abstract

We have developed a group problem solving method called ITM(Iterated Task Migration) for dynamic task allocation problems. ITM takes unoptimal task allocation to agents as input, then the agents try to change their task allocations toward the globally optimal allocation by iteration of task migrations. The cooperation by the agents offers the advantages of adaptability and parallelism, which contribute to solving the dynamic task allocation problems. ITM works in two ways, in a synchronized way(called ITM-S) and in an asynchronized way(called ITM-A). To evaluate ITM-S and ITM-A, we have applied them to the trouble recovery problem in the area of transportation. This paper describes the algorithms of ITM-S and ITM-A, and their evaluations by the experimental results.

1 Introduction

Task allocation problems, which decide allocation of tasks to resources to achieve constraint-satisfied or optimal/semi-optimal solution, are one of fundamental problems in many fields such as engineering. They are divided into two categories:

- *the static characteristic problem*: generation of a plan before the allocated tasks are executed.
- *the dynamic characteristic problem*: modification of the plan during the execution of the tasks because of changes of situation.

How to reduce the amount of computation by avoiding combinatorial explosion is important in solving them. For this requirement, two approaches have been researched, the one is based on mathematical programming[7] and the other is based on knowledge engineering[2][4]. In both of them, the dynamic problems in the above categories are less studied than the static problems. One of the major requirements for the dynamic problems is quick response. If in the time required to reach a solution the situation changes, then the solution can no longer be applied. Difficulty in solving the dynamic problems mainly comes from the treatment of time.

For the dynamic problems, we have taken group problem solving approach[1]. In this approach, a group of autonomous entities called agents(e.g., men or computers) act cooperatively, so that the group, as a whole, keeps its consistencies or achieves its goals.

The advantages of this approach are adaptability to environments and parallelism, which contribute to solving the dynamic problems.

Researches in the group problem solving have been done. However, they are concepts[6] or languages[3][5], therefore, they are too general to supply appropriate representation or effectiveness when they are applied to specific tasks or domains. In order to establish methods for a speccific task, we have developed a group problem solving method especially for dynamic task allocation problems, called ITM(Iterated Task Migration).

This paper describes the framework of *ITM* and its evaluation by experimental results. The research was done as part of the Fifth Generation Computer Systems project of Japan.

2 Task Allocation Problems

In this section, we show the task allocation problem ITM treats. Some terms are defined below:

- *a set of resources:* $RS = \{R_1, R_2, \ldots, R_m\}$
- *a set of tasks:* $TS = \{T_1, T_2, \ldots, T_n\}$
- *a subset of the tasks, allocated to* R_i*:* $TALC(R_i)$
  for $\forall i, \forall j (i \neq j)$, $TALC(R_i) \cap TALC(R_j) = \phi$
- R_i's function for evaluation: $EVF_{R_i}(TALC(R_i))$

  A resource uses its EVF to evaluate the allocated tasks. EVF_{R_i} maps $TALC(R_i)$ onto non-negative value. EVFs may be either identical or different among the resources.

The problem is as follows:

Given RS, TS and $EVF(R_i)$ for $\forall i$, then find $TALC(R_i)$ for $\forall i$ which minimize $\sum_{k=1}^{m} EVF_{R_k}(TALC(R_k))$ as possible. In addition, resources or tasks may be added or removed before the solution is found.

3 Overview of ITM

For the problem described in the previous section, ITM, based on mutual interaction among autonomous agents, tires to change initial task allocations towards better ones. ITM takes unoptimal task allocation to agents(corresponding to resources one to one) because of changes of resources or tasks, as input. Then the agents try to change their task allocations toward the globally optimal allocation by iteration of task migrations among them through the exchanges of messages. As the iteration proceeds, the result of the re-allocation is improved. A sketch of ITM's behavior is shown in figure 1.

The agents are assumed to be able to communicate each other. Each of them locally has the information on the current tasks allocated, its EVF and other agents' names. Other information can be acquired by the exchanges of the messages. The control

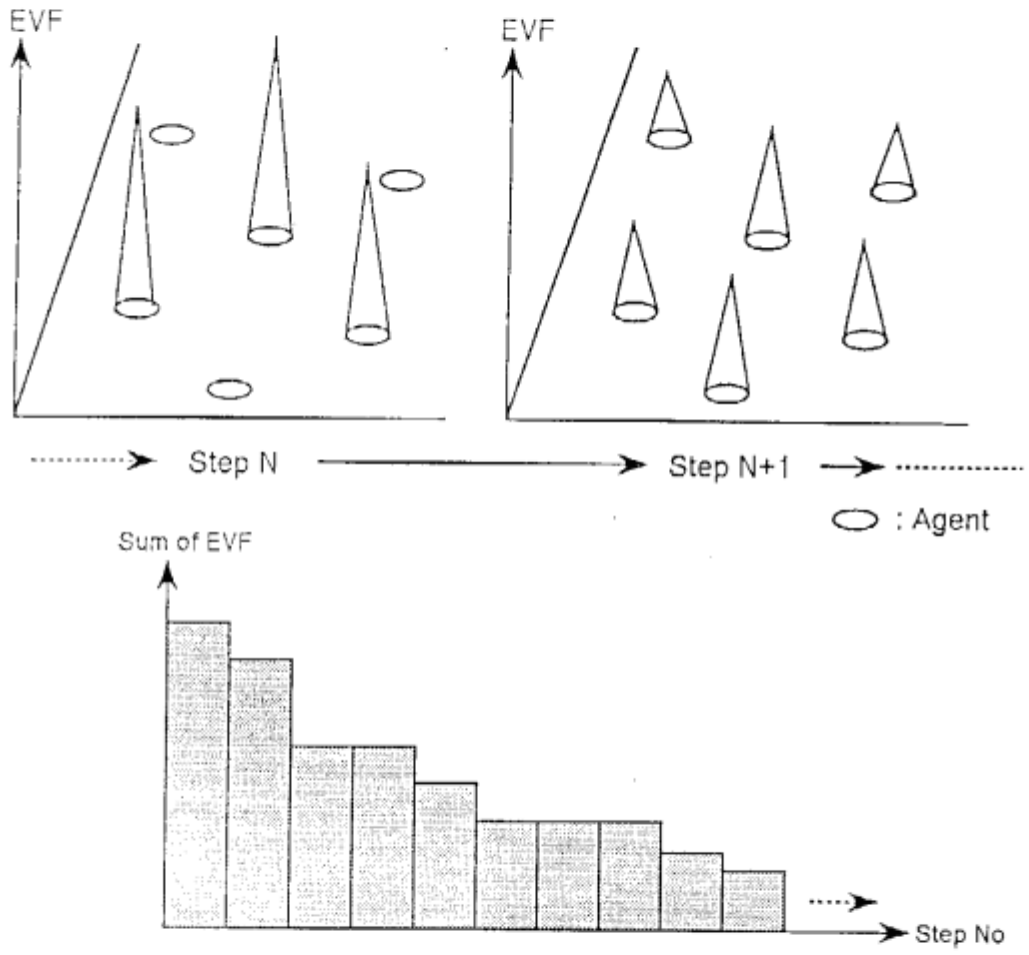

Figure 1: Sketch of Behavior of ITM

mechanism for these cooperative agents is an extension of contract net[3], i.e., a sequence of task announcement, bid and award. We call this sequence a task migration step.

Since ITM is based on a group of loosely-coupled autonomous agents, it offers the following advantages:

- *adaptability:* Agents are rather independent, and each of them has the cooperation mechanism, so that they continue to search for a better task allocation, in spite of changes of agents or tasks during the search.
- *parallelism:* Each agent may act in parallel, so that it needs less time to reach the new task allocation.
- *built-in search strategy:* In general, problem solving for these kinds of problems requires knowledge about controlling search, to avoid combinatorial explosion. However, there are problems which are too complex to extract the knowledge. ITM's built-in search strategy can be applied to such hard problems.

ITM works in two ways, in a synchronized way(called ITM-S) and in an asynchronized way(called ITM-A). In the next two sections, they are described in detail.

4 ITM-S

In ITM-S, the task migration step is iterated synchronously among agents. First, the task announcement phase occurs simultaneously in all agents, followed by the bid phase, then the agents change their task allocations. After that, the next task migration step begins simultaneously.

Here are shown the definitions of some terms used in the description of ITM.

- *overloaded* : An agent whose EVF value is positive is called an overloaded agent.
- $dEVF(A, SignT)$: The amount of the decrease of an agent A's EVF value because of migration of a task T. Negative value represents the increase of the EVF value. Sign is "+" or "−". "+" means that T is added to A. "−" means that T is removed from T.

The procedure for one task migration step of ITM-S is shown below:

1. *Task announcement:* An agent A_P checks if it is overloaded by computing $EVF_{A_P}(TALC(A_P))$.
   - *If A_P is overloaded:* A_P selects a task T_P from the current A_P's tasks, such that $dEVF(A_P, -T_P) > 0$ and $dEVF(A_P, -T_P)$ is the largest. Then A_P broadcasts the task announcement message including the contents of T_P and $dEVF(A_P, -T_P)$, to call for candidates to execute T_P instead of A_P. If A_P cannot findsuch a task, A_P broadcasts the retirement message to show that A_P doesn't call for candidates in this task migration step,
   - *If A_P is not overloaded:* A_P broadcasts the retirement message.

2. *bid:* By receiving the task announcement messages or the retirement messages from all other agents, each of the agents can classify that the other agents into two classes, i.e., a manager(calling for candidates) and a contractor(not calling for candidates). An agent A_Q reacts to the task announcement message from A_P as follows: A_Q computes $CON(A_P, A_Q)$ and replies to A_P with the bid message including $CON(A_P, A_Q)$. CON represents the amount of the change of sum of two agents' EVF values by migration of a task from the one to the other. $CON(A_M, A_C)$ is equal to

$$dEVF(A_M, -T_M) + dEVF(A_C, +T_M)$$

where A_M is a manager, A_C is a contractor, and T_M is one of A_M's tasks.

3. *bid sharing:* The managers receive the bid messages including CON values. Then, to assure globally conflict free and optimal task allocations in this task migration step, they share their CON values by sending the messages each other.

4. *award:* After sharing CON values about all agents, each of the managers independently tries to find an agent to which its announced task is allocated, under the condition of satisfying no-conflict and optimality. There may be some managers who cannot find such an agent because of this condition. After the selection, each of the managers broadcasts the award messasge to show which agent is selected, then the involved agents change their task allocations.

Sum of all agents' EVF values decreases by sum of CON values involved in the task migrations. The iteration of the task migration step terminates when each of the overloaded agents has no task to be announced.

Behaviors of the agents according to ITM-S is shown in figure 2.

- *In the task announcement phase:* An overloaded agent A_1 selects a task T_1 such that $dEVF(A_1, [-T_1])$ is the largest, then sends the task announcement message to A_2, A_3 and A_4. A non-overloaded agent A_2 sends the retirement message to A_1, A_3 and A_4.

- *In the bid phase:* A_2 computes $CON(A_1, A_2)(= dEVF(A_1, -T_1) + dEVF(A_2, +T_1))$, then replies to A_1 with the bid message including $CON(A_1, A_2)$. A_2 also replies to A_3 with the bid message including $CON(A_3, A_2)$. A_1, though it is overloaded, replies to A_3 as well.

- *In the bid sharing phase:* A_1 sends $CON(A_1, A_2)$, $CON(A_1, A_3)$ and $CON(A_1, A_4)$ to A_3. A_3 sends $CON(A_3, A_1)$, $CON(A_3, A_2)$ and $CON(A_3, A_4)$ to A_1.

- *In the award phase:* By using the CON values, A_1 finds A_2 to allocate T_1 and broadcasts the award message telling that T_1 is re-allocated from A_1 to A_2. Similarly A_3 broadcasts the award message telling that T_3 is re-allocated from A_3 to A_4. these task re-allocations decrease $CON(A_1, A_2) + CON(A_3, A_4)$ from sum of EVF values of all agents.

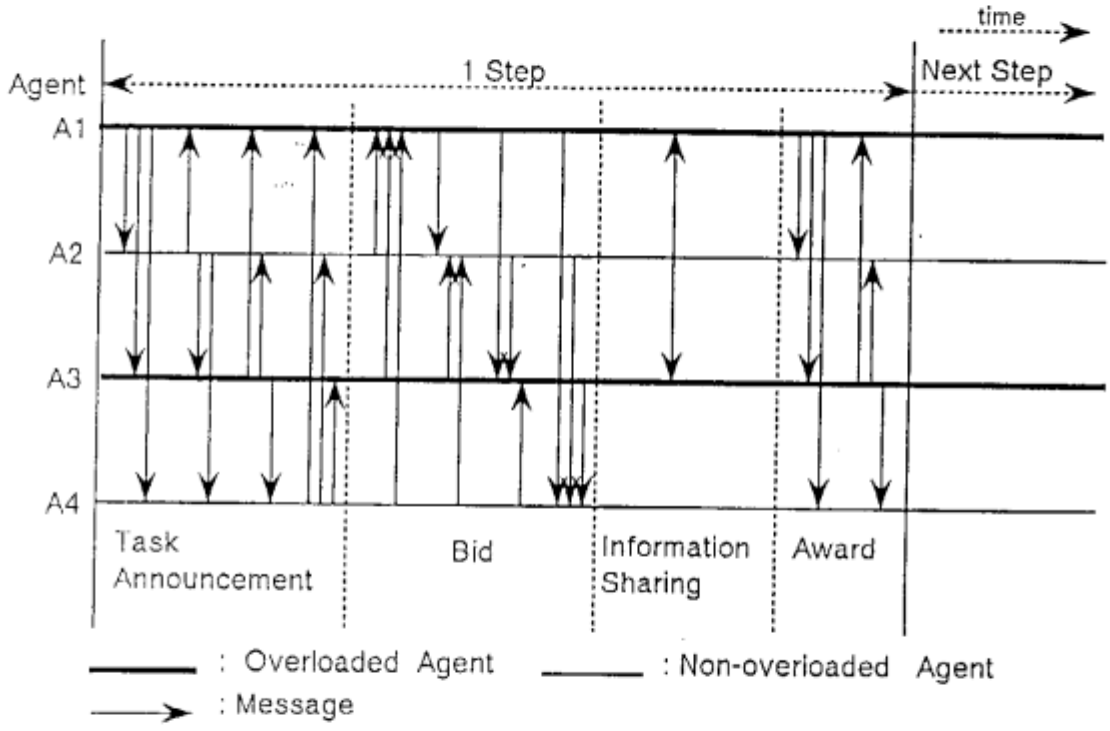

Figure 2: Behavior of ITM-S Agents

The advantage of ITM-S is adaptability. Since every task migration step begins simultaneously, therefore, at that time, ITM-S is open to increase or decrease of agents or tasks during the problem solving. For instance, An overloaded agent because of unexpectedly allocated tasks tires to decrease sum of EVF values of all agents by sending task announcement message.

5 ITM-A

In ITM-A, agents act asynchronously, e.g., some are in the task announcement phase, while others are in the bid phase. Similarly to ITM-S, the task migration step begins with checking if each of the agents checks if it is overloaded.

(i) Procedure for non-overloaded agent

Consider a non-overloaded agent A_P.

1. *TA-waiting:* A_P waits for the task announcement message from other agents.
2. *bid:* When A_P receives the task announcement message from an agent an overloaded agent A_Q, A_P evaluates the announced task by computing $CON(A_Q, A_P)$. If $CON(A_Q, A_P) > 0$, A_P replies to A_Q with the bid message including $CON(A_Q, A_P)$, to register as the candidates for executing the announced task. Because, for A_P to execute the announced task instead of A_Q reduces the sum of EVF values of A_P and A_Q. After that, A_P waits for the award message in the A-waiting stage. If $CON(A_Q, A_P) \leq 0$, A_P replies to A_Q with the rejection message, to show that A_P gives up registering as the candidates. After that, A_P waits for another task announcement message in the TA waiting phase again.
3. *A-waiting:* A_P waits for the award message from A_Q.
4. *award:* A_P receives the award message from A_Q. If the task is allocated to A_Q, A_Q modifies its task allocation.

To prevent double bidding, Any task announcement messages reached in the A-waiting phase or the award phase, is replied with the rejection message.

(ii) Procedure for overloaded agent

Consider a non-overloaded agent A_Q.

1. *task announcement:* Similarly to ITM-S, A_Q selects a task to be announced and broadcasts the task announcement message to call for the candidates. If A_Q finds no task to be announced, it sends no message.
2. *B-waiting:* A_Q waits for the replies(bid messages or the rejection messages) from the other agents.
3. *award:* After receiving all the replies, A_Q selects an agent A_R from the candidates, such that $CON(A_Q, A_R)$ is the largest

   Then A_Q sends the award message to all of the candidates to show that the announced task is allocated to A_R.

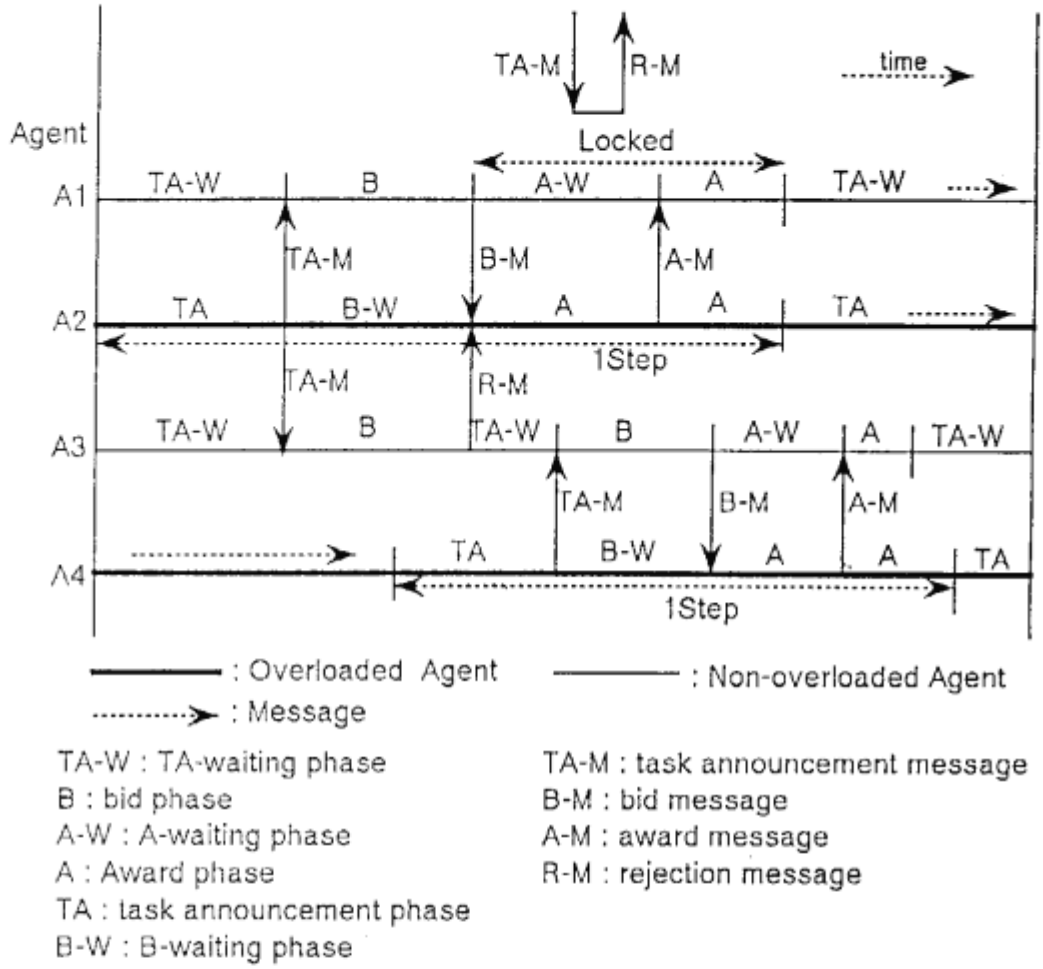

Figure 3: Behavior of ITM-A Agents

The iteration of the task migration step terminates when each of the overloaded agents has no task to be announced.

Behaviors of the agents according to ITM-A is shown in figure 3. An overloaded agent A_2 broadcasts the task announcement message. Agents A_1 and A_3, which are in the TA-waiting phase, evaluate the announced tasks. Since $CON(A_2, A_1) > 0$, A_1 replies to A_2 with the bid message, then waits for the award message from A_2. Since $CON(A_2, A_1) < 0$, A_3 replies to A_2 with the rejection message, then waits for another task announcement message. A_2, receiving the bid message only from A_1, sends the award message to A_1, then modifies its task allocation and goes to the next task migration step. A_1, receiving the award message from A_2, modifies its task allocation and waits for another task announcement message. A_3 receives the task announcement message from A_4. This time, since $CON(A_4, A_3) > 0$, A_3 replies to A_4 with the bid message, then, the interaction between A_3 and A_4 follows like that between A_1 and A_2.

Comparing with ITM-S, ITM-A runs faster. This comes from less amount of time for synchronization and less amount of messages for information sharing. However, since ITM-A decides task allocations by less amount of information, ITM-A generally reduces sum of EVF values of all agents less than ITM-S.

6 Application

To evaluate ITM, we have applied ITM to the trouble recovery problem in the area of transportation. This section describes the problem domain.

Consider a group of trucks, each of which is delivering shipments with due-date requirements to many destinations. Initially, the shipments are appropriately allocated to the trucks so that they are delivered before their due-times. During the delivery, a trouble happens in one of the trucks, so that it can no longer carry its allocated shipments. The problem is to acquire new shipment allocations to the remaining trucks, in order to minimize the sum of over-due-times of all shipments because of the trouble.

The correspondence between ITM and the problem domain is shown below:

- *a task* → a shipment.
- *an agent* → a truck.
- *EVF* — sum of the over-due-times of the shipments allocated to the truck. This value depends on the routing, i.e., the order of visiting the destinations. Each truck re-schedules its routing to minimize its total over-due-time.

In ITM, first, the shipments allocated to the truck in the trouble, are distributed randomly, then the task migrations follow towards minimizing sum of the over-due-times of the all shipments.

7 Experimental Results

We have implemented ITM on the parallel inference machine MultiPSI[8], developed by ICOT. MultiPSI executes programs written in the parallel logic programming language KL1[8] in prallel, by a group of processor elements(called PEs) with no shared memory. We use 16PEs for our experiments with 15,30,60 trucks and about 1,200 shipments. The trucks' actions according to ITM are distributed onto PEs equally.

Figure 4 shows the total execution times and the final solution(sum of delay time).

The total execution time by ITM-S is from 5 to 11 times much as that by ITM-A. ITM-S gives better final solution than ITM-A in the experiments with 15 and 60 trucks, and ITM-A gives better solution in the experiment with 30 trucks.

One of the criteria for quick response in dynamic problems is high speed and, if possible, good solution. From this viewpoint, ITM-A is more suitable than ITM-S, because of its extremely high speed comparing with ITM-S.

Another criterion is interactiveness, i.e., how easily or quickly the changes of the problem can be reflected in the course of the problem solving. ITM-S can reflect such changes at every point of the beginnings of the task announcement phases. ITM-A

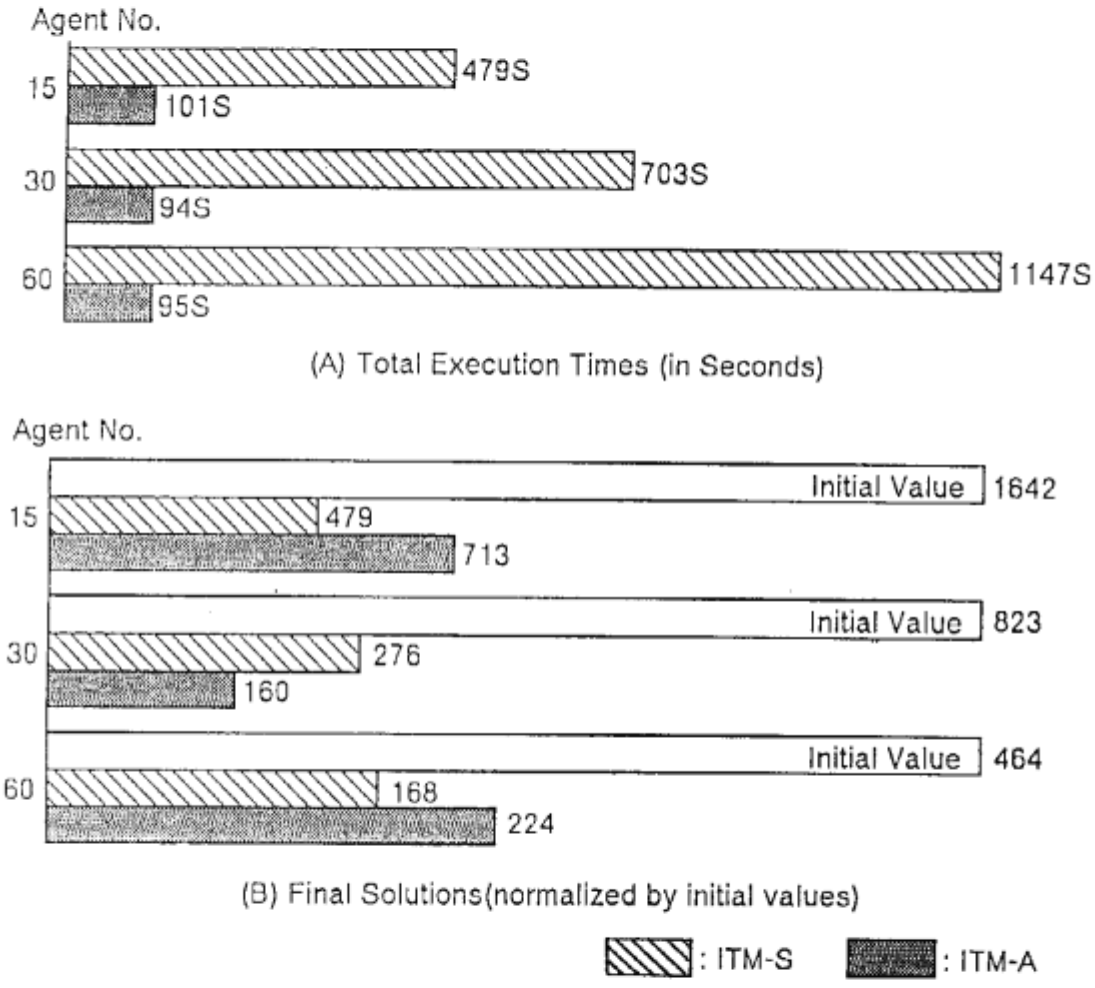

Figure 4: Experimental Results

cannot reflect such changes, until the final solution without reflecting the changes, is acquired. Table 1 compares the response times of ITM-S and ITM-A, i.e., compares the average times for 1 task migration step by ITM-S with the total execution times by ITM A. This table shows that, in all the cases, the response times by ITM-S is smaller than those by ITM-A. From the viewpoint of interactiveness, ITM-S is more suitable than ITM-A, because of its quicker response comparing with ITM-A.

agent no.	ITM-S	ITM-A
15	18s	101s
30	37s	94s
60	71s	95s

Table 1 Response times(in seconds) by ITM-S and ITM-A

8 Conclusion

This paper has described group problem solving methods ITM-A and ITM-S, for dynamic task allocation problems. Quick response is critical for the dynamic problems. Our experimental results have shown that ITM-S improves the adaptability of the group of the agents, while ITM-A improves the execution time of the group. The two factors contribute to improving the quick response.

ITM, based on a group of autonomous agents, can be well applied to distributed environments such as production scheduling among distributed factories and task distribution in computer networks.

On the other hands, ITM doesn't assure the optimality of its solutions. Therefore, ITM cannot be applied to the problems requiring strict optimality of the solutions. Another limitation is the number of agents. The more the number of the agents is, the more the amount of communication grows, which cause the drop of the response time. Clarifying the degree of the quality of the solution, and the relationship between the number of agents and the response times, remains to be done in the future.

Acknowledgment

This research was done as one of the subprojects of the Fifth Generation Computer Systems(FGCS) project. We would like to thank Dr.K.Fuchi(Director of ICOT) for the opportunity of doing this research and Dr.K.Nitta(Chief of 7th Laboratory) and Dr.N.Ichiyoshi(Deputy Chief of 7th Laboratory for their advice and encouragement.

References

[1] Bond,A.H. and Gasser,L. *"Readings in Distributed Artificial Intelligence"* Morgan Kaufmann Publishers,Inc., California, 1988.

[2] Cammarata,S., McArthur,D. and Steeb,R *"Strategies of Cooperation in Distributed Problem Solving"* Proc. of the 8th International Joint Conference on Artificial Intelligence, 1983, pp.767-770.

[3] Davis,R. and Smith,R.G. *"Negotiation as a Metaphor for Distributed Problem Solving"* Artificial Intelligence Vol.20, 1983, pp.63-109.

[4] Fox,M.S. and Smith,S.F. *"ISIS – a knowledge-based system for factory scheduling"* Expert Systems Vol.1, No.1, 1984, pp.25-49.

[5] Gasser,L., Braganza,C and Herman,N. *"MACE: A flexible Testbed for Distributed AI Research"* Distributed Artificial Intelligence, Pitman Publishing/Morgan Kaufmann Publishers,Inc., London/California, 1987, pp.271-287.

[6] Hewitt,C. *"Offices are open systems"* ACM Tracsactions on Office Information Systems, Vol.4, No.3, July 1986, pp.271-287.

[7] Ibaraki,T. and Katoh,N. *"Resource Allocation Problems: Algorithmic Approaches"* The MIT Press, Cambridge, Mass., 1988

[8] Nakajima,K., Inamura,Y., Ichiyoshi,N., Rokusawa,K. and Chikayama,T *"Distributed Implementation of KL1 on the Multi-PSI/V2"* Proc. of the 6th International Conference on Logic Programming, 1989, pp.436-451.