Improving The Optimal Solution Attainment Rate by Multiplexing Method

Yasuki Iizuka

School of Science, Tokai University, Kanagawa Japan

Summary
Distributed constraint optimization problems (DCOP) have attracted attention as a means of resolving distribution problems in multiagent environments. We [1] proposed a multiplex method targeting the improved efficiency of a distributed nondeterministic approximation algorithm for distributed constraint optimization problems. The multiplex method targeting the improved efficiency of a distributed nondeterministic approximation algorithm has been proposed for distributed constraint optimization problems. Since much of the computation time is used to transmit messages, improving efficiency using a multiplex computation of distributed approximation algorithms might be feasible, presuming that the computation time of each node or a small change in message length has no direct impact. Although it is usually impossible to guarantee that the approximation algorithm can obtain the optimal solution, we managed to do so, using a theoretically determined multiplex method. In addition, we show the feasibility of an optimal solution attainment rate of 0.99 by an experiment using a Distributed Stochastic Search Algorithm.
Key words:
Distributed Constraint Optimization Problem, Optimal Solution

1. Introduction

Distributed Constraint Optimization Problems (DCOP) are fundamental frameworks in distributed artificial intelligence and have recently attracted considerable attention [2, 3]. As for DCOP, Traffic control, Sensor Network, Multi-Robot System , Smart Grid, Disaster Evacuation Assist, etc. are expected. In the smart grid, distributed control of energy is demanded and the problem matches the framework of DCOP. Several complete and approximation algorithms, [4, 5, 6] and [7, 8] respectively, have been proposed as resolutions for DCOP. However, when tackling real-world problems, high efficiency algorithms are required [9], hence the purpose of this study is to increase the efficiency of distributed approximation algorithms. Although an approximation algorithm has a short computation time, obtaining the optimal solution cannot be guaranteed. In this paper, when an approximation algorithm is repeatedly performed, the probability that the optimal solution will be obtained is called the optimal solution attainment rate. This research targets an approximation algorithm with an optimal solution attainment rate of 0.99. Via the multiplexed execution of one kind of distributed approximation algorithm, despite the use of an approximation algorithm, it is shown that an optimal solution attainment rate can be arbitrarily established.

The next section describes an overview of DCOP. Details of the multiplex method for DCOP and the potential to establish an optimal solution attainment rate in the multiplex method are discussed in Section 3, while in Section 4, the verification results of the multiplex method through an example problem are described.

2. Distributed Constraint Optimization Problems (DCOP)

DCOP is defined as follows [6, 4]. A set of variables x_1, x_2,…, x_n exists, each of which is assigned a value taken from a finite and discrete domain D_1, D_2, …, D_n and each of which is also assigned to multiple agents a_1, a_2, …, a_m. Constraints c_{ij}: $D_i \times D_j \rightarrow \{true, false\}$ are defined between x_i and x_j and a cost function $g_{ij}(x_i, x_j) : D_i \times D_j \rightarrow R^+$ exists for each constraint. The agent a_k only has the following information: information about x_k, which is assigned to a_k, c_{ij}, which is a constraint of x_k, and the cost function g_{k*} . In this case, the purpose of DCOP is to obtain an assignment for variable A that minimizes the summation of the cost function $G(A) = \Sigma g_{ij}(A)$ (Variable n is handled as $n = m$ here.)

In DCOP, an assignment A_o that offers the minimum G(A_o) amongst all possible assignments A is defined as the optimal solution. When an assignment A_p is $l = G(A_p) - G(A_o)$, A_p is defined as a solution of distance l from the optimal solution in this paper. Distance l is one of the measurement bases used to evaluate the quality of the solution. In DCOP, agents whose variables are associated by constraints solve problems by exchanging values of the variable through message transmission.

Well-known algorithms used to solve DCOP include ADOPT [4], DPOP [5], OptAPO [6], and distributed stochastic search algorithm (DSA) [8].

As for a complete algorithm, an optimal solution is guaranteed, despite the extended computing time. When using DCOP for real-world problems, particularly when

Manuscript received May 5, 2013
Manuscript revised May 20, 2013

solving problems involving robotics and sensor networks, problems must be solved in distributed environments with minimal computation resources [10, 8]. To express complex issues, many variables are needed. Under such circumstances, seeking an optimal solution with a complete algorithm is not always the best method, and there is a greater need for a fast and efficient approximation method. The multiplex method was proposed targeting the improved efficiency of a distributed non-deterministic approximation algorithm for DCOP[1].

Related works such as algorithm portfolios exist [11], and a study of the parallel execution of approximation algorithms [12] one decade previously showed that not only could computation times be reduced but the quality of solutions could also be increased. Gomes[13] also showed how random restarts can effectively eliminate heavy-tailed behavior. Ringwelski[14] proposed a system that multiplexes the tree-based algorithm of DisCSP, in order to reduce the risk of heuristics selection. However, only the effect of the experiment was evaluated, despite engaging in theoretical considerations. Compared to these existing related works, the effect of multiplexing the non-deterministic approximation algorithm of DCOP, has the significant feature of being theoretically analyzed using extreme value theory. In this paper we theoretically show the probability of obtaining an optimal solution that can be arbitrarily set according to this multiplex method.

3. Multiplex Method for Distributed Approximation Algorithms

This section explains the multiplex method of the distributed approximation algorithm, and based on the following presumption: Multiplexing, in this research, means the execution of several searches simultaneously without increasing computation resources. As the problem was originally distributed for the DCOP, it is calculated by distributed agents and the use of the multiplex method means each agent's computation resource need not be increased. To avoid confusion with parallel processing, in which multiple computation resources are used, the term "multiplex" is used in this paper.

The computation time and quality of the solution of distributed approximation algorithms are subject to variation.

Most of the computation time in distributed algorithms is spent in message transmission[14]. Measures commonly used to evaluate distributed algorithms include ideal time complexities and communication complexities. Ideal time complexities refer to the overall number of message rounds, while each agent simultaneously exchanges and processes messages until the algorithm stops. Presupposing the adoption of this measure, if the number of messages remains constant, the processing time at each node and a slight increase in the length of each transmitted message will not impact significantly on the execution time of the distributed algorithm.

3.1 Multiplexing distributed approximation algorithm

When the results of trial *S* conform to some kind of probability distribution, if the trial is repeated, and the minimum (or maximum) value of the repeated trials is selected as trial *M*, the latter conforms to a probability distribution different from trial *S*. For instance, if the scores are pointed to by a roll of the dice, the probability distribution becomes 1/6 each of $X = \{1, 2, 3, 4, 5, 6\}$, and the expected value μ_s of 3.5. If a dice is thrown *m* times, or a total of *m* dice are thrown simultaneously and the scores resulting from the minimum value of the dice are taken as trial *M*, the probability distribution will be non-uniform, and as the value of *m* rises, the expected value μ_m will become closer to 1.

```
1: Algorithm S
2: while(TerminationConditionIsNotMet)do
3:     mc := 0; /* Number of Messages */
4:     while(mc < #Neighbors) do
5:         neighborsStatus[mc] := Receive; mc++;
6:     endwhile
7:     x := NewValue(neighborsStatus[ ])
8:     Send(x)
9: endwhile
```

Figure 1. Simple distributed iterative improvement algorithm S

```
11: Algorithm Multiplexed-S
12: while(TerminationConditionIsNotMet)do
13:     mc := 0; /* Number of Messages */
14:     while(mc < #Neighbors) do
15:         neighborsStatus[mc][ ] := Receive; mc++;
16:     endwhile
17:     forall i in m do /* for each plane */
18:         x[i] := NewValue(neighborsStatus[ ][i])
19:     end
20:     Send(x[ ])
21: endwhile
```

Figure 2. Multiplexed distributed iterative improvement algorithm Multiplexed-S

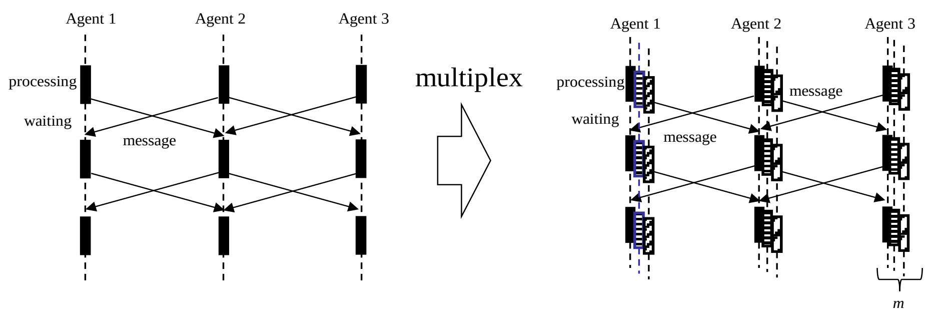

Figure 3. Concept of the multiplexed algorithm

If this fundamental is applied to distributed approximation algorithms, it may help improve their efficiency. Namely, if a distributed approximation algorithm is executed *m* times and the maximum or minimum value is selected as a result or if the algorithm is executed simultaneously in m layers and the maximum or minimum value is selected as a result, it may be feasible to gain a better computational performance.

In this paper, multiplexing the computations and messages at each respective node executing the distributed algorithm is considered, based on the above presumption, without increasing the number of messages.

Fig. 1 is a simple example of distributed iterative improvement approximation algorithm *S* for DCOP. The agent executing this algorithm receives messages from the neighbor agents, and after determining its value *x*, informs the latter of this value. The procedure *NewValue*( ) is assumed to include nondeterministic elements. In this paper, the ordinary execution of an algorithm is considered as a plane. When multiple planes execute the algorithm simultaneously, it is defined here as multiplexing. Fig. 2 shows *Multiplexed-S*, which multiplexes algorithm *S* of Fig 1. The idea of multiplexing is shown in Fig 3. Agents on m planes have independent values *x*[*i*] and searches are performed independently on each plane when multiplexed, and though the length of the message is increased, there is no increase in the number of messages because in the messages exchanged with neighbor agents, the multiplexed value is exchanged in a single message.

In this paper, the number of planes simultaneously executed is defined as multiplicity and expressed as m.

3.2 Analysis of the Expected Multiplexing Results

In this section, the analysis result of the effect acquired by multiplexing is explained. It is presumed that the distance from the optimal solution of the solution of a distributed approximation algorithm *S* follows the probability distribution of the distribution function $F_s(w)$ and the probability density function $f_s(w)$. When algorithm *S* is executed multiply on *m*-plex planes, the minimum value is selected as a result. The execution on *m*-plex planes is presumed to be independent. In this case, the probability that the distance from the optimal *y* can be obtained through multiplicity *m* follows the "minimum value distribution" of the original probability distribution according to the extreme value theory.

The probability distribution function $F_m(y)$ and probability density function $f_m(y)$ are given in the following formulas:

$$F_m(y) = 1 - (1 - F_s(y))^m \tag{1}$$

$$f_m(y) = m(1 - F_s(y))^{m-1} f_s(y) \tag{2}$$

The expected values μ_m and distribution $\sigma_m^2(m)$ of this probability distribution can be expressed as a function of *m*, but an exact solution becomes very complex. Therefore, in general, analyze an extreme value distribution of the minimum value, a type-3 asymptomatic minimum value distribution is used that assumes the existence of a lower limit in the original probability distribution. Assuming the lower limit of the original probability distribution as 0 (= optimal), if a type-3 asymptomatic minimum value distribution is used, the expected values μ_m appear as follows:

$$\mu_m(m) = \frac{\delta}{m^{1/k}} \Gamma(1 + \frac{1}{k}) \tag{3}$$

Similarly, the distribution $\sigma_m^2(m)$ is expressed as follows:

$$\sigma_m^2(m) = \left(\frac{\delta}{m^{1/k}}\right)^2 \left(\Gamma(1 + \frac{2}{k}) - \Gamma(1 + \frac{1}{k})^2\right) \tag{4}$$

Where, Γ( ) is a gamma function, and δ and *k* are parameters that are dependent on the shape of the original probability distribution. If the original expected values of the probability distribution are μ_s, since this is $\mu_m(1) = \mu_s$, let us simplify Formula (3) and substitute 1/*k* for *h* (*h* = 1/*k*). Then

$$\mu_m(m) = m^{-h}\mu_s \tag{5}$$

$\sigma_m^2(1) = \sigma_s^2$ can also be used to describe the distribution

$$\sigma_m^2(m) = m^{-2h}\sigma_s^2 \tag{6}$$

where *h* is an index of the multiplexing effects, and the larger *h* is, the greater the multiplexing effect obtained. From approximation (5), the effects of multiplexing can be expressed as m^{-h}, which is the power of multiplicity *m*, and from approximation (6), the distribution can be expressed as m^{-2h}. Since the index of the effects of multiplexing *h* is the inverse of *k*, it will follow on the shape of the original probability distribution.

The Formula (5) shows that the effect acquired by multiplexing gradually diminished, even if enlarging multiplicity *m*.

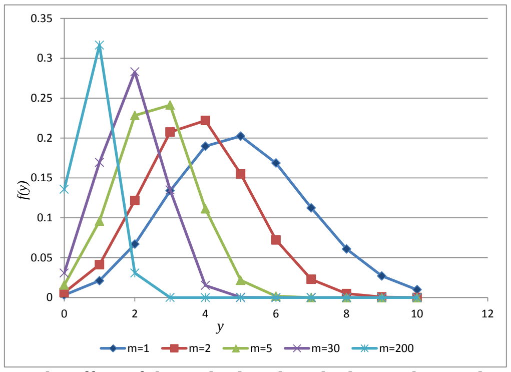

Figure 4. The effect of the multiplexed method given binary distribution of F_s

As shown in Fig.4, if a certain trial is multiplexed, the probability distributions will change. This is the probability distribution at the time of multiplexing by *m*=1, 2, 5, 30, 200 (*m* = 1 is the original probability distribution), when binomial distribution is assumed to the original probability distributions. Thus, distribution is small while the mode of probability distributions will move to the left, if *m* is made to increase. The probability distributions after multiplexing turn into extremum distribution (Weibull distribution) rather than binomial distribution.

3.3 Optimal solution attainment rate and multiplicity

When a distributed approximation algorithm is performed *n* times, suppose that it was n_{opt} times the number of times that the optimal solution was obtained. At this time, $P_{opt} = n_{opt} / n$ is called the optimal solution attainment rate of that algorithm. There follows an argument concerning the optimal solution attainment rate at the time of multiplexed execution of the algorithm.

In [1], we argued that a near-optimal solution was obtained by multiplexed execution of the algorithm. When multiplexed execution is performed, the mode of the probability distributions of the solution obtained diminishes, and likewise the distribution (Fig.4). Subsequently, whether the attainment to the optimal solution can be guaranteed by raising the degree of multiplex is considered. For example, suppose that there is a distributed approximation algorithm with a low attainment rate of the optimal solution. Can the optimal solution certainly be gained by multiplexed execution of this algorithm? In this section, the optimal attainment rate P_{sopt} of the distributed approximation algorithm before multiplexing(m = 1) is assumed to be known.

The distance from the optimal solution of the solution, as calculated by a certain distributed approximation algorithm, presumes that it becomes probability distributions, with the optimal solution the minimum of the latter. When the optimal solution attainment rate obtained by performing this distributed algorithm simply (m = 1) is P_{sopt}, the optimal solution attainment rate P_{mopt} obtained by multiplexed execution is calculated by the following formula:

$$P_{mopt} = 1-(1-P_{sopt})^m \tag{7}$$

By using Formula (7), the required multiplicity m can be calculated by establishing a target attainment rate relative to the optimal solution. For example, m in the case of an optimal solution attainment rate of 0.99, can be calculated as follows:

$$1-(1-P_{sopt})^m \geq 0.99 \tag{8}$$

By using common logarithms here, the conditions of multiplicity *m* become the following formula:

$$m \geq \frac{-2}{\log(1-P_{sopt})} \tag{9}$$

It considers setting an optimal solution attainment rate to 0.99 by the multiplexed execution of the distributed approximation algorithm of the optimal solution attainment rate 0.2. Since *m* is an integer, the condition is $m \geq 21$. Furthermore, the condition is $m \geq 31$ to set an optimal solution attainment rate to 0.999. A logarithm is used to calculate the degree of multiplex, when the conditions of the optimal solution attainment rate 0.9 are $m \geq m_1$, the conditions for an optimal solution attainment rate of 0.99 are $m \geq 2\ m_1$, and those for an optimal solution attainment rate of 0.999 are $m \geq 3\ m_1$.

Although the optimal solution attainment rate can be increased to 0.9999 etc., attainment rate 1 to the optimal solution cannot be guaranteed. However, when raising the optimal solution attainment rate (when multiplicity m is enlarged), the distribution of the probability distributions of the distance from the optimal solution of that originally obtained diminishes, hence the distance from the optimal solution can become very small.

The above calculation shows the following, even if it is a distributed approximation algorithm with a low optimal solution attainment rate, namely that attainment is highly likely to be possible for the optimal solution using multiplexed execution. Although 0.99 was shown here, this rate can be freely established.

3.4. Selection of efficient multiplicity for calculating the optimal solution

It was shown that an optimal solution attainment rate can be established by multiplexed execution of the distributed approximation algorithm by the argument in the previous paragraph. In this section, the case whereby the character of the distributed approximation algorithm not featuring multiplexed execution is an anytime algorithm[8] is considered. If the following two cases are compared, which will be efficient, this will involve the case of large multiplicity with a short execution round, and that of small multiplicity with a long execution round, respectively.

For distributed approximation algorithms that are not multiplexed, according to the computation time, monotonically decreasing distance from the optimal solution is assumed. Let l be the expected value of the distance from the optimal solution, r_e the number of rounds to the end of an algorithm and a and b constants. In algorithm *DSt*, which was used for the experiment in [1], these relations have been approximated as in the following formula:

$$l = ar_e^{-b} \tag{10}$$

When not multiplexing an algorithm, the computation time is proportional to r_e.

First, we would like to denote the optimal solution attainment rate of this algorithm by the function of the number of rounds r_e to a stop. Henceforth, in this section, to simplify the model, the weight of violation of constraints is considered to be 1. Furthermore, it is assumed that the distribution of distance from the optimal solution obtained by the distributed approximation algorithm, with no multiplexed execution, is binomial.
Parameters include the number of constraints n, and the probability $p(r_e)$ by which a certain constraint is not satisfied (when the number of rounds is r_e). At this time, the expected value can be described as follows as a function of r_e based on the binomial distribution definition:

$$\mu_s(r_e) = np(r_e) \tag{11}$$

Since this decreases by $l= ar_e^{-b}$, this can be described by the next formula.

$$np(r_e) = ar_e^{-b} \tag{12}$$

thus

$$p(r_e) = cr_e^{-b} \tag{13}$$

in which c is taken as a constant. The case where the number of violations of constraints in the optimal solution is 0 is considered for simplification. The probability of the calculated solution being an optimal solution can be expressed as a function of the number of rounds r_e to an algorithm stop.

$$\begin{aligned} P_{sopt}(r_e) &= (1 - p(r_e))^n \\ &= (1 - cr_e^{-b})^n \end{aligned} \tag{14}$$

Next, it asks for the minimum multiplicity for the optimal solution attainment rate 0.99 to be obtained as a function of r_e. Here, Formula (9) takes an equals sign and Formula (14) is substituted, whereupon

$$m(r_e) = \frac{-2}{\log(1 - (1 - cr_e^{-b})^n)} \tag{15}$$

By multiplex execution, we assumed that the computation time increases m times in the worst case. When multiplex execution of the algorithm applies and is made to stop at r_e round, the actual computation time becomes the following formula:

$$T_c(m) = r_e \times m \tag{16}$$

When Formula (15) is substituted here, the computation time of the algorithm for multiplexed execution can be expressed as a function of the number of rounds r_e:

$$T_c(r_e) = r_e \times \frac{-2}{\log(1 - (1 - cr_e^{-b})^n)} \tag{17}$$

An example result of numerical analysis using the conditions $0 \le cr_e^{-b} \le 1$ is shown in Fig. 5. Here, a figure is calculated by setting it to $c = 2$, $b = 0.8$, $n = 20$, with r_e set as the horizontal axis and T_c as the vertical axis. The parameter set here is one of the examples. As shown in this figure, Formula (17) may have a local minimum according to the conditions of the parameter. In other words, as a value peculiar to an algorithm, in order to calculate the optimal solution, the most efficient r_e and corresponding m will exist.

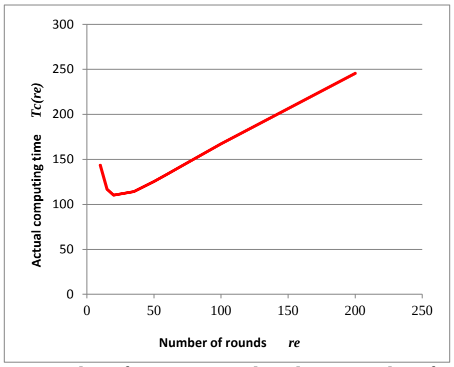

Figure 5. Example of a numerical-analysis result of theoretical calculation

There are some important considerations: m calculated in this section is a value when observing the optimal solution attainment rate, while m calculated in [1] is a value for minimizing computation time. Since this m changes based on the character of the algorithm, detailed analysis is required in future.

4. Verification by an Experiment to Determine the Optimal Solution Attainment Rate

Here, the optimal solution attainment rate by multiplex execution of the approximation algorithm for DCOP is verified by an experiment. A simple distributed approximation algorithm for DCOP is prepared, and an investigation performed to determine whether the optimal solution attainment rate becomes the theoretical value by the multiplexed execution of the same.

4.1. Experimental conditions

In this experiment, as performed in the literature [4, 7, 8], distributed graph coloring problems are solved using a simulator, and the results were evaluated using the number of rounds in which messages are exchanged (ideal time complexity) and the optimal solution attainment rate. A distributed constraint satisfaction problem that lacks a satisfying assignment must be chosen for the distributed graph coloring problem prepared for this experiment, and handled as a DCOP by expressing the condition’s constraint satisfaction as a cost function of {1,0}. The number of agents is 30, the number of constraints is 90, and the number of domains is 3, the constraints graph is a random graph.

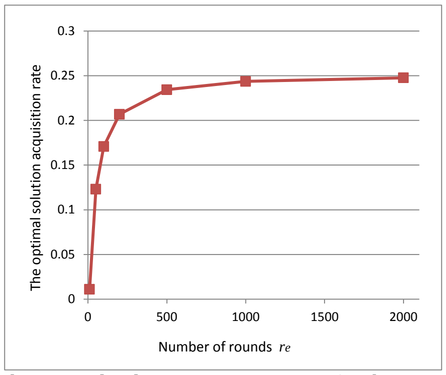

Figure 6. The optimal solution attainment rate (in the case of m = 1) experimental result of an algorithm DSA-B

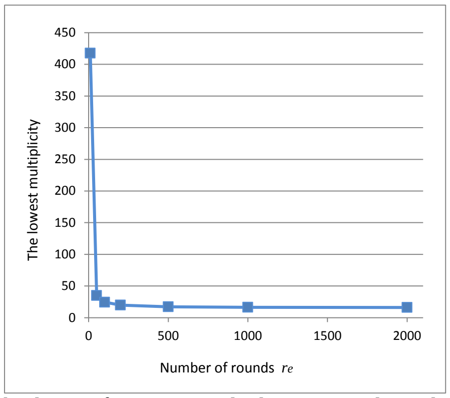

Figure 7. The degree of minimum multiplex; computed in order to obtain an optimal solution attainment rate of 0.99 from an experimental result (Fig. 6)

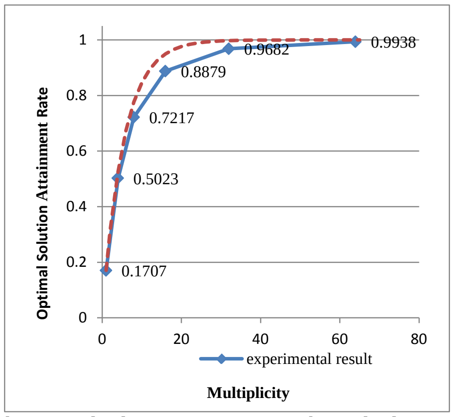

Figure 8. The optimal solution attainment rate by multiplex execution

The distributed approximation algorithm DSA-B[8] was used for this experiment, the parameter of which was set to p = 0.5. In this experiment, since the purpose was to measure the effect of multiplex execution, the simulator was stopped at an arbitrary time and the solution of the plane having acquired the best value was chosen. Moreover, results were compared for the case where an algorithm was not multiplexed (m = 1), and that where it was (m > 1).

4.2. Experimental results

Each of the following experiments involved 1000 prepared problems each being solved 1000 times, and the average of the optimal solution attainment rate being investigated. First, the optimal solution attainment rate of algorithm DSA without multiplexed execution, equivalent to Formula (13), is shown in Fig. 6.

Based on the result obtained here, the minimum multiplicity for obtaining the optimal solution attainment rate of 0.99 was calculated, with the result shown in Fig. 7. From these figures, the multiplicities for obtaining optimal solution attainment rates of 0.99 were respectively set up, and the multiplex execution of the algorithm was performed. The result is shown in Fig. 8. When it stops at 100 rounds, the optimal solution attainment rate 0.99 should be theoretically obtained by *m* = 25. However, this experiment is the average of 1000 problems. Therefore, the 1000 results vary and the optimal solution attainment rate is lower than the theoretical value. When the experiment was performed using one problem, the same curve as the theory was obtained as an experimental result.

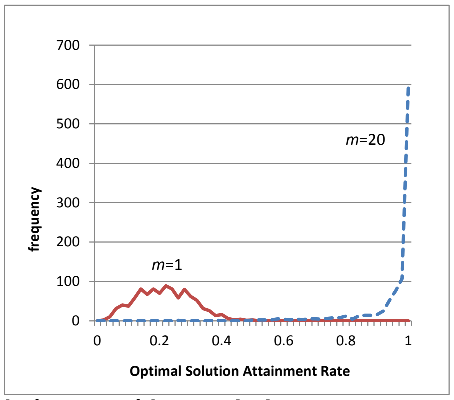

Figure 9. The frequency of the optimal solution attainment rate. (random graph *m* = 1 and *m* = 20)

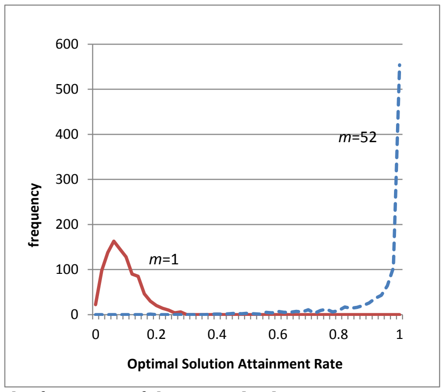

Figure 10. The frequency of the optimal solution attainment rate. (scale-free graph, *m* = 1 and *m* = 52)

All experiments involved performing 1000 problems 1000 times and calculating the average. However, the optimal solution attainment rate differs for every problem and cannot be determined until we solve it. The optimal solution attainment rate to problems of the same class is almost the same. Subsequently, we checked the frequency distribution of the optimal solution attainment rate in *m* = 1 for about 1000 problems used for the experiment and similarly checked the frequency distribution of the optimal solution attainment rate in *m* = 20. The frequency distribution is shown in Fig. 9. Thus, the problem made from the same parameter becomes the distribution of the bell curve in *m* = 1. If the degree of multiplex is increased, the frequency distribution will gather near 1.0.

The constraint graph of the problem used for these experiments was a random graph. For comparative experiments, we created 1000 problems on a scale free graph. The frequency distribution of the optimal solution attainment rate of those problems is shown in Fig. 10. The curve of the graph showed a tendency similar to Fig. 9. Thus, with the creation parameter of the problems, although the optimal solution attainment rate differed, each had become a bell curve. Therefore, in many cases, the optimal solution is obtained by setting up a suitable degree of multiplex, presuming an optimal solution attainment rate.

As shown in Fig. 6, within the experimental range, the optimal solution attainment rate of algorithm DSA used for this experiment was less than 0.25 and remained constant, even when the computation time was extended. However, optimal solution attainment rates of 0.99 were attained, as shown in Fig. 8, as a result of multiplexed execution of the algorithm. In other words, the optimal solution can also be efficiently obtained by multiplex execution for an approximation algorithm with a low optimal solution attainment rate. The effect of multiplex execution is considerable.

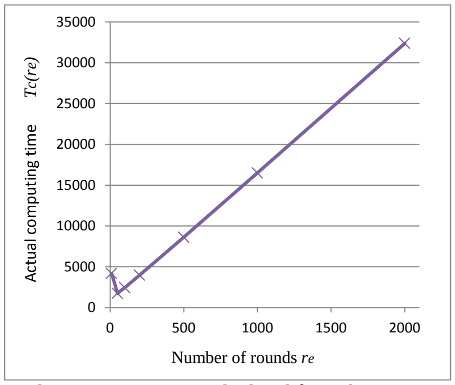

Figure 11. Real computation time calculated from the experimental result (Figs. 6, 7)

With regard to the theoretical value of multiplicity from the experimental result, the result of having applied the number of rounds and multiplicity is shown in Fig. 11, which is equivalent to the actual measurement of the theoretical-analysis result of Fig. 5. Both Fig. 11 and 5 include a local minimum.

5. Considerations

With regard to the optimal solution attainment rate at the time of multiplexed execution of the distributed approximation algorithm, a theoretical examination was performed in Section 3.3 and experimented in Section 4. Consequently, even when a distributed algorithm with a low optimal solution attainment rate was used, it was shown that an optimal solution attainment rate of 0.99 etc. could be realized by multiplexed execution.

When multiplex execution is carried out, the distribution of the distance from the optimal solution diminishes [1]. Therefore, even if the optimal solution is not obtained, the obtained solution is one in which the distance from the optimal solution is negligible.

The fundamental view of this multiplexing solution is as follows: Each plane is searched independently, the search is stopped independently, and an optimal solution is chosen, which resembles the concept of a genetic algorithm. Therefore, this method is considered applicable not only for DCOP but also for the distributed search algorithm.

6. Conclusion

In this study, when the multiplex execution of the distributed approximation algorithm for DCOP was carried out, it was shown that an optimal solution attainment rate could be set. Since a simple algorithm was applied for this experiment, further study of a method for the autonomous adjustment of multiplicity might improve the effectiveness of the solution attainment rate.

REFERENCES

[1] Y. Iizuka and K. Iizuka, “Exceeding the efficiency of distributed approximate algorithms enabling by the multiplexing method,” in Knowledge-Based and Intelligent Information and Engineering Systems, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, vol. 6883, pp. 366–377.

[2] M. Yokoo and K. Hirayama, “Algorithms for distributed constraint satisfaction: A review,” in Autonomous Agents and Multi-Agent Systems, vol. 3, No. 2, 2000, pp. 198–212.

[3] M. Calisti and N. Neagu, “N.: Constraint satisfaction techniques and software agents,” in In: Agents and Constraints Workshop at AIIA ’04, 2004.

[4] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, “Adopt: asynchronous distributed constraint optimization with quality guarantees,” Artif. Intell., No. 161, pp. 149–180, 2005.

[5] A. Petcu and B. Faltings, “A scalable method for multiagent constraint optimization,” in Proceedings of the International Joint Conference on Artificial Intelligence, 2005, pp. 266–271.

[6] R. Mailler and V. Lesser, “Solving distributed constraint optimization problems using cooperative mediation,” in in Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems. IEEE Computer Society, 2004, pp. 438–445.

[7] W. Yeoh, S. Koenig, and A. Felner, “Idb-adopt: A depth-first search dcop algorithm,” in Eighth International Workshop on Distributed Constraint Reasoning, in Twentieth International Joint Conference on Artificial Intelligence, 2007, pp. 56–70.

[8] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg, “Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks,” Artif. Intell., No. 161, pp. 55–87, 2005.

[9] R. Junges and A. L. C. Bazzan, “Evaluating the performance of dcop algorithms in a real world, dynamic problem,” in AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, 2008, pp. 599–606.

[10] S. Fitzpatrick and L. Meertens, “An experimental assessment of a stochastic, anytime, decentralized, soft colourer for sparse graphs,” in 1st Symposium on Stochastic Algorithms: Foundations and Applications, 2001, pp. 49–64.

[11] C. P. Gomes and B. Selman, “Algorithm portfolios,” Artif. Intell., vol. 126, pp. 43–62, 2001.

[12] I. D. Falco, R. D. Balio, E. Tarantino, and R. Vaccaro, “Improving search by incorporating evolution principles in parallel tabu search,” in 1994 IEEE Conference on Evolutionary Computation, 1994, pp. 823–828.

[13] C. P. Gomes, B. Selman, N. Crato, and H. A. Kautz, “Heavy-tailed phenomena in satisfiability and constraint satisfaction problems,” Journal of Automated Reasoning, vol. 24, no. 1/2, pp. 67–100, 2000.

[14] G.Ringwelski and Y.Hamadi, “Boosting distributed constraint satisfaction,” in CP-2005, 2005, pp.549–562.