International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol. 3, Issue. 5, Sep - Oct. 2013 pp-3181-3187 ISSN: 2249-6645

Effect Estimation Method of Parallel Computing Based on Dynamic Generation of Equivalent Transformation Rules

Hiroshi Mabuchi[1]
[1](Software and Information Science, Iwate Prefectural University, Japan)

ABSTRACT: *Recent studies in parallel computing have mainly focused on physically producing multiple computing sites for computational speed-up and not considered theories and methodologies, which are the essence of parallel computing, and correctness as well. It means that the studies have mostly considered cost-effectiveness, that is, how many computing sites are produced determines how much computing speed improves. This study proposes an algorithm to estimate the effectiveness of parallel computing based on the model with established theories and methodologies, which are the essence of parallel computing, and with assured correctness, instead of exploring such cost-effectiveness. Moreover, we will demonstrate the effectiveness of the proposed method by applying it to one of constraint satisfaction problems, Pic-a-Pix Puzzle, and comparing sequential computing time with estimated parallel computing time based on the dynamic generation of equivalent transformation (ET) rules.*

Keywords: *parallel Computing, Effect Estimation Algorithm, Equivalent Transformation Rule, Rule Dynamic Generation.*

I. INTRODUCTION

Recently, there has been an increase in study of parallel computing and in its importance [1,5,6,8]. Parallelization can be mainly divided in two categories: in local area (LAN) and in CPU [12,14]. The former speeds up the computation by using a supercomputer to make the master processor and multiple slave processors share the computing. The latter executes multithreaded applications on the multi-core processor.

Recent studies in parallel computing have mainly focused on physically producing multiple computing sites for computational speed-up and not considered theories and methodologies, which are the essence of parallel computing, and correctness as well [5,7,8]. It means that the studies have mostly considered cost-effectiveness, that is, how many computing sites are produced determines how much computing speed improves. The purpose of this study is to propose an algorithm to estimate the effectiveness of parallel computing based on the model with established theories and methodologies, which are the essence of parallel computing, and with assured correctness, instead of exploring such cost-effectiveness. Specifically, we adopt the parallel computing model (with multiple processors and CPUs) based on the dynamic generation of equivalent transformation (ET) rules, which is on the basis of the sequential computation model (with one processor) [2,3,9] based on the dynamic generation of ET rules [4]. This model executes costly generation of ET rules in dynamic and parallel way in multiple computing sites. ET is to preserve the meaning of the problem and to transform it to another simplified problem. An ET rule is a meaning-preserving transformation rule. It operates by, first, rewriting a definite clause by replacement of its body and, then, making clause specialization. Based on this model, it is possible that we conduct actual measurement of the effectiveness of parallel computing using with multiple slave processors and CPUs. However, this study provides the model with multiple abstract computing sites and independently operating virtual CPUs and proposes an algorithm to estimate the effectiveness of parallel computing without actual measurement.

Moreover, we will demonstrate the effectiveness of the proposed method by applying it to one of constraint satisfaction problems [10,11,13,15], Pic-a-Pix Puzzle [16], and comparing sequential computing time with estimated parallel computing time based on the dynamic generation of ET rules.

II. SEQUENTIAL COMPUTING MODEL BASED ON THE DYNAMIC GENERATION OF ET RULES

This study is based on a sequential computing model on the basis of the dynamic generation of ET rules, and the sequential computing model is used for a comparison in the experiment. We will define, therefore, the sequential computing model first in this section, then outline "Pic-a-Pix Puzzle [16]", which is used in the experiment of this study, and describe the generation of ET rules with the specific example.

2.1 Definition of The Sequential Computing Model

A sequential computing model is a model which successively simplifies problems in order to obtain solutions in the end by repeating the following process: if there is an applicable rule in a pre-constructed ET rule set (program), the rule will be applied to the problem; if not, a new rule will be generated dynamically and the rule will be applied to the problem. Fig. 1 shows the outline of this model [2,3].

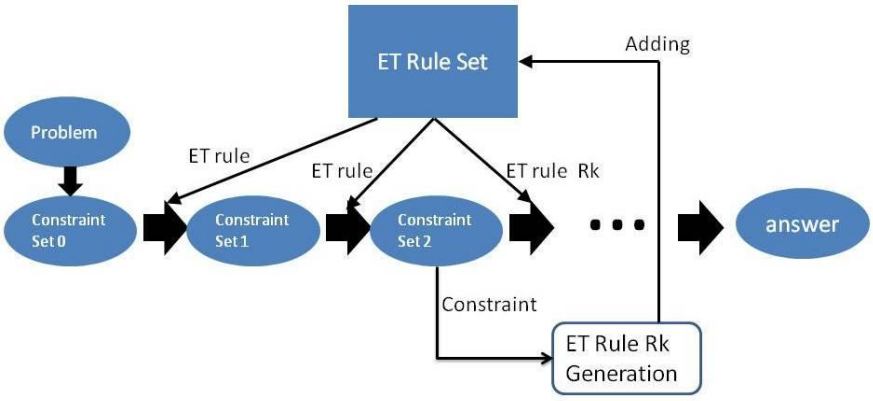

Fig. 1 Sequential Computing Model

ET is to preserve the meaning of the problem and to transform it to another simplified problem. The first step of sequential computing corresponds to the application of ET rules.
In Fig. 1, there is no applicable rule to the constraint set 2. A new ET rule, then, will be generated, added to the ET rule set and applied to the constraint set 2 so that equivalent transformation will proceed.

2.2 Pic-a-Pix Puzzle

Consider a Pic-a-Pix puzzle in Fig.2. It consists of a blank grid and clues, i.e., block patterns, on the left of every row and on the top of every column, with the goal of painting blocks in each row and column so that their length and order correspond to the patterns and there is at least one empty square between adjacent blocks.

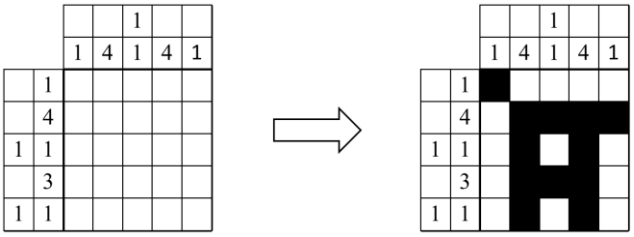

Fig. 2 A Pic-a-Pix Puzzle

For example, the constraints in Fig. 1 in Section 2.1 correspond to each row or column of the puzzle, and all the constraints are computed with the same algorithm. This will allow each constraint of the problem to be shared and computed in a number of computing sites.

2.3 ET Rule

An ET rule is a meaning-preserving transformation rule.
The following is a format of ET rules.
H , {Cs} → {Es} , Bs.

Here, H is an atom, and each of Cs, Es, and Bs is an atom string. An atom means an indivisible minimum sentence and is represented as a sequence of a predicate and zero or more of arguments. H , Cs, and Es is called head, condition part and execution part, respectively, and an atom which is an element of Bs is called replaced atom. Connecting atoms (string) with an arrow means replacing an atom (string) with another atom (string).

2.4 ET Rule Generation

This section describes a method of generating ET rules with an example of Pic-a-Pix Puzzle.

Dynamic generation of ET rules is that if there is no applicable rule during the process of ET by applying ET rules in the problem, a new rule will be dynamically generated. Adding the new rule to the program will allow the ET to be continued. It is shown in the part of Fig. 1, that is, generating ET rule Rk from "Constraint Set 2" and adding it to ET rule set.

In the case of Pic-a-Pix Puzzle, as shown in Fig. 2, rules are generated with the following algorithm depending on each sequence of numbers of the puzzle.

(1) From the list (row or column) consisting of the first argument, which is the number of blocks to be painted, and the second argument, which is n elements (block), the common part of the list is obtained, and if it is determined, a rule is generated.

(2) From the beginning of the list, sort out undetermined elements with each case, apply 1 or 0 to every case, specialize the common part depending on the number of blocks to be painted and determine either 1 or 0.

For the fourth row of the puzzle shown in Fig. 2, Fig. 3 shows a method of generating ET rules.

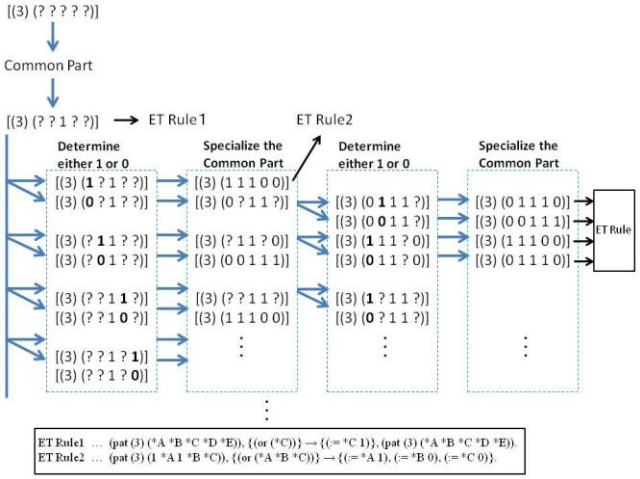

Fig. 3 An Example of Rule Generation

ET Rule1 and ET Rule2 shown in Fig. 3 are explained.

ET Rule1 is a rule which transforms the list consisting of "the number of painted block is three" and "five blank blocks" into a predicate which suggests painting the common part, the third (*C) block. ET Rule2 is a transformation rule in which the list consisting of "the number of painted block is three" and "the first and third blocks are painted" transforms into "the second block is painted {(:= *A 1)}" and "the fourth and fifth blocks become blank {(:= *B 0) , (:= *C 0)} in order to satisfy the condition to paint three consecutive blocks.

III. PARALLEL COMPUTING MODEL BASED ON THE DYNAMIC GENERATION OF ET RULES

A parallel computing model based on the dynamic generation of ET rules [4] consists of the Master computer and multiple computing sites (Slaves). Costly rule generation is done in each computing site, and the fastest generated ET rule is applied to the constraint sets to obtain a new more simplified constraint set. A solution will be obtained by repeating this operation. Fig. 4 shows the outline of this model.

The computation of this model follows from 1 through 4 below.

1. Generate a set of constraints (Constraint Set 0) to solve a constraint satisfaction problem.
2. Each atom of the constraint set is sent to multiple computing sites (Slaves) and costly ET rule generation is done in each Slave.
3. Apply the first generated ET rule of the ET rules generated in 2 to the constraint sets successively and obtain a new simplified constraint set.
4. If the solution of the problem is obtained, the computing is finished. If not, it goes back to 2.

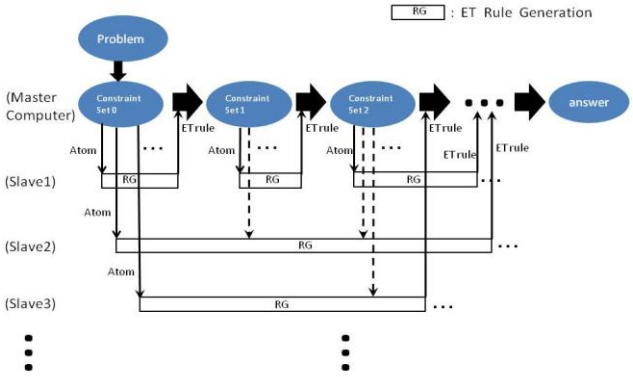

Fig.4 Parallel Computing Model

IV. EFFECT ESTIMATION OF PARALLEL COMPUTING BASED ON THE DYNAMIC GENERATION OF ET RULES

This section proposes a method to estimate to what extent the computing become more efficient in the case of parallelization.

4.1 Effect Estimation Method of Parallel Computing

An effect estimation method of parallel computing is a method in which operations will be continued repeatedly until no constraint of the problem (atom) exists, and when no atom exists, an estimated parallel computing time is output.

First, an ET rule is generated for each atom in constraint sets, then the fastest generated ET rule is applied to the constraint set, and a new constraint set is obtained. The applied ET rule is deleted.

An algorithm to estimate the effect of parallel computing is shown below.

Given a set C of atoms. Here, T is an estimated parallel computing time. S is a set of pairs of a generated rule and a time. rt is a time for rule r to be generated. t is the time taken until a new constraint set is obtained after the rule is applied to the constraints sets.

(1) T= 0, S = {}.
(2) while C =/= {} do
(2-1) For any atom a in C
gen(a) = (rt, r). // rule r is generated.
S = S ∪ {(T + rt +τ+ t, r)} // τ is a delay for sending and receiving.
(2-2) S' = sort(1, S). // sort by the first argument.
(2-3) Let (rt, r) be the first pair in S'.
(2-4) If T < rt, then T = rt.
(2-5) Apply the rule r to an atom in C to have new C. (The time at this point is t.)
(2-6) S = S - {(rt, r)}.
(3) return T

This algorithm is explained.
(1) Initialize an estimated parallel computing time and a set of pairs of a generated rule and a time.
(2) Repeat the operation from (2-1) to (2-6) until no atom in the problem exists.
(2-1) For each atom, create a pair of an ET rule and its generating time. Then, create a set of pairs of T, rt, τ, t and an ET rule.
(2-2) Sort the set created in (2-1) by ascending order of total time of T, rt, τ and t.
(2-3) Take out the first sorted pair of S'.
(2-4) If rt is more than T, let rt be T.
(2-5) Apply the rule taken out in (2-3) to the atom and obtain a new atom.
(2-6) Delete the applied pair of the ET rule and generating time.
(3) When there is no atom in the problem, return the estimated parallel computing time as a solution.

4.2 Estimation of Parallel Computing Time in Pic-a-Pix Puzzle

Fig. 5 shows the estimated parallel computing time based on dynamic generation of ET rules which was obtained using a 7 x 7 Pic-a-Pix Puzzle (let τ, which is defined in Section 4.1, be 0msec).

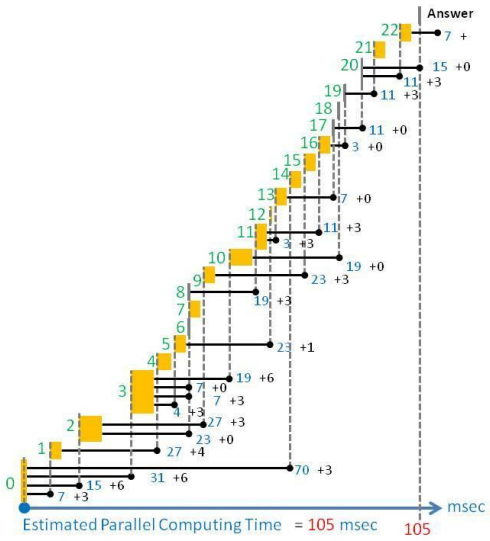

Fig.5 Calculation of Estimated Parallel Computing Time
Fig. 5 is explained based on Section 4.1.

The numbers from 0 to 22 represent the constraint set number. For example, four ET rules are generated in the constraint set 0. The fastest generated rule (in this case, it is the bottom of four rules in Fig. 5, which is generated in 7msec) is applied an atom (constraint), and a new atom is generated. The time to generate this new atom is 3msec. That is, it takes 10msec (= 7+3) to the obtainment of a new atom from the generation of the bottom rule and the application of it to an atom.

Next, the fastest generated rule in the sum of the rules of the unused rule set generated in the constraint set 0 and the rule set generated in the constraint set 1 (in this example, the second rule from the bottom generated in the constraint set 0) is

applied to an atom, and a new atom is obtained. In this case, it takes 15msec to generate the rule, and 6msec from the application of the rule to an atom to the obtainment of a new atom.

Next, the fastest generated rule in the sum of the rules of the unused rule set generated in the constraint set 0, that in the constraint set 1 and the rule set generated in the constraint set 2 (in this example, the third rule from the bottom generated in the constraint set 0) is applied to an atom, and a new atom is obtained. In this case, it takes 31msec to generate the rule, and 6msec from the application of the rule to an atom to the obtainment of a new atom. Repeat the same process and obtain the Answer by applying the generated rule in constraint set 20(the rule above the two rules generated in the constraints set 20).

The followings show "Constraint set before the rule is applied → The rule which is applied → Constraint set after the rule is applied ··· Estimated time" from the constraint set 0 to the Answer.

Constraint Set 0 → Apply the second rule from the top generated in the Constraint Set 0
→ Constraint Set 3 ··· 37msec (= 31 + 6)
Constraint Set 3 → Apply the second rule from the top generated in the Constraint Set 3
→ Constraint Set 8 ··· 7msec (= 7 + 0)
Constraint Set 8 → Apply the rule generated in the Constraint Set 8
→ Constraint Set 11 ··· 22msec (= 19 + 3)
Constraint Set 11 → Apply the rule under the rules generated in the Constraint Set 11
→ Constraint Set 13 ··· 6msec (= 3 + 3)
Constraint Set 13 → Apply the rule generated in the Constraint Set 13
→ Constraint Set 17 ··· 7msec (= 7 + 0)
Constraint Set 17 → Apply the rule generated in the Constraint Set 17
→ Constraint Set 20 ··· 11msec (= 11 + 0)
Constraint Set 20 → Apply the above the rules generated in the Constraint Set 20
→ Answer ··· 15msec (= 15 + 0)

The total of these estimated parallel computing times is 105msec(= 37 + 7 + 22 + 6 + 7 + 11 + 15). The sequential computing time for this example is 1920msec. Then, if the sequential computing time is set to 1, the estimated parallel computing time becomes 0.05. The computing time, thus, is found to be greatly reduced.

V. CALCULATION OF THE PARALLEL ESTIMATED COMPUTING TIME TAKING INTO ACCOUNT THE COMMUNICATION TIME

In this section, it is studied how the estimated parallel computing time changes if the communication time is taken into account. That is, τ defined in section 4.1 is considered.

5.1 Calculation of The Estimated Time Taking into The Fixed Communication Time

In this section, set the communication time to 10msec in the example in section 4.2 and calculate the estimated parallel computing time. The result is shown in Fig. 6. The process flow of the parallel computing time and the estimated time change by adding the communication time of 10msec for the calculation made in Fig. 5. For instance, in Fig. 6, four rules are generated in the constraint set 0. The bottom rule, which is generated in the fastest time, 7msec, is applied to an atom, and a new atom is obtained. The time for this is 3msec and the communication time is 10msec. The process flow to obtain the estimated parallel computing time until the solution is obtained is the same as that in Fig. 5.

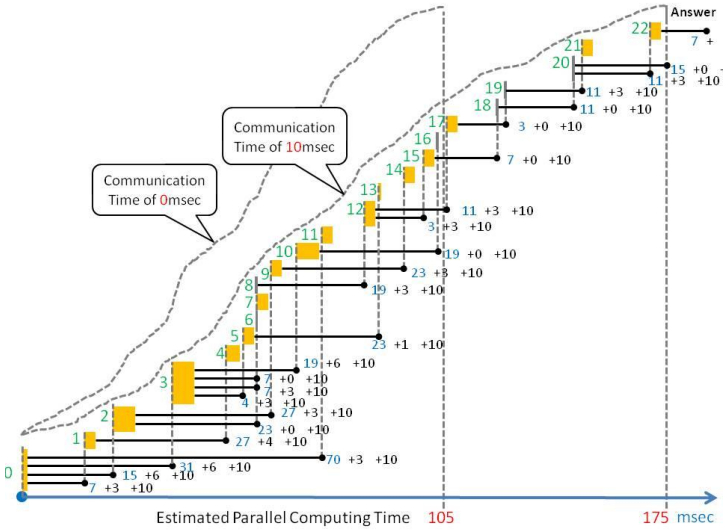

Fig.6 Calculation of Estimated Parallel Computing Time (Taking into Account Communication Time of 10msec)

Fig. 6 shows that in the case of taking into account the communication time of 10msec, the estimated parallel computing time is 175msec. When the sequential computing time is set to 1, the estimated parallel computing time becomes 0.09, and the computing time is greatly reduced.

5.2 The Change of The Estimated Time Due to The Change of The Communication Time

In this section, it is studied how the estimated time changes when the communication time is changed from 0msec to 200msec. The result is shown in Fig. 7. This graph shows that the communication time of which the range is from 0msec to 30msec greatly affects the computing time, however, the communication time over 30msec does not much affect it.

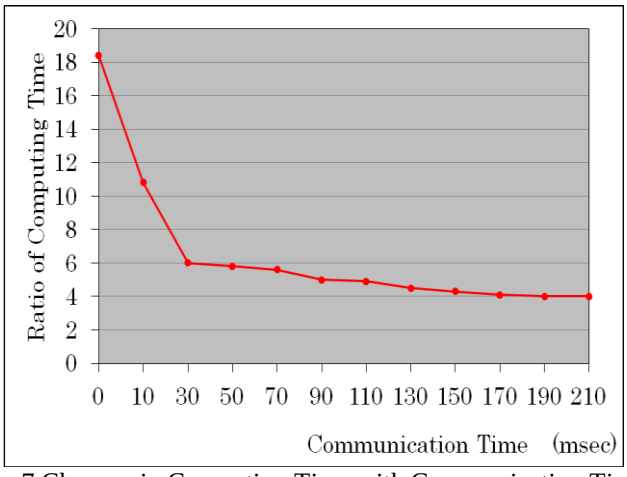

Fig.7 Changes in Computing Time with Communication Time

VI. CALCULATION OF THE ESTIMATED PARALLEL COMPUTING TIME BY THE PROGRAM

This section compares the estimated computing time with the sequential computing time when a Pic-a-Pix Puzzle is solved with the program, which is created using an ET programming language [17] to obtain the estimated parallel computing time.

As experimental data, we chose five 15 x 15-size Pic-a-Pix Puzzles from http://starscafe.net/riddle/logic/ [16] and compared the average sequential computing time of ten times with the average estimated parallel computing time of ten times. The result is shown in Table 1. Here, Table 1 shows the ratio of the estimated parallel computing time when the sequential computing time is set to 1.

Problem	Sequential Computing Time	Estimated Parallel Computing Time
Problem 1	1	0.72
Problem 2	1	0.2
Problem 3	1	0.05
Problem 4	1	0.04
Problem 5	1	0.01

Table 1 Comparison of Computing Time

It can be found from Table 1 that the parallel computing time is greatly reduced compared with the sequential computing time. It is assumed that this is because ineffective rules which make computational efficiency lower were not applied to the problem when the problem was solved with the parallel computation. Available rules are applied regardless of anything else in sequential computing, whereas such ineffective rules can be eliminated in parallel computing.

VII. CONCLUSIONS

We proposed an algorithm to estimate the effect of parallel computing using a parallel computing model based on the dynamic generation of ET rules. This study provided a model with multiple abstract computing sites and independently operating virtual CPUs and estimated the effect of parallel computing on the model. Furthermore, we have demonstrated the effectiveness of the proposed method by creating a program which estimates the effect of parallel computing, solving several Pic-a-Pix Puzzles with the program and comparing the sequential computing time with the parallel computing time. This study used Pic-a-Pix Puzzles as examples of constraint satisfaction problems. We would like to demonstrate the effectiveness of the proposed method by applying it to various examples in the future.

REFERENCES

[1] T. Abe, T. Hiraishi, Y. Miyake, T. Iwashita. Job-level Parallel Executions for Satisfying Distributed Constraints. *Information Processing Society of Japan, Vol.59*, 2011, 1- 8.
[2] K. Akama and E. Nantajeewarawat. Formalization of the equivalent transformation computation model. *Journal of Advanced Computational Intelligence and Intelligent Informatics, 10*, 2006, 245-259.
[3] K. Akama, E. Nantajeewarawat, and H. Ogasawara. A Processing Model Based on Equivalent Transformation and a sufficient Condition for Program Correctness. *International Journal of Foundations of Computer Science 3*, 2010.
[4] K. Akama, E. Nantajeewarawat, H. Koike. Constructing Parallel Programs Based on Rule Generators. *The First International Conference on Advanced Communications and Computation*, 2011, 173-178.
[5] Guy E. Blelloch. Programming parallel algorithms. *Magazine Communications of the ACM CACM Homepage archive, Vol. 39, Issue 3*, 1996.
[6] D.C.Gras and M.Hermenegildo. Non-strict independence-based program parallelization using sharing and freeness information. *Theoretical Computer Science, 410*, 2009, 4704-4723.
[7] G.Gupta, E.Pontelli, K.Ali, M.Carlsson, and M.Hermenegildo. Parallel execution of Prolog programs: a survey. *ACM Transactions on Programming Languages and Systems, 23*, 2001, 472-602.
[8] W. Daniel Hillis, Guy L. Steele, Jr. Data parallel algorithms. *Magazine Communications of the ACM - Special issue on parallelism CACM Homepage archive, Vol. 29, Issue 12*, 1986.
[9] H. Mabuchi, K. Akama, and T. Wakatsuki. Equivalent transformation rules as components of programs. *International Journal of Innovative Computing, Information and Control, 3*, 2007, 685-696.
[10] Y. Makoto, K. Hirayama. Distributed Breakout: Iterative Improvement Algorithm for Solving Distributed Constraint Satisfaction Problem (Special Issue on Parallel Processing). *Information Processing Society of Japan, Vol.39, No.6*, 1998.
[11] . Makoto, E.H.Durfee, T.Ishida, K.Kuwabara. The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering, Vol. 10*, 1998, 673-685.
[12] K. Minami. Performance Improvement of Application on the "K computer". *IEICE Transactions, 95(2)*, 2012, 125-130.
[13] S. Nishihara. Fundamentals and Perspectives of Constraint Satisfaction Problems. *Japanese Society for Artificial Intelligence, Vol.12, No.3*, 1997, 351-358.
[14] N. Nishimura. Grid Computing and Parallel Computing. *Journal of The Japan Society for Computational Engineering and Science, 10(3)*, 2005, 1198-1202.
[15] K. Uchino, S. Kubota, H. Konoh, S. Nishihara. Classifying CSPs On the Basis of Parallelism. *Information Processing Society of Japan, Vol.35*, 1994, 2676-2684.
[16] http://starscafe.net/riddle/logic/
[17] http://assam.cims.hokudai.ac.jp/eti/index.html