Vol. 45 No. 9 IPSJ Journal Sep. 2004

Regular Paper

An Easy-Hard-Easy Cost Profile in Distributed Constraint Satisfaction[☆]

Katsutoshi Hirayama,[†] Makoto Yokoo[††] and Katia Sycara[†††]

We first present an algorithm called *multi-ABT* as a baseline algorithm for solving distributed constraint satisfaction problems where each agent has multiple local variables. Then, we show a cost profile of multi-ABT for various numbers of *intra-agent constraints* (constraints defined over variables of one agent) and *inter-agent constraints* (constraints defined over variables of multiple agents) in a distributed graph-coloring problem. This cost profile enabled us to make the following observations: (1) the satisfiability thresholds are identified in the narrow region on the x-y plane (where the x-axis is the number of intra-agent constraints and the y-axis is the number of inter-agent constraints) in which the sum of intra- and inter-agent constraints is constant, and problem instances in the region (called the *crossover belt*) are likely to be expensive in terms of the median cost; (2) among problem instances on the crossover belt, those with a smaller number of intra-agent constraints and a larger number of inter-agent constraints may be more expensive; and (3) for a fixed total number of variables, problem instances on the crossover belt may be more expensive as the number of agents increases or the number of variables per agent decreases. Our further experiments suggest that these observations can be generalized to cases where different algorithms are applied or different sets of parameters of the problem are used.

1. Introduction

Many problems in artificial intelligence can be formalized as constraint satisfaction problems (CSPs). A CSP instance consists of variables with finite and discrete domains and constraints over subsets of variables. A solution to a CSP instance is a set of values for the variables that satisfies all of the constraints.

CSPs are typical NP-complete problems, and thus it has been said that no algorithm has a worst-case search cost that is not exponential in relation to the size of the problem. However, recent studies have observed that for certain CSPs, such as the propositional satisfiability problem, the graph-coloring problem, and the binary CSP, an order parameter exists on which the median computational costs of algorithms show an *easy-hard-easy* cost profile[2),11),15)]. The search cost grows very rapidly when the problem size increases in the hard region on the parameter, but relatively slowly in the other regions on it[3)]. In other words, really computationally expensive problem instances are concentrated on some specific region of the parameter.

For example, in the graph-coloring problem, an instance of which consists of a graph, a specified number of available colors, and the requirement to find a color for each node of the graph such that no adjacent nodes have the same color, such an order parameter is the ratio of the number of links to the number of nodes. For randomly generated instances of the graph 3-coloring problem, it has been shown that the peak of the median computational cost is located at 2.3, i.e., where the number of links is 2.3 times as many as the number of nodes[8)].

Distributed CSPs (DisCSPs)[17),18)] are CSPs where variables and constraints are distributed among multiple agents, each of which tries to solve its own problem. Various application problems in multi-agent systems that are concerned with finding a consistent combination of agent actions (e.g., the distributed resource allocation problem[4)], the distributed scheduling problem[16)], the distributed interpretation task[10)], and the multi-agent truth maintenance task[9)]) can be formalized as DisCSPs.

Recently, several researchers have devel-

[☆] Notice for the use of this material The copyright of this material is retained by the Information Processing Society of Japan (IPSJ). This material is published on this web site with the agreement of the author (s) and the IPSJ. Please be complied with Copyright Law of Japan and the Code of Ethics of the IPSJ if any users wish to reproduce, make derivative work, distribute or make available to the public any part or whole thereof. All Rights Reserved, Copyright (C) Information Processing Society of Japan. Comments are welcome. Mail to address editj@ipsj.or.jp, please.

[†] Faculty of Maritime Sciences, Kobe University
[††] Graduate School of Information Science and Electrical Engineering, Kyushu University
[†††] The Robotics Institute, Carnegie Mellon University

oped distributed constraint satisfaction algorithms[1),7),14),18)~20)]. In a distributed constraint satisfaction algorithm, since an agent usually has *intra-agent constraints* (constraints defined over variables of one agent) and *inter-agent constraints* (constraints defined over variables of multiple agents), an agent must not only perform local computation to satisfy its intra- and inter-agent constraints but must also communicate with other agents to satisfy its inter-agent constraints. We can therefore assume that the cost of a distributed constraint satisfaction algorithm can vary with the numbers of intra- and inter-agents constraints. Our goal is to understand how the numbers of intra- and inter-agent constraints affect the cost of a standard distributed constraint satisfaction algorithm, since such information may provide a hint for formulating some problems as DisCSPs and designing more efficient distributed constraint satisfaction algorithms.

In this paper, we first present an algorithm called *multi-ABT*, which is an extension of the *asynchronous backtracking algorithm* (ABT)[18)], as a baseline algorithm for solving DisCSPs where each agent has multiple local variables. Then, we show a cost profile of multi-ABT for various numbers of intra- and inter-agent constraints in the distributed graph-coloring problem. As we will show in this paper, the cost profile exhibits an interesting pattern on the x-y plane (where the x-axis is the number of intra-agent constraints and the y-axis is the number of inter-agent constraints), and the pattern is closely related to the satisfiability thresholds where the ratios of solvable problem instances rapidly change from one to zero. Since the pattern seems independent of algorithms and instance generation methods, we expect that our results could serve as the basis of further theoretical or experimental analyses.

This paper is organized as follows. In Section 2, we provide a general definition of the DisCSPs and give an instance of the distributed graph-coloring problem. In Section 3, we present multi-ABT as a baseline algorithm for solving DisCSPs where each agent has multiple local variables. In Section 4, we describe the details of our experimental settings and results. Finally, in Section 5, we conclude our discussion and outline our plans for future work.

2. Distributed Constraint Satisfaction Problem

A CSP instance consists of variables with finite and discrete domains and constraints over subsets of the variables. A constraint can be represented as a predicate that is defined over some variables' domains and becomes true when the constraint is satisfied and false otherwise. A solution to a CSP instance is a set of values for the variables that satisfies all of the constraints.

The DisCSPs is the CSP where variables and constraints are distributed among multiple agents. A DisCSP instance consists of the following:

- a set of agents $1, 2, \ldots, k$
- a set of CSP instances $P_1, P_2, \ldots, P_k$ such that P_i belongs to agent i $(i = 1, \ldots, k)$

A solution to a DisCSP instance is the state where all of the agents find solutions to their own CSP instances. We usually assume that each agent has a CSP instance that includes inter-agent constraints. An *inter-agent constraint* is a constraint that is defined over variables of multiple agents. On the other hand, an *intra-agent constraint* is a constraint that is defined over variables of one agent. Consideration of inter-agent constraints complicates the search process over DisCSPs, because communication is required among agents to satisfy those constraints.

Figure 1 illustrates an instance of the distributed graph-coloring problem. In this problem, nodes are partitioned among agents and a link is assigned to the agent(s) having a node that is involved in the link. It is known that the scheduling problem in a distributed sensor network can be formulated as the distributed graph-coloring problem[12),21)]. In Figure 1, each of the three agents A, B, and C has nodes in the corresponding ellipse and links that are connected to the nodes. For example, agent A has the nodes n_1 and n_2 and the links l_{12}, l_{13}, and l_{26}. In other words, agent A has a CSP instance with the variables n_1 and n_2 and the constraints not_equal(n_1, n_2), not_equal(n_1, n_3), and not_equal(n_2, n_6). Note that there are 7 links (constraints) in this figure, of which three, l_{13}, l_{34}, and l_{26}, are inter-agent constraints, and the other four, l_{12}, l_{45}, l_{46}, and l_{56}, are intra-agent constraints.

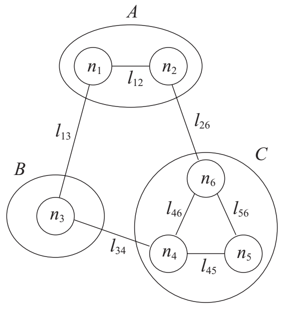

Fig. 1 Instance of the distributed graph coloring problem.

3. Multi-ABT

The *asynchronous backtracking algorithm* (ABT) belongs to the first generation of distributed constraint satisfaction algorithms, and is basically designed for solving DisCSPs where each agent has exactly one local variable[18]. Although several researchers have been trying to extend ABT in various directions[1),14),18)], not much effort has been made to extend ABT in such a way that it can explicitly handle DisCSPs where each agent has multiple local variables. Therefore, as a baseline algorithm for such problems, we present multi-ABT in this paper. This algorithm is basically similar to ABT, but differs in that it can explicitly handle DisCSPs where each agent has multiple local variables. Figure 2 shows the main part of this algorithm (the procedure to be followed when an agent receives an *ok?* message). The outline of this algorithm is as follows. Note that we are using graph-coloring terminology.

- A priority order among agents is introduced such that an agent with more links has higher priority (ties are broken in favor of an agent with a smaller identifier). Also, for each agent, a priority order among (internal) nodes is introduced such that a node with more connected links has higher priority (ties are broken in favor of a node with a smaller identifier).
- Each agent starts the procedure by randomly assigning a color to each of its nodes and sending these colors to *lower-priority neighbors* (a set of lower-priority agents that have nodes included in this agent's inter-agent constraints) via *ok?* messages.
- When an agent receives an *ok?* message, it updates an *agent_view*, which records the latest colors for nodes of *higher-priority neighbors* (a set of higher-priority agents that have nodes included in this agent's inter-agent constraints), and then executes the following procedure for each of its nodes, say n, in order of node priority.
  - Select a color for n that is consistent with the agent_view and the colors of this agent's higher-priority nodes, and then move to the next node. If the current color for n is already consistent, do not change that color.
  - If there is no such consistent color for n, generate a *nogood* by using the method described in Hirayama and Yokoo[6]. A nogood is a set of colors for some nodes under which there is no consistent color for n, and it can be considered a new constraint discovered during algorithm execution. Since no superset of a nogood can be a solution, an empty nogood eliminates all possible colors for nodes. Therefore, an agent can detect the fact that there is no solution if it generates an empty nogood.
  - If any of this agent's nodes are contained in the generated nogood, record the nogood as a new constraint, backjump to the lowest-priority node among those nodes, and reselect colors for the node and its subsequent nodes. On the other hand, if none of this agent's nodes is contained in the generated nogood, keep the current color for n, push the nogood into a *stack*, and move to the next node.

  When an agent finishes selecting colors for all of its nodes, it sends each nogood in the stack via a *nogood* message to the lowest-priority agent among those having nodes in the nogood, and then sends the changed colors to lower-priority neighbors via *ok?* messages.
- When an agent receives a *nogood* message, it records the content of that message as a new constraint and examines all of its nodes in the same way as described above. Note that when a received nogood includes an unknown node that belongs to a non-neighboring agent, an agent requests the non-neighboring agent to keep informed about a color for the node.

Since agents record all nogoods discovered in their concurrent search processes, this algorithm is complete, that is, it can reach a solu-

```
when received (ok?, (sender, node, color)) do
  add (sender, node, color) to agent_view;
  L :=an ordered list of all nodes within this agent (descending
      order of priority);
  N := null;
  if check_nodes(L, N) then
    wait for a next message;
  else
    broadcast "no solution exists";
end do;

procedure check_nodes(L, N)
  if L is empty then
    send each nogood in N to the agent whose priority is the lowest
    in the nogood;
    send all changed colors to neighbors;
    return true;
  else
    n :=the first node in L;
    if n's color is consistent (with the colors of higher-priority
        agents' nodes and the colors of this agent's higher-priority
        nodes) then
      L := L with n deleted; check_nodes(L, N);
    else
      if there is another consistent color, c, for n then
        change n's color to c;
        L := L with n deleted; check_nodes(L, N);
      else
        nogood :=a set of nodes' colors causing this conflict;
        if nogood is empty then
          return false;
        else
          if nogood includes nodes of this agent then
            m :=the lowest-priority node among these nodes;
            L :=a list of m and all lower-priority nodes than m;
            record nogood as a new constraint;
            check_nodes(L, N);
          else
            add nogood to N;
            L := L with n deleted; check_nodes(L, N);
```

Fig. 2 Multi-ABT.

tion if a problem instance has at least one solution, or it can establish that no solution exists.

4. Experiments

Our next question is how multi-ABT behaves on the space constructed from the numbers of intra- and inter-agent constraints of the distributed graph-coloring problem.

We implemented multi-ABT on a simulator of a *synchronous distributed system*. A synchronous distributed system is a model of a distributed system, in which every agent synchronously performs the following cycle: (1) receive all of the messages that were sent to the agent in the previous cycle and (2) perform local computation to change its internal state and determine the contents of messages and then send those messages to other agents. Although multi-ABT can work on any type of distributed system, we assumed a synchronous distributed system in this study, since it is one of the simplest distributed systems and we wanted to make the underlying distributed system simple in order to focus on the characteristics of the algorithm. We would like to point out that a similar simulator has traditionally been used to evaluate the performance of distributed constraint satisfaction algorithms[17),18)].

Using this simulator, we measured *cycles* and *maxccks* as the costs of an algorithm. The value of *cycles* represents the number of iterations in which the agents concurrently perform local computation until an algorithm is terminated. This measure has been conventionally used to estimate the cost of distributed constraint satisfaction algorithms[17),18)]. We should, however, notice that the number of cycles is completely unrelated to how much computational effort an individual agent makes in one cycle. Such computational effort must be considered when an agent has multiple local variables along with some intra- and inter-agent constraints, since it would impose a non-negligible cost on an algorithm. Therefore, we introduce another measure, *maxccks*, representing the total sum of the maximum number of consistency checks over the agents in each cycle until the algorithm is terminated. More specifically, the value of maxccks is measured as follows: in each cycle we identify a bottleneck agent that performed the most consistency checks among the agents and sum up all of those maximum numbers of consistency checks over the running cycles.

Several parameters are used to specify an instance of the distributed graph-coloring problem, such as the number of agents, the number of nodes, and the average number of nodes per agent. Since our goal is to clarify the relation between the cost of algorithms and the numbers of intra- and inter-agent constraints, we varied the numbers of intra- and inter-agent constraints while keeping the other parameters constant. In our experiments, we randomly generated an instance of the distributed graph-coloring problem using the method in Figure 3, where the inputs are the number of nodes, the number of available colors for a node, the number of agents, and the numbers of intra- and inter-agent constraints.

We first set the number of nodes to 100 and the number of available colors for a node to 3. Then, we randomly generated instances for two cases—2 agents with 50 nodes each and 5 agents with 20 nodes each—while varying the numbers of intra- and inter-agent constraints.

```
INPUTS: # of nodes ($n$), # of available colors for a node ($c$), #
    of agents ($k$), # of intra-agent constraints ($x$), # of inter-agent
    constraints ($y$);
OUTPUT: an instance of the distributed graph-coloring problem;
Step 1 Distribute $n$ nodes each with $c$ available colors among $k$
    agents such that these agents have numbers of nodes that are
    as nearly equal as possible;
Step 2 Define each of $x$ intra-agent constraints as follows:
    Step 2.1 Randomly select one agent from $k$ agents;
    Step 2.2 Randomly select two nodes from those belonging to
        the selected agent to define a link between them. If there
        already exists a link between the nodes, repeat the selection
        until two nodes with no link are found;
Step 3 Define each of $y$ inter-agent constraints as follows:
    Step 3.1 Randomly select two agents from $k$ agents;
    Step 3.2 Randomly select one node each from those belonging
        to the above two agents to define a link between these two
        nodes. If there already exists a link between the nodes,
        repeat the selection until two nodes with no link are found;
Step 4 Accept the graph if it is connected. If it is not connected,
    delete all of the links and go to Step 2;
```

Fig. 3 Method for generating instances.

Figure 4 shows the ratios of solvable instances for 5 agents with 20 nodes each on the x-y plane. Since the result we obtained for 2 agents with 50 nodes each was very similar, the figure is omitted for reasons of space. For each case, we generated 500 instances at every combination of the number of intra-agent constraints, $intra \in \{80, 90, \ldots, 180\}$, and the number of inter-agent constraints, $inter \in \{100, 110, \ldots, 160\}$, and plotted the ratio of solvable instances among 500 instances at each data point. The figure clearly shows that when the numbers of intra- and inter-agent constraints are increased, we can observe a rapid drop from almost one to zero in the ratio of solvable instances. For each case, the region where such a rapid drop occurs lies around $210 \leq intra + inter \leq 240$ on the x-y plane. We refer to such a region as a *crossover belt.*

Figures 5 and 6 show the median cycles and the median maxccks for 2 agents with 50 nodes each, respectively, and Figures 7 and 8 show those for 5 agents with 20 nodes each, respectively. We can observe the following from these results.

Observation 1: *Both the median cycles and the median maxccks on the crossover belt are higher than those in the other regions. For each of these figures, we can see that there is a "ridge" along the crossover belt and that the top of the ridge lies around* $230 \leq intra + inter \leq 240$ *on the x-y plane, which corresponds to the region where the ratios of solvable instances are roughly between 0.3 and 0.*

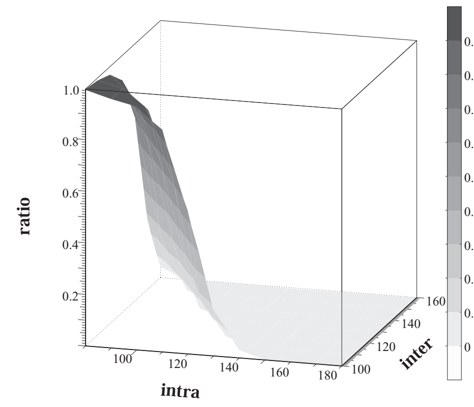

Fig. 4 Ratio of solvable instances (5 agents with 20 nodes each, random 3col).

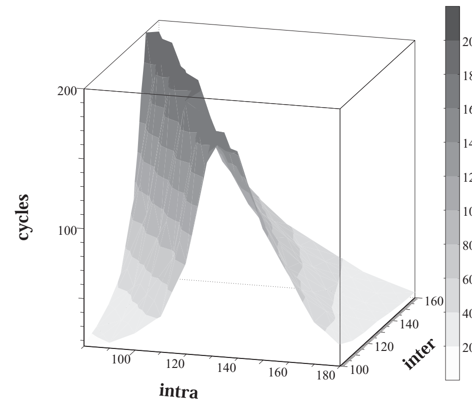

Fig. 5 Median cycles (2 agents with 50 nodes each, random 3col, with multi-ABT).

According to Hogg and Williams[8], for randomly generated instances of the (centralized) graph 3-coloring problem, the median computation cost of a depth-first backtracking search with the Brelaz heuristic rises to a peak when the ratio of the number of links to the number of nodes is 2.3, and the location of the peak coincides with the point at which the ratio of solvable instances is 0.5. In our experiments, although we used a distributed constraint satisfaction algorithm, we can observe a basically similar result, where the top of the ridge of the median cost appears when the ratio of the total number of links ($intra + inter$) to the number of nodes (100) is about 2.3 or 2.4. Its location, however, does not coincide with the region where the ratio of solvable instances is 0.5.

Observation 2: *When the number of intra-agent constraints decreases and the number of inter-agent constraints increases, the top of the*

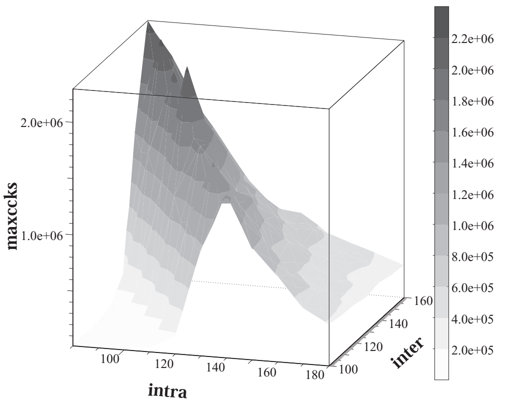

Fig. 6 Median maxccks (2 agents with 50 nodes each, random 3col, with multi-ABT).

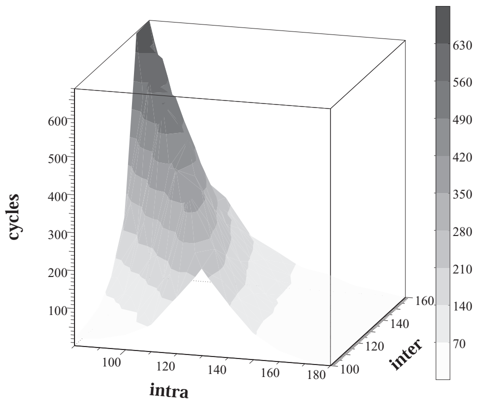

Fig. 7 Median cycles (5 agents with 20 nodes each, random 3col, with multi-ABT).

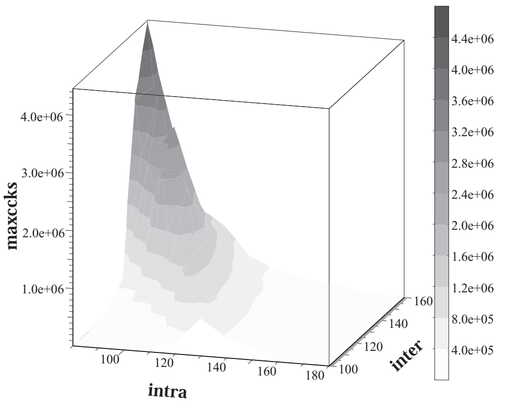

Fig. 8 Median maxccks (5 agents with 20 nodes each, random 3col, with multi-ABT).

ridge becomes higher.

This implies that the instances on the crossover belt are not uniformly hard. Among the instances on the crossover belt, those with a small number of intra-agent constraints and a large number of inter-agent constraints tend to be expensive both in terms of cycles and maxccks.

The increase in cycles in that direction could be explained as follows. When solving an instance with a small number of intra-agent constraints and a large number of inter-agent constraints, an agent is likely to select misleading colors for its nodes, that is, colors that appear to be correct from its local point of view but actually are incorrect from the global point of view. Once an agent (especially a higher-priority agent) selects such misleading colors, other agents have to perform a considerable amount of search to make the agent change the misleading colors, and thus the number of cycles grows in such an instance.

On the other hand, the increase in maxccks could be explained as the result of two competing effects: a decrease in the computational cost of a cycle and an increase in the total number of cycles. For a problem instance with a small number of intra-agent constraints and a large number of inter-agent constraints, the computational cost of a cycle decreases, because an agent tends to perform nogood sending rather than internal backjumping. On the other hand, as mentioned above, we see that the total number of cycles can increase in such an instance. These two phenomena might have conflicting effects on maxccks, but our experimental result shows that the latter overrides the former, leading to an increase in maxccks in that direction.

This result might offer some insight into how to describe a problem as a DisCSP instance. If possible, we should make local problems tight (by increasing intra-agent constraints) and global problems loose (by decreasing inter-agent constraints) in order to reduce the costs of algorithms.

Observation 3: *The top of the ridge has a steeper slope for 5 agents with 20 nodes each than for 2 agents with 50 nodes each. As a result, the former case has a higher peak.*

For example, for 5 agents with 20 nodes each, the maximum and minimum cycles of the top of the ridge are 680 (at $intra = 80$ and $inter = 160$) and 232 (at $intra = 130$ and $inter = 100$), respectively, and for 2 agents with 50 nodes each they are 200 (at $intra = 80$ and $inter = 150$) and 170 (at $intra = 130$ and $inter = 100$), respectively. This result implies that the misleading colors mentioned above can have a great impact on the entire performance if the number of agents increases or the number

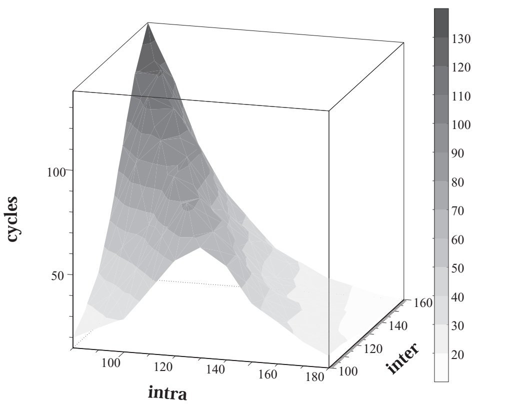

Fig. 9 Median cycles (5 agents with 20 nodes each, solvable 3col, with multi-ABT).

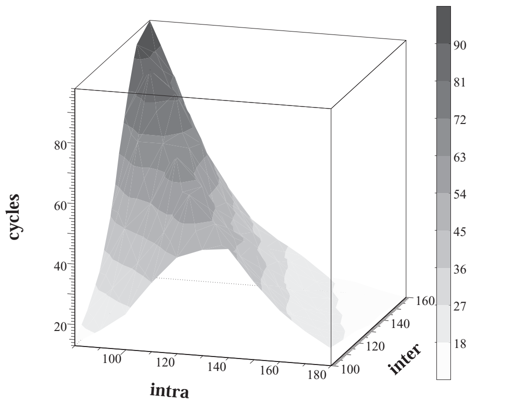

Fig. 10 Median cycles (5 agents with 20 nodes each, solvable 3col, with multi-AWC).

of nodes per agent decreases.

This result might also offer some insight into how to describe a problem as a DisCSP instance. If possible, we should make local problems large (or equivalently reduce the number of agents) in order to reduce the costs of algorithms.

In order to test the generality of our findings, we conducted more experiments using other combinations of problem settings and algorithms. First, we conducted experiments for solvable instances of the distributed 3-coloring problem when using multi-ABT and multi-AWC[20]. The results of these experiments are shown in Figures 9 and 10 (median cycles of 500 instances for 5 agents with 20 nodes each). A solvable instance was generated using a method similar to that in Figure 3, but modified to be able to specify a solution in advance and avoid any link that would eliminate the solution. For each case we can observe a ridge in almost the same region as in Figures 5–8, and the qualitative characteristics of the cost profile were consistent with our findings. Next, we conducted another experiment for random instances of the distributed 4-coloring problem when using multi-ABT. The method used to generate an instance was the same as the one in Figure 3, except that we set the number of available colors for a node to 4. A cost profile (median cycles of 500 instances for 5 agents with 10 nodes each) is shown in Figure 11. We again found a qualitatively similar pattern, where the top of the ridge was around $200 \leq intra+inter \leq 210$ and became higher when the number of intra-agent constraints decreased and the number of inter-agent constraints increased.

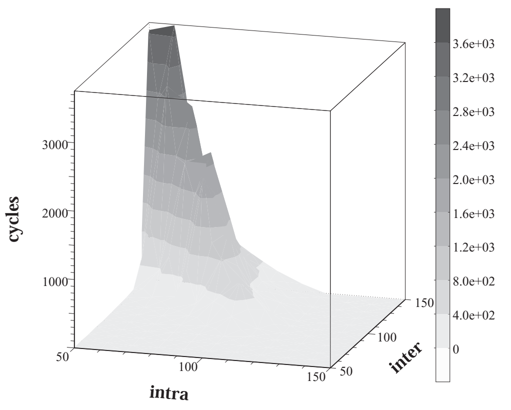

Fig. 11 Median cycles (5 agents with 10 nodes each, random 4col, with multi-ABT).

5. Conclusions and Future Work

In this paper, we have presented multi-ABT as a baseline algorithm for solving distributed constraint satisfaction problems where each agent has multiple local variables, and shown a cost profile of multi-ABT for various numbers of intra- and inter-agent constraints in the distributed graph-coloring problem. From our experimental results, we can conclude that a really hard DisCSP instance is the one on the crossover belt where each agent has a limited amount of knowledge of the instance, that is, a small number of intra-agent constraints and a small number of variables per agent. This view may be helpful not only in formulating some application problems as DisCSP instances, but also in developing effective strategies, especially for distributed constraint satisfaction algorithms where agents are allowed to enlarge (or join) their local problems during algorithm execution[5),13)].

Obviously, much still remains to be done to obtain a more accurate picture of the cost profiles of distributed constraint satisfaction algorithms.

As with most of the work on the cost profiles of centralized constraint satisfaction algorithms, this work has concentrated on specific cases where multi-ABT and multi-AWC, which we believe are standard distributed constraint satisfaction algorithms, are applied to the distributed graph-coloring problem. However, in order to strongly confirm the findings, we may need to conduct more experiments using other combinations of algorithms and problems.

Also, as with most of the previous work, the terms "easy" and "hard" are intuitively used in this work. Recently, a detailed investigation of a phase transition in computational complexity, namely, the complexity shifts from polynomial (easy) to exponential (hard) in the order, has been reported for the random 3-SAT problem[3]. It may be interesting to explore where such complexity shifts occur on the x-y plane for multi-ABT and multi-AWC.

References

1) Bessière, C., Maestre, A. and Meseguer, P.: Distributed Dynamic Backtracking, *Proceedings of the IJCAI-01 Workshop on Distributed Constraint Reasoning*, pp. 9–16 (2001).

2) Cheeseman, P., Kanefsky, B. and Taylor, W. M.: Where the Really Hard Problems Are, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 331–337 (1991).

3) Coarfa, C., Demopoulos, D. D., Aguirre, A. S. M., Subramanian, D. and Vardi, M. Y.: Random 3-SAT: The Plot Thickens, *Constraints*, Vol. 8, No. 3, pp. 243–261 (2003).

4) Conry, S. E., Kuwabara, K., Lesser, V. R. and Meyer, R. A.: Multistage Negotiation for Distributed Constraint Satisfaction, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 6, pp. 1462–1477 (1991).

5) Hirayama, K. and Toyoda, J.: Forming Coalitions for Breaking Deadlocks, *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 155–162 (1995).

6) Hirayama, K. and Yokoo, M.: The Effect of Nogood Learning in Distributed Constraint Satisfaction, *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, pp. 169–177 (2000).

7) Hirayama, K. and Yokoo, M.: Local Search for Distributed SAT with Complex Local Problems, *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems*, pp. 1199–1206 (2002).

8) Hogg, T. and Williams, C. P.: The Hardest Constraint Problems: A Double Phase Transition, *Artificial Intelligence*, Vol. 69, pp. 359–377 (1994).

9) Huhns, M. N. and Bridgeland, D. M.: Multiagent Truth Maintenance, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 6, pp. 1437–1445 (1991).

10) Mason, C. and Johnson, R.: DATMS: A Framework for Distributed Assumption based Reasoning, *Distributed Artificial Intelligence* (Gasser, L. and Huhns, M.(eds.)), Vol. 2, Morgan Kaufmann, pp. 293–318 (1989).

11) Mitchell, D., Selman, B. and Levesque, H.: Hard and Easy Distributions of SAT Problems, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 459–465 (1992).

12) Modi, P. J., Jung, H., Tambe, M., Shen, W.-M. and Kulkarni, S.: A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation, *Proceedings of the Seventh International Conference on Principles and Practice of Constraint Programming*, pp. 685–700 (2001).

13) Numazawa, M., Kurihara, M. and Noto, M.: Distributed Constraint Satisfaction for Mobile Agents, *Proceedings of the Tenth Workshop on Multiagent and Cooperative Computation* (2001) (in Japanese).

14) Silaghi, M. C., Sam-Haroud, D. and Faltings, B.: Asynchronous Search with Aggregations, *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pp. 917–922 (2000).

15) Smith, B. M.: The Phase Transition in Constraint Satisfaction Problems: A Closer Look at the Mushy Region, Research Report 93.41, School of Computer Studies, University of Leeds (1993).

16) Sycara, K. P., Roth, S., Sadeh, N. and Fox, M.: Distributed Constrained Heuristic Search, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 6, pp. 1446–1461 (1991).

17) Yokoo, M.: *Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems*, Springer (2001).

18) Yokoo, M., Durfee, E. H., Ishida, T. and Kuwabara, K.: The Distributed Constraint Satisfaction Problem: Formalization and Algorithms, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 5, pp. 673–685 (1998).

19) Yokoo, M. and Hirayama, K.: Distributed Breakout Algorithm for Solving Distributed Constraint Satisfaction Problems, *Proceedings of the Second International Conference on*

Multi-Agent Systems, pp. 401–408 (1996).

20) Yokoo, M. and Hirayama, K.: Distributed Constraint Satisfaction Algorithm for Complex Local Problems, *Proceedings of the Third International Conference on Multi-Agent Systems*, pp. 372–379 (1998).

21) Zhang, W. and Xing, Z.: Distributed Breakout vs. Distributed Stochastic: A Comparative Evaluation on Scan Scheduling, *Proceedings of the Third International Workshop on Distributed Constraint Reasoning*, pp. 192–201 (2002).

(Received July 18, 2003)
(Accepted June 8, 2004)

Katsutoshi Hirayama was born in 1967. He received his M.E. and Ph.D. degrees from Osaka University in 1992 and 1995, respectively. Since then he had been working at Kobe University of Mercantile Marine as a research associate since 1995, a lecturer since 1997, and an associate professor since 2001. After consolidation of Kobe University and Kobe University of Mercantile Marine in 2003, he has been in the faculty of maritime sciences, Kobe University as an associate professor. During the period, he had been in Carnegie Mellon University as a visiting scientist from 1999 to 2000. His current research interests are constraint satisfaction/optimization algorithms and distributed problem solving. He is a member of Japanese Society for Artificial Intelligence (JSAI), Japan Society for Software Science and Technology (JSSST), Operations Research Society of Japan (ORSJ), and American Association for Artificial Intelligence (AAAI).

Makoto Yokoo received the B.E. and M.E. degrees in electrical engineering, in 1984 and 1986, respectively, form the University of Tokyo, Japan, and the Ph.D. degree in information and communication engineering in 1995, from the University of Tokyo, Japan. From 1986 to 2004, he was a research scientist of Nippon Telegraph and Telephone Corporation (NTT). He is currently a Professor of Information Science and Electrical Engineering, Kyushu University. His research interests include multi-agent systems, constraint satisfaction, and mechanism design among self-interested agents. He received the ACM SIGART Autonomous Agents Research Award in 2004. He is an associate editor of Journal of Artificial Intelligence Research. He is on the board of directors of International Foundation for Multiagent Systems (IFMAS) and Japan Society for Software Science and Technology.

Katia Sycara was born in Athens, Greece. She received her B.S degree from Brown University, her MS from University of Wisconsin and her PhD from Georgia Institute of Technology in 1987. Since then she has been working at Carnegie Mellon University where she is currently Professor in the School of Computer Science. She has been conducting and directing research on Multi Agent Systems, Semantic Web Services, Negotiation, Multiagent Learning. She is a Fellow of the American Association of Artificial Intelligence and the recipient of the 2002 ACM Agents Research Award. She is a founding Editor-in-Chief of the International Journal of Autonomous Agents and Multiagent Systems, and on the editorial Board of 5 other journals. She has servied as General Chair of the Second International Conference on Autonomous Agentsm (Agents 98), as Program Chair of the Second International Semantic Web Conference (ISWC 03), and of the Conference on Ontologies and Data Bases (ODBASE 04). She is a founding member of the board of the International Federation of Multiagent Systems (IFMAS), she serves on the Autonomous Agents Steering Committee, and she is a founding member of the Semantic Web Science Association. She is the US co-chair of the Semantic Web Services Initiative (SWSI). She is the author or co-author of more than 200 scientific papers and has given numerous intivited talks. She is on the Scientific Adivsory Board of France Telecom. She holds an honorary Doctorate from the University of the Aegean (2004).