Effect of DisCSP Variable-Ordering Heuristics in Scale-free Networks

Tenda Okimoto, Atsushi Iwasaki, and Makoto Yokoo

Kyushu University, Fukuoka 8190395, Japan
{tenda@agent., iwasaki@, yokoo@}is.kyushu-u.ac.jp

Abstract. A Distributed Constraint Satisfaction Problem (DisCSP) is a constraint satisfaction problem in which variables and constraints are distributed among multiple agents. Various algorithms for solving DisCSPs have been developed, which are intended for general purposes, i.e., they can be applied to any network structure. However, if a network has some particular structure, e.g., the network structure is scale-free, we can expect that some specialized algorithms or heuristics, which are tuned for the network structure, can outperform general purpose algorithms/heuristics.
In this paper, as an initial step toward developing specialized algorithms for particular network structures, we examine variable-ordering heuristics in scale-free networks. We use the classic asynchronous backtracking algorithm as a baseline algorithm and examine the effect of variable-ordering heuristics. First, we show that the choice of variable-ordering heuristics is more influential in scale-free networks than in random networks. Furthermore, we develop a novel variable-ordering heuristic that is specialized to scale-free networks. Experimental results illustrate that our new variable-ordering heuristic is more effective than a standard degree-based variable-ordering heuristic. Our proposed heuristic reduces the required cycles by 30% at the critical point.

1 Introduction

A surprisingly wide variety of Artificial Intelligence (AI) problems can be formalized as constraint satisfaction problems (CSPs). A CSP is a problem that finds a consistent assignment of values to variables. A Distributed Constraint Satisfaction Problem (DisCSP) is formalized as a CSP in which variables and constraints are distributed among multiple agents [1]. In DisCSP, agents assign values to variables, attempting to generate a locally consistent assignment that is also consistent with all the constraints between agents.

Asynchronous BackTracking algorithm (ABT), which was first presented by Yokoo [2], is the most basic algorithm for solving DisCSPs. It is also the first complete and asynchronous search algorithm for DisCSPs. ABT allows agents to act asynchronously and concurrently without any global control, while guaranteeing the completeness of the algorithm. Various algorithms have been developed for solving DisCSPs, e.g., Distributed BackTracking algorithm [3], an ABT

based algorithm without adding links [4], Dynamic Distributed BackJumping [5], Asynchronous Partial Overlay [6], and Dynamic ordering for ABT [7, 8].

Recently, *scale-free graphs* in complex networks, introduced by Barabási and Albert [9, 10], has become a very popular interdisciplinary research topic. These graphs have been proposed as a generic and universal model of network topologies that exhibit power-law distributions in the connectivity of network nodes. A scale-free network is inhomogeneous in nature, i.e., there exist a small number of nodes that have many connections, while most nodes have very few connections,

Although there has been a lot of research in network structure for CSPs [11, 12], there is little research in network structure for DisCSP. In this paper, as an initial step toward developing specialized algorithms/heuristics for particular network structures in DisCSP, we examine the effect of variable-ordering heuristics of ABT in scale-free networks. Although a variety of more efficient, sophisticated algorithms have been developed for solving DisCSPs, we focus on ABT as a baseline algorithm, since it is one of the simplest algorithms and is suitable for our purpose. We believe that our analysis and results can be applied to other sophisticated algorithms.

In the rest of this paper, we show that the choice of variable-ordering heuristics is more influential in scale-free networks than in random networks. Specifically, we show that the performances of ABT in the former network depend on which variable-ordering heuristics is used much more than that in the latter network, since the degree distribution of scale-free networks is significantly different from that of random networks. We examine how the performance of ABT in scale-free networks changes in terms of the depth and number of the backedges of pseudo-trees. Given a variable-ordering, ABT determines a pseudo-tree and searches for a solution from it. Since the depth and number of backedges greatly affect the network structure, it is expected that the performance of ABT changes based on those factors. However, surprisingly, our experiments reveal that the performance does not significantly change.

Furthermore, we develop a novel variable-ordering heuristic called Average Length between Hubs (ALH) specialized for scale-free networks. Our experiments show that ALH outperforms a standard degree-based variable-ordering heuristic in scale-free networks. As far as the authors aware, there exists virtually no work on variable-ordering heuristics specialized for scale-free networks in DisCSP, although many studies have dealt with variable-ordering heuristics [8, 13, 14, 15, 16].

The rest of our paper is organized as follows. We describe the definition of a DisCSP (Section 2) and introduce a scale-free network (Section 3). We examine the performance of ABT in scale-free and random networks (Section 4). Next, we present a novel variable-ordering heuristic that is specialized to scale-free networks and show that our new variable-ordering heuristic is effective for scale-free networks (Section 5). Finally, we give a discussion (Section 6) and present a conclusion and some future work (Section 7).

2 Distributed Constraint Satisfaction Problem

A Constraint Satisfaction Problem (CSP) [17] consists of m variables $x_1, ..., x_m$, whose values are taken from finite, discrete domains $D_1, ..., D_m$, respectively, and a set of constraints on their values. A constraint is defined by a predicate. That is, the constraint $p(k; x_{k1}, ..., x_{kj})$ is a predicate that is defined on Cartesian product $D_{k1} \times ... \times D_{kj}$. This predicate is true iff the value assignment of these variables satisfies this constraint. Solving a CSP is equivalent to finding an assignment of values to all variables such that all constraints are satisfied.

A Distributed Constraint Satisfaction Problem (DisCSP) is a CSP in which the variables and constraints are distributed among multiple agents [1]. We assume the following communication model:

- Agents communicate by sending messages. An agent can send messages to other agents iff the agent knows the addresses of the agents.
- The delay in delivering a message is finite, although random. For transmission between any pair of agents, messages are received in the order in which they were sent.

Each agent has variables and tries to determine their values. However, there exist inter-agent constraints, and the value assignment must satisfy these inter-agent constraints.

2.1 Asynchronous Backtracking

Asynchronous BackTracking algorithm (ABT), which was first presented by Yokoo [1], is the most basic algorithm for solving DisCSPs. We make the following assumptions while describing this algorithm for simplicity. Relaxing these assumptions to general cases is relatively straightforward:

- Each agent has exactly one variable.
- All constraints are binary.
- Each agent knows all constraint predicates relevant to its variable.

In ABT, the priority order among agents is determined. First, agents instantiate their variables concurrently and send their assigned values to the agents that are connected to them by outgoing links, i.e., there exists a link between two agents who are involved by a binary constraint, and the link is directed from the higher priority agent to the lower priority agent. Then all agents wait for and respond to messages. After each update of its assignment, an agent sends its new assignment to all outgoing links. An agent that receives an assignment from an incoming link, tries to find an assignment for its variable that does not violate a constraint with the assignment it received.

The main message types communicated among agents are *ok*? messages and *nogood* messages. An *ok*? message carries an assignment of an agent. When agent A_i receives an *ok*? message from agent A_j, it places the received assignment in a data structure called Agent_View, which holds the last assignment A_i received

from higher priority neighbors such as A_j. Next, A_i checks if its current assignment is still consistent with its Agent_View. If it is consistent, A_i does nothing. If not, then A_i searches its domain for a new consistent value. If it finds one, it assigns its variable and sends *ok*? messages to all lower priority agents linked to it. Otherwise, A_i backtracks.

The backtrack operation is executed by sending a *nogood* message that contains an inconsistent partial assignment. *nogood* messages are sent to the agent with the lowest priority among the agents whose assignments are included in the inconsistent tuple in the *nogood* message. Agent A_i that sends a *nogood* message to agent A_j assumes that A_j will change its assignment. Therefore, A_i removes from its Agent_View the assignment of A_j and makes an attempt to find an assignment for its variable that is consistent with the updated Agent_View.

3 Scale-Free Network

In recent years, various complex networks have been identified as having a scale-free structure [9, 18, 19], e.g., the Internet, SNS, and the citation relation graphs of scientific articles. Traditionally, these networks are approximated as random graphs, but the degree distributions of these networks (and other networks in nature) are significantly different from the degree distribution of random graphs. The study of random graphs was originally initiated by Erdös and Rényi (ER model) [20]. In this model, most nodes have approximately the same degree and the degree distribution follows a Poisson distribution.

A scale-free network is characterized by a power-law degree distribution as follows:

$$p(k) \propto k^{-\gamma},$$

where k is the degree and γ is the exponent that depends on each network structure. Scale-free networks have no *scale* because there is no typical number of links. The random network models assume that the probability that two nodes are connected is random and uniform. In contrast, most real networks exhibit some preferential connectivity. For example, a newly created webpage will more likely include edges to well-known, popular documents that already have high connectivity. This example indicates that the probability with which a new node connects to an existing node is not uniform, but there is a higher probability of being linked to a node that already has a large number of connections.

4 Influence of Variable-Ordering Heuristics in Scale-Free Networks

In this section, we show that the choice of variable-ordering heuristics is more influential in scale-free networks than in random networks. We used a discrete event simulation [21] to compare the performance of ABT-based algorithms, where each agent maintains its own simulated clock. An agent 's time is incremented by one simulated time unit whenever it performs one computation cycle.

One cycle consists of reading all incoming messages, performing local computation, and sending messages. We assume that a message issued at time t is available to the recipient at time t + 1. We analyzed the performance for the number of cycles required to solve the problem.

There are some other simulations to evaluate DisCSP algorithms, e.g., Non Concurrent Constraints Checks (NCCCs). However, we analyze different variable ordering heuristics on a single algorithm and the computational cost for each cycle is almost identical. Therefore, we believe that using only cycles rather than NCCCs is enough.

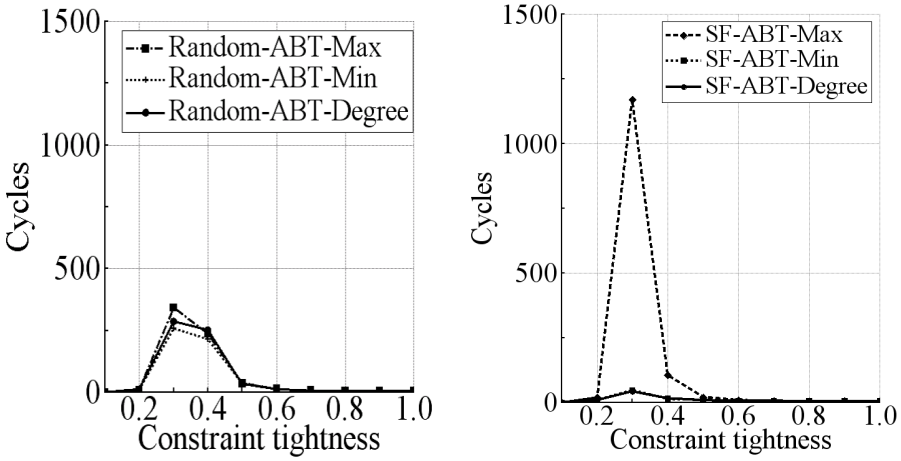
In this paper, the Java program developed by Sun Microsystems Laboratories is used as a scale-free network formation tool [22]. This program can generate scale-free networks giving the number of *nodes*, exponent γ, and the minimal degree of each agent *md*. More specifically, this program can generate a power-law list of nodes and edges.

We examine the performance of ABT in random and scale-free networks. Scale-free networks are generated by the tool with the following parameters: *nodes*=100, *md*=2, and γ=1.8. To generate random networks, we chose *nodes*=100 and *edges*=247, so that the number of constraints will resemble those of the scale-free networks[1]. We set the domain size of each variable to three, i.e., *domain*=3 which means $|D_1|$=,...,=$|D_m|$=3 for m variables $x_1, ..., x_m$. For the evaluations, we generate ten random and ten scale-free networks. Assume r_1,...,r_{10} for the ten random networks and sf_1,...,sf_{10} for the ten scale-free networks. For each network, the constraint tightness is varied from 0.1 to 0.9 by 0.1. For each constraint tightness, 100 random problem instances are generated. Thus, the results represent the averages of these 100 instances in all ten networks. For a variable-ordering of ABT, we determine ten different random variable-orderings.

In Figure 1, we show the performance of ABT with three different random variable-orderings in random network r_1 and scale-free network sf_1 that exhibit characteristic results. When the constraint tightness is less than 0.3 or greater than 0.3, ABT can terminate early, i.e., ABT can easily find a solution for less than 0.3, and it can easily find that the problem is unsolvable for greater than 0.3. When the constraint tightness equals 0.3, the required cycles of ABT are maximum in r_1 and sf_1. We call such a peak the *critical point*.

Random-ABT-Max (Random-ABT-Min) represents the performance of ABT in random network r_1, whose required cycles at the critical point are maximum (minimum). SF-ABT-Max and SF-ABT-Min represent the performance of ABT as above in scale-free network sf_1. In addition, Random-ABT-Degree and SF-ABT-Degree represent the performance of ABT with a standard degree-based variable-ordering heuristic. In this heuristic, the priority of nodes is determined one by one. First, we choose n_{1st}, which has the highest degree. Second, we choose node n_{2nd}, which has the highest degree and connected to n_{1st}. Similarly, we keep on choosing a node, that has the highest degree with the nodes already chosen, breaking ties using the degree with the unchosen nodes.

[1] For γ=2.2, 2.6, 3.0, the essential results did not change.



(a) Performance of ABT in random network r_1

(b) Performance of ABT in scale-free network sf_1

Fig. 1. Performance of ABT in random and scale-free networks

The performance of ABT significantly depends on variable-ordering in scale-free networks. In random network r_1, the required cycles at the critical point vary from 235 to 283 cycles (Figure 1(a)). On the other hand, in scale-free network sf_1, the required cycles vary from 47 to 1171 cycles (Figure 1(b)). We confirmed that similar results were obtained in other networks, i.e., in r_2,...,r_{10} and sf_2,...,sf_{10}. Particularly, in scale-free networks, ABT with a standard degree-based variable-ordering heuristic requires the smallest cycles at the critical point.

Furthermore, we examine the effect of the depth and the number of backedges in a pseudo-tree on the performance of ABT in scale-free networks. According to a variable-ordering, a pseudo-tree is determined whose depth is the length of the

Table 1. Depth and number of backedges of pseudo-trees and required cycles at critical point

ABT	Depth	Backedges	Cycles
ABT 1	17	207	10253
ABT 2	14	176	7815
ABT 3	15	220	2279
ABT 4	22	327	1673
ABT 5	13	173	777
ABT 6	12	175	380

longest path from the root agent to one of the leaf agents. A backedge is a link between two agents that are not in a direct parent-child relationship. Our initial expectation was that the performance of ABT would improve with shallower depth and fewer backedges. In Table 1, we show the depth and the number of backedges of the pseudo-trees and the required cycles of ABT at the critical point with six different variable-orderings, where *domain*=10. Here, we increased the domain size to make the required cycles vary significantly according to variable-orderings. As shown in Table 1, we cannot see any direct relationship between the performance and the parameters we examined (i.e., tree depth and number of backedges). For example, in "ABT 1", the required cycle at the critical point is 10253, the depth is 17, and the number of backedges is 207. On the other hand, in "ABT 4", the required cycle at the critical point is 1673, the depth is 22, and the number of backedges is 327.

The experimental results reveal that the choice of variable-ordering heuristics is influential in scale-free networks. We don't see any direct relationship between the performance of ABT and the parameters of a pseudo-tree (i.e., depth and number of backedges).

5 A Variable-Ordering Heuristic for Scale-Free Networks

The previous section reveals that the performance of ABT is affected by the priority of the order of agents, i.e., variable-ordering. In this section, we propose a novel variable-ordering heuristic called Average Length between Hubs (ALH). Based on the results so far, since ALH focuses on the average length between hubs, it is specialized for scale-free networks. This section introduces our proposed variable-ordering heuristics and evaluates the performance by simulation.

5.1 Heuristic

Let $G = (N, E)$ be a graph, where $N = \{n_i | i \in \mathbb{N}\}$ is a set of nodes (agents) and $E = \{e(n_i, n_j) | n_i, n_j \in N, n_i \neq n_j\}$ is a set of edges. A node is called a hub if it has a larger number of connections than constant $c \in \mathbb{N}$. Let H be set of hubs

$$H = \{n_i | n_i \in N, deg(n_i) \geq c\}$$

where $deg(n_i)$ is the degree of node n_i. Each agent knows whether he belongs to H.

Next, we define *border-set* nodes by using the distance between nodes $dis : N \times N \rightarrow \mathbb{N}$, i.e., $dis(n_i, n_j)$ gives the number of the edges of the shortest path between n_i and n_j. For node n_i, the average distance of the shortest paths to each hub in H is defined as follows:

$$n_i^{av} = \Sigma_{n_j \in H} dis(n_i, n_j)/|H|.$$

The average distance between hubs is defined as follows:

$$h^{av} = \Sigma_{n_i \in H} n_i^{av}/|H|.$$

Then, border-set BS is defined as:

$$BS = \{n_i \mid n_i^{av} \leq h^{av}\}.$$

The priorities of agents are determined using BS. Basically, a node in BS has a higher priority than a node that is not in BS. Between two nodes in BS, the node that is not in H has a higher priority. If two nodes, n_i and n_j, in BS are also in H, then n_i has a higher priority than n_j when $deg(n_i) > deg(n_j)$ (and vice versa). If two nodes, n_i and n_j, in BS are not in H, then n_i has a higher priority than n_j when $n_i^{av} < n_j^{av}$ (and vice versa). Ties are broken using the degrees. Further ties are broken using the lexicographical order of identifiers. Then the priority among two nodes that are not in BS is determined by the total distance between BS. More specifically, for node $n_i \notin BS$, denote the total distance to the nodes of BS as $td(n_i) = \sum_{n_j \in BS} dis(n_i, n_j)$. For two nodes, n_i and n_j that are not in BS, n_i has a higher priority than n_j when $td(n_i) < td(n_j)$ (and vice versa). Ties are broken using the degrees. Further ties are broken using the lexicographical order of identifiers.

If all hubs are directly connected, the ALH becomes equivalent to a degree-based heuristic, since BS contains only nodes in H. Consider a scale-free network where each hub is not directly connected. In ALH, the nodes in BS have the highest priority, i.e., the node in BS has a higher priority than the hubs. Let us consider the pseudo-tree defined by this ordering. In the pseudo-tree, the hubs are placed below the nodes in BS, i.e., the hubs are siblings of the nodes in BS. Also, under each hub, we can expect that there exists a cluster of nodes, which is independent from other clusters, given that the values of variables in BS are determined. Thus, we can expect that ABT can efficiently solve such a problem instance since these clusters can be solved independently. The cost of implementing the proposed heuristic ALH is enough low compared to the cost of the ABT, since finding a shortest path can be done O(n).

Let us show a simple example. A constraint network of a DisCSP represented as Figure 2(a) exists, where H_1 and H_2 are hubs. For nodes $H_1, H_2, n_1, \ldots, n_4$, their degrees satisfy the following condition: $deg(H_1) > deg(H_2) > deg(n_1) > deg(n_2) > deg(n_3) > deg(n_4)$. Figure 2(b) represents the pseudo-tree determined by a degree-based heuristic. Since H_1 has the highest degree, it becomes the root of this pseudo-tree. Since h^{av}=1, n_1^{av}=n_2^{av}=1, n_3^{av}=2, and n_4^{av}=3, BS is determined as follows:

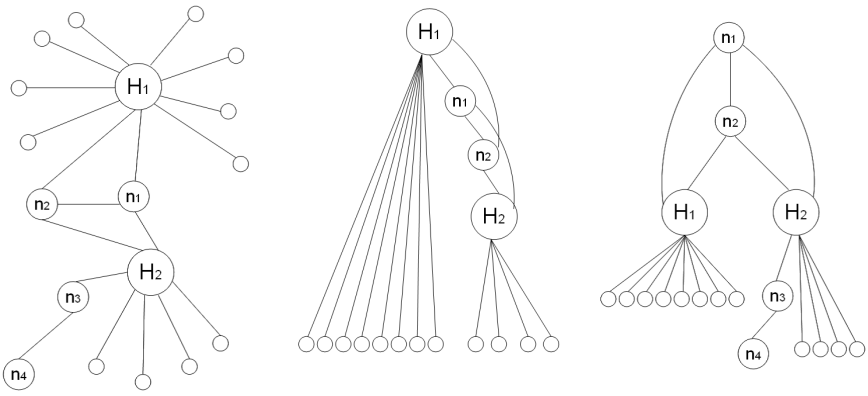
$$BS = \{H_1, H_2, n_1, n_2\}.$$

Thus, among these nodes, priority ordering is determined as:

$$n_1, n_2, H_1, H_2, n_3, n_4,$$

where n_1 is the highest and n_4 is the lowest.

Figure 2(c) represents the pseudo-tree determined by ALH. The nodes in BS are placed around the root of the pseudo-tree. The hubs are placed just below the root as siblings, and in particular, hubs H_1 and H_2 are placed on different branches in this pseudo-tree.



(a) Constraint network representing a DisCSP

(b) Pseudo-tree determined by degree-based heuristic

(c) Pseudo-tree determined by ALH

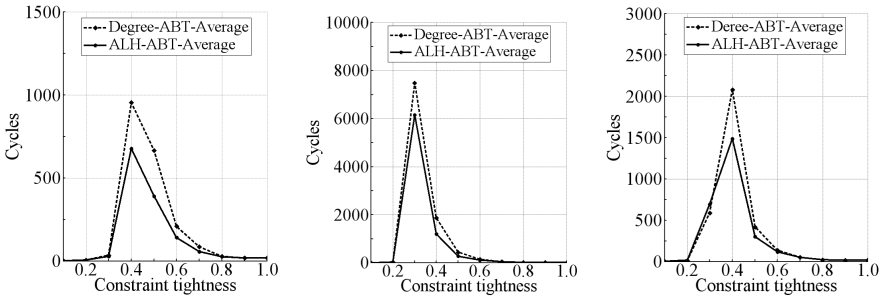
Fig. 2. Example

5.2 Evaluations

In our evaluations, we show that ALH is effective and can reduce the required cycles at the critical point in scale-free networks. More specifically, we compare the effect of ALH compared to a standard degree-based heuristic in the following three kinds of scale-free networks:

(SFN 1): $nodes$ =100, γ=1.8, and md=2,
(SFN 2): $nodes$ =200, γ=1.8, and md=2,
(SFN 3): $nodes$ =100, γ=1.8, and md=3.

The evaluations were conducted with $domain$=10. For each parameter we generated ten scale-free networks. For each network, the constraint tightness was varied from 0.1 to 0.9 by 0.1. For each constraint tightness, 100 random constraint instances were generated. The results represent the averages of these 100 instances for all ten scale-free networks (1000 in total). The experimental results in SFN 1 are summarized in Figure 3(a), in which ALH-ABT-Average represents the performance of ABT with the ALH variable-ordering heuristic and Degree-ABT-Average represents the performance of ABT with the standard degree-based variable-ordering heuristic. Here, the critical point appears when the constraint tightness is around 0.4. At the critical point, ALH-ABT-Average requires 678 cycles while Degree-ABT-Average requires 955 cycles. Thus, ALH-ABT-Average performs approximately 30% better than Degree-ABT-Average at the critical point in SFN 1. We confirmed that ALH is also effective in SFN 2 and SFN 3.



(a) Effect of ALH and standard degree-based heuristics in SFN 1

(b) Effect of ALH and standard degree-based heuristics in SFN 2

(c) Effect of ALH and standard degree-based heuristics in SFN 3

Fig. 3. Effect of ALH compared to standard degree-based heuristics in SFN 1, SFN 2 and SFN 3.

The experimental results in SFN 2 are summarized in Figure 3(b). The network in SFN 2 is larger than that in SFN 1, i.e, the number of nodes in SFN 2 is 200, compared to 100 in SFN 1. ALH-ABT-Average requires 6156 cycles and Degree-ABT-Average requires 7487 cycles at the critical point. Thus, ALH-ABT-Average performs approximately 19% better than Degree-ABT-Average.

The experimental results in SFN 3 are summarized in Figure 3(c). The network in SFN 3 is more complicated than SFN 1, i.e., the minimal degree of each agent increased from md=2 to md=3. ALH-ABT-Average requires 1489 cycles and Degree-ABT-Average requires 2083 cycles at the critical point. ALH-ABT-Average performs approximately 30% better than Degree-ABT-Average at the critical point in SFN 3.

The experimental results reveal that ALH outperforms the standard degree-based heuristic in three scale-free networks, varying the number of nodes and the minimal degree. We also confirmed that fact did not change with other parameter settings.

6 Discussion

The previous section showed that the standard degree-based heuristic is outperformed by the ALH heuristic. One might expect that it is also outperformed by the other simple heuristics, since the ALH heuristic, particularly its way of determining the border set, is somewhat complicated. Thus, we consider a simple variable-ordering heuristic, called a *naive* heuristic described below.

Let us define a naive border-set (NBS) for that heuristic, instead of a BS for ALH. Whether a node belongs to NBS is determined by the distance between the two nearest hubs to the node. Formally, for node $n_i \in N$, denote the two

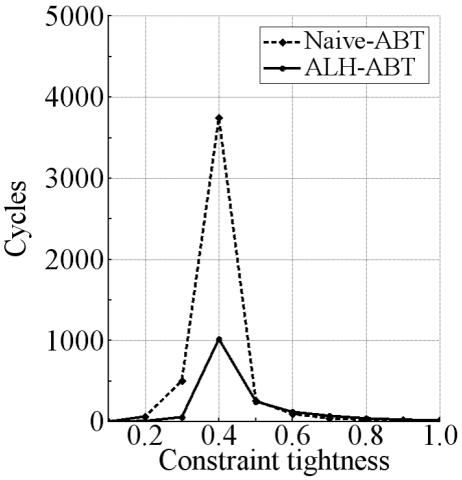


Fig. 4. Effect of ALH compared to *naive* heuristic in SFN 3

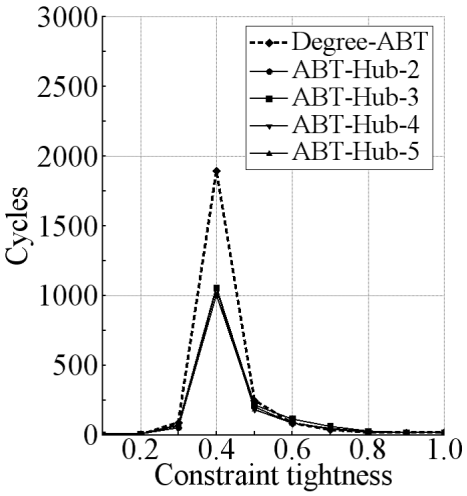


Fig. 5. Effect of ALH by increasing number of hubs from 2 to 5 in SFN 3

nearest hubs from n_i as $h_{i,1st}, h_{i,2nd} \in H$ and denote the distances to $h_{i,1st}$ and $h_{i,2nd}$ from the node as $dis_{1st}(n_i)$, $dis_{2nd}(n_i)$, respectively. Then, we define NBS as follows:

$$NBS = \{n_i \mid |dis_{1st}(n_i) - dis_{2nd}(n_i)| \leq 1\}.$$

In short, NBS contains nodes that lies exactly in the middle of the two nearest hubs. The priority among nodes is determined in exactly the same way as ALH, except that we use NBS instead of BS. The experimental result is summarized in Figure 4. Naive-ABT represents the performance of ABT with the naive heuristic. ALH-ABT performs approximately 3.7 times better than Naive-ABT at the critical point. Precisely, the required cycles for ALH-ABT is 1016, while that for Naive-ABT is 3744 at that point. As a result, we can say that the simplified version of the heuristics fails to perform as well as ALH.

We examined the reason why the performance of the *naive* heuristic is much worse than that of ALH, and found that the size of NBS of the *naive* heuristic is much larger than the size of BS of ALH. In fact, NBS contains at least twice as many agents as BS. Therefore, we conjecture that the size of BS should be small; otherwise, the priority based on BS becomes less informative.

In previous evaluations, we set the number of hubs to two. This seems to be a reasonable choice to make the size of the border-set (BS) small. We further examine the performance of ABT with the ALH heuristic by varying the number of hubs from two to five. The experimental results are summarized in Figure 5, in which ABT-Hub-k represents the performance of ABT when choosing the number of hubs as k. The performance is basically unchanged even if we change the number of hubs, i.e., the required cycles at the critical point for ABT-Hub-5 is 1048, while that for ABT-Hub-2 is 1004. These results imply that the choice

of the number of hubs is not so influential to the performance of ABT with the ALH heuristic.

Note that, this result does not explain how many hubs we should choose in any scale-free networks. We will research a good value for some of constants (constant for selecting hubs) as a future work.

7 Conclusions

In this paper, we showed that the choice of variable-ordering heuristics is more influential in scale-free networks than in random networks. We observed that in scale-free networks there is more significant difference between maximum and minimum of the required cycles than in random networks.

Furthermore, we examined how the performance of ABT in scale-free networks changes in terms of the depth and number of backedges of pseudo-trees. The experimental result revealed that these parameters do not significantly affect the performance of ABT in scale-free networks.

Finally, we developed a novel variable-ordering heuristic called Average Length between Hubs (ALH) specialized for scale-free networks. We showed that ALH outperforms a standard degree-based variable-ordering heuristic in scale-free networks and can reduce the required cycles by 30% at the critical point.

As future works, we hope to develop dynamic variable-ordering heuristics/algorithms that are specialized to scale-free networks.

References

[1] Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K.: The distributed constraint satisfaction problem: formalization and algorithms. IEEE Transactions on Knowledge and Data Engineering **10**(5) (1998) 673–685

[2] Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K.: Distributed constraint satisfaction for formalizing distributed problem solving. In: Proceedings of the Twelfth IEEE International Conference on Distributed Computing Systems. (1992) 614–621

[3] Hamadi, Y.: Backtracking in distributed constraint networks. In: International Journal on Artificial Intelligence Tools. (1998) 219–223

[4] Bessiere, C., Brito, I., Maestre, A., Meseguer, P.: Asynchronous backtracking without adding links: a new member in the ABT family. Artificial Intelligence **161** (2005) 2005

[5] Nguyen, V., Sam-Haroud, D., Faltings, B.: Dynamic distributed backjumping. In: CSCLP. (2004) 71–85

[6] Mailler, R., Lesser, V.R.: Asynchronous partial overlay: A new algorithm for solving distributed constraint satisfaction problems. Journal of Artificial Intelligence Research (JAIR) **2005** (2006) 2006

[7] Zivan, R., Meisels, A.: Dynamic ordering for asynchronous backtracking on DisCSPs. Constraints **11**(2-3) (2006) 179–197

[8] Silaghi, M.C.: Framework for modeling reordering heuristics for asynchronous backtracking. In: IAT. (2006) 529–536

[9] Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286** (October 15 1999) 509–512
[10] Barabási, A.L.: Linked: The new science of networks. J. Artificial Societies and Social Simulation **6**(2) (2003)
[11] Walsh, T.: Search on high degree graphs. In: IJCAI. (2001) 266–274
[12] Jin, X., Liu, J.: Agent network topology and complexity. In: AAMAS. (2003) 1020–1021
[13] Hamadi, Y.: Interleaved backtracking in distributed constraint networks. International Journal on Artificial Intelligence Tools **11**(2) (2002) 167–188
[14] Arbelaez, A., Hamadi, Y.: Exploiting weak dependencies in tree-based search. In: SAC. (2009) 1385–1391
[15] Sultanik, E., Lass, R.N., Regli, W.C.: Dynamic configuration of agent organizations. In: IJCAI. (2009) 305–311
[16] Ezzahir, R., Bessiere, C., Wahbi, M., Benelallam, I., Bouyakhf, H.: Asynchronous inter-level forward-checking for DisCSPs. In: CP. (2009) 304–318
[17] Mackworth, A.: Constraint Satisfaction. In Shapiro, S.C., ed.: Encyclopedia of Artificial Intelligence, John Wiley & Sons (1992) 285–293 Volume 1, second edition.
[18] Buchanan, M.: Nexus: Small worlds and the groundbreaking science of networks. J. Artificial Societies and Social Simulation **6**(2) (2003)
[19] Li, L., Alderson, D., Doyle, J.C., Willinger, W.: Towards a theory of scale-free graphs: Definition, properties, and implications. Internet Mathematics **2** (2005) 4
[20] Erdös, P., Rényi, A.: On random graphs i. Publicationes Mathematicae Debrecen **6** (1959) 290–297
[21] Yokoo, M., Hirayama, K.: Algorithms for distributed constraint satisfaction: A review. Journal of Autonomous Agents and Multi-agent Systems **3**(2) (2000) 189–211
[22] Densmore, O.: An exploration of power-law networks. (2009) http://backspaces.net/sun/PLaw/index.html.