Efficient methods for pseudo-tree based distributed best first search

Toshihiro Matsui and Hiroshi Matsuo [*]

Abstract— **Distributed constraint optimization problem is an area of research in multi agent system. In recent years, a distributed constraint optimization algorithm, which performs best-first search in bottom up manner according to pseudo tree, was proposed. In this paper, we propose several efficient methods for the distributed bottom up best-first search. Derivation of partial solution is introduced to decrease number of backtracking among agents, Synchronization control method is applied to decrease number of communication message cycles. Additionally, error bounds are applied to obtain quasi optimal solution within less number of message cycles. Results of experiments are shown for the efficiency evaluation of the proposed heuristics methods.**

Keywords: multiagent system, distributed constraint optimization problem, pseudo-tree, dynamic programming, branch and bound

1 Introduction

Distributed constraint optimization problem (DCOP) [1] [2] [3] [4] is an area of research in multiagent system.

In recent years, a number of algorithms, which solve DCOPs using pseudo tree, have been developed [4], [7]. The pseudo tree is an efficient structure, which provides divide and concur techniques for search algorithms. These distributed search algorithms perform dynamic programming and branch-and-bound method. Another pseudo tree based algorithm[8] performs best-first search in bottom up manner. The best-first search is a well known heuristic method. However, simple bottom up computation causes a large number of backtracking.

In this paper, we propose several efficient methods for the distributed bottom up best-first search. Derivation of partial solution is introduced to decrease number of backtracking among agents, Synchronization control method is applied to decrease number of communication message cycles. Additionally, error bounds are applied to obtain quasi optimal solution within lesser number of message cycles.

[*]Nagoya Institute of Technology, Nagoya-shi Showa-ku Gokiso-cho Aichi Japan 466-8555. Email: {matsui.t, matsuo}@nitech.ac.jp

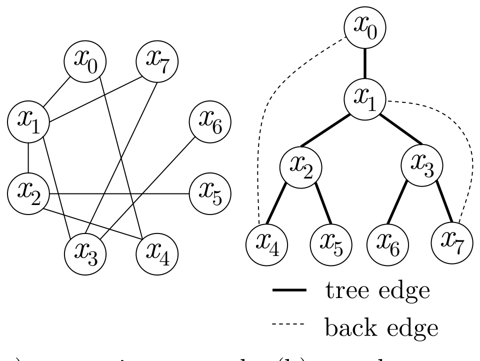

Figure 1: Pseudo tree

2 Formalization

In this section, we formalize distributed constraint optimization problem, pseudo-tree, and best-first search method.

2.1 Distributed constraint optimization problem

A distributed constraint optimization problem is formalized as follows.

- $X = \{x_1, \cdots, x_n\}$ is a set of variables.
- $D = \{D_1, \cdots, D_m\}$ is a set of domain of variables.
- F is a set of binary function. $f_{i,j} \in F$ is a function $D_i \times D_j \rightarrow R$. The function $f_{i,j}$ denotes utility of a pair of values (i.e. $\{(x_i, d_i), (x_j, d_j)\}$).

The optimal solution of the problem is an assignment which maximizes sum of utilities. Each agent (node) i has an own variable x_i. Each agent performs message passing communication.

2.2 Pseudo tree

Pseudo tree[7] gives variable ordering of constraint network. A typical pseudo tree is generated by depth first search on constraint network. An example of pseudo tree is shown in figure 1. Pseudo tree contains all nodes and all edges of original graph. The edges are categorized into tree edges and back edges. There are no edges between different sub-trees. Therefore, divide and concur techniques can be used in search algorithm. Moreover, the pseudo tree contains spanning tree of original graph. The spanning tree determines communication paths of distributed computation. We will use several notations as follows.

- p_i: Parent node of node i.
- C_i: Set of child nodes of node i.
- PP_i: Subset of ancestor nodes of node i. The nodes in PP_i are directly connected to node i through back edges.
- $\overline{PP_i}$: Subset of ancestor nodes of node i. A node in $\overline{PP}_i$ is directly connected to at least one nodes of sub tree which is routed at node i. $\overline{PP}_i$ includes p_i.

2.3 Best-first search using pseudo-tree

Best-first search is a well known heuristic method. Boundary of utility for distributed bottom-up best-first search, which uses pseudo-tree, is formalized as follows[8].

Let S_i denote a set of partial solutions which is related to sub tree rooted at node i. S_i contains assignments of values of variables for all node in $\{i\} \cup \overline{PP}_i$. If i is a root node of pseudo tree, S_i contains assignments for node i. If i is a leaf node, S_i contains assignments for all nodes in $\{i\} \cup \{p_i\} \cup PP_i$.

Let $u(s)$ denote utility of partial solution s. $u(s)$ takes a value in $[0, \infty)$.

Each node sends partial solution s and $u(s)$ to its parent node. When node i sends s and $u(s)$ to parent node p_i, node i removes (x_i, d_i) from the solution s. Let S_i^- a set of partial solution such that its (x_i, d_i) is removed. Let S_i^{-sent} a set of partial solutions which are already sent by i. S_i^{-sent} is a sub set of S_i^-. An important assumption is that each node i sends solution and utility in best-first manner. The assumption is shown as follows.

$$\forall s' \in S_i^{-sent},\ \forall s \in S_i^- \backslash S_i^{-sent},\ u(s') > u(s). \quad (1)$$

Let $S_{i,j}$ a set of partial solutions which are sent from node $j \in C_i$ and received by node i. $S_{i,j}$ is a sub set of S_i. Let $s_j^{last} \in S_{i,j}$ denote partial solution which is most recently received. Boundary of utility for $S_{i,j}$ is shown as follows.

$$UB_{i,j}(s) = \begin{cases} \infty & S_{i,j} = \phi \\ u(s_j^{last}) & S_{i,j} \neq \phi, \\ & \{s' | s' \simeq s, s' \in S_{i,j}\} = \phi \\ \max\limits_{s \simeq s_j, s_j \in S_{i,j}} u(s_j) & otherwise \end{cases} \quad (2)$$

$$LB_{i,j}(s) = \begin{cases} \max\limits_{s \simeq s_j, s_j \in S_{i,j}} u(s_j) & \{s' | s' \simeq s, s' \in S_{i,j}\} = \phi \\ 0 & otherwise \end{cases} \quad (3)$$

Algorithm 1: ODPOP (synchronized)

```
Initiation:
$S_i^{-sent} \leftarrow \phi$; $asked_i \leftarrow$ **false**; $WaitForReply_i \leftarrow \phi$;

Management:
if ($i$ is the root node $\vee$ $asked_i$)$\wedge|WaitForReply_i| = 0$ {
 if sufficient condition is satisfied {
  if $i$ is the root node {
   select best assignment $(x_i, d_i^*)$;
   send VALUE($\{(x_i, d_i^*)\}$) messages to $j \in C_i$;
   terminate; }
  else {
   select next best solution $s$;
   send GOOD($s$,$u(s)$) message to $p_i$; $asked_i \leftarrow$ **false**; }}
 else {
  send ASK messages to $j \in C_i$;
  $WaitForReply_i \leftarrow C_i$; }}
wait for message receiving;

ASK message handler:
$asked_i \leftarrow$ **true**;
goto Management;

GOOD($s_j$,$u(s_j)$) message handler:
$S_{i,j} \leftarrow S_{i,j} \cup \{s_j\}$; record $u(s_j)$;
$WaitForReply_i \leftarrow WaitForReply_i \backslash \{j\}$;
goto Management;

VALUE($s_{pi}$) message handler:
select best assignment $(x_i, d_i^*)$ for $s_{pi}$;
send VALUE($s_{pi} \cup \{(x_i, d_i^*)\}$) messages to $j \in C_i$;
terminate;
```

Let $\delta_i(s)$ denote *local* sum of utilities.

$$\delta_i(s) = \sum\nolimits_k f_{k,i}(\{(x_k, d_k), (x_i, d_i)\}) \quad (4)$$
$$where\ \ k \in \{p_i\} \cup PP_i, \{(x_k, d_k), (x_i, d_i)\} \in s.$$

Boundary of utility for $s \in S_i$ is shown as follows.

$$UB_i(s) = \delta_i(s) + \sum\nolimits_{j \in C} UB_{i,j}(s) \quad (5)$$

$$LB_i(s) = \delta_i(s) + \sum\nolimits_{j \in C} LB_{i,j}(s) \quad (6)$$

Boundary of utility for $s \in S_i^-$ is shown as follows.

$$LB_i^-(s) = \max_{d_i \in D_i} \{LB_i(s \cup \{(x_i, d_i)\})\} \quad (7)$$

$$UB_i^-(s) = \max_{d_i \in D_i} \{UB_i(s \cup \{(x_i, d_i)\})\} \quad (8)$$

The sufficient condition to select *next* solution $s \in S_i^- \backslash S^{-sent}$, according to best-first ordering, is shown as follows.

$$\exists s \in S^- \backslash S^{-sent}, \forall s' \in S^- \backslash (S^{-sent} \cup \{s\}), \quad (9)$$
$$LB_i^-(s) \geq UB_i^-(s')$$

If a solution $s \in S^- \backslash S^{-sent}$ satisfies the condition, node i is able to send the solution s and utility $u(s) = LB_i^-(s)$ to p_i.

Algorithm 2: Synchronization control in message passing

```
Initiation:
$S_i^{-sent} \leftarrow \phi$; $asked_i \leftarrow$ **false**; $askedSolution_i \leftarrow$ **none**;

Management:
if $i$ is the root node $\vee$ $asked_i$ {
 if sufficient condition is satisfied {
  if $i$ is the root node {
   /* same as Algorithm 1 */ }
  else {
   select next best solution $s$;
   send GOOD($s$,$u(s)$) message to $p_i$;
   if $s \simeq askedSolution_i$ { $asked_i \leftarrow$ **false**; }}}
 else {
  select next best solution $s$;
  if ASK($s$) messages have not been sent {
   send ASK($s$) messages to $j \in C_i$; }}}
message receiving; goto Management;

ASK($s_{pi}$) message handler:
$asked_i \leftarrow$ **true**; $askedSolution_i \leftarrow s_{pi}$
goto Management;

GOOD($s_j$,$u(s_j)$) message handler:
$S_{i,j} \leftarrow S_{i,j} \cup \{s_j\}$; record $u(s_j)$;
goto Management;

VALUE($s_{pi}$) message handler:
/* same as Algorithm 1 */
```

In the root node r of pseudo tree, first best solution is optimal. When the first best solution is evaluated, node r is able to decide the optimal solution. Then, node r sends the optimal solution to child nodes. In other nodes, optimal solution is decided according to optimal solution of their parent nodes.

ODPOP[8] is a distributed constraint optimization algorithm which performs best-first search using pseudo-tree. Brief sketch of the ODPOP is shown in Algorithm 1 [1]. The algorithm consists of two phases. In first phase, utilities of partial solutions are computed in bottom up manner according to pseudo-tree. In second phase, optimal solution is decided in top down manner according to pseudo-tree. In this paper, we propose efficient methods based on the algorithm.

3 Proposed Method

In this section, several efficient methods for ODPOP based algorithms are proposed.

3.1 Derivation of partial solution

In bottom up computation of utility, each node has no knowledge about utilities between ancestor nodes. Each node i selects its *next* partial solution $s \in S_i^- \backslash S^{-sent}$ with utilities which are related to $\overline{PP}_i$. Utilities between ancestor nodes are estimated as zero. The estimated utilities are evaluated in ancestor nodes after GOOD message propagation. Such late evaluation may causes backtracking in higher ancestor node. The backtracking increases ASK/GOOD propagation.

[1]For convenience, description of algorithm is modified.

We apply derivation of *small* partial solution to reduce over estimation of utility. The small solution is compatible with original solution. However, the small solution does not include several value assignments about higher ancestor nodes. Therefore, it reduces backtracking in higher ancestor nodes.

The small solution is introduced for utility of *next* solution. Let s^* denote small solution for $u(s)$ of *next* solution s. s^* is a subset of $s' \in S^-$. Therefore s^* does not include several value assignment in s'. Let $\overline{S^*}$ denote a set of partial solutions which are not included in s^*. s^* must satisfy the conditions as follows.

$$\forall t \in \overline{S^*}, LB^-(s^* \cup t) \geq u(s) \tag{10}$$

To decide the assignments which are excluded from s, additional search processing is necessary. In this study, we limit the search range of variables in s, from highest ancestor node to Lth node.

Node i sends GOOD messages, which include small solution s^* and $u(s^*) = u(s)$, to its parent node p_i. Value assignments, which are not included in s^*, can be considered as *don't-care* variables. Therefore, s^* compatible with more number of solutions in ancestor nodes.

The small solution s^* includes a lesser number of value assignments, if utility $u(s^*)$ takes a lower value. Several small solutions in $S_{i,j}$, which have different utilities, may be overlapped in solution space. Therefore, we also use best-first ordering for evaluation of utility.

3.2 Synchronization control in message passing

If ODPOP algorithm is performed in synchronous manner, pairs of ASK/GOOD messages are exchanged between parent and child nodes. However, such strict message passing causes delay of processing. On the other hand, continuous sending of GOOD messages without any ASK messages, is not efficient.

We propose a heuristic method that parent node indicate next best solution in ASK messages. When node i send ask messages, node i select next best solution which has not been sent to p_i. The solution is sent to $c_j \in C_i$ with ASK messages. Each child node c_j registers the next solution. And each child node c_j continuously sends solutions until a solution, which is compatible with registered solution, has been sent. The processing is shown in Algorithm 2.

3.3 Error bounds of utility

In worst case, time complexity of best-first search and space complexity of dynamic programming, are exponen-

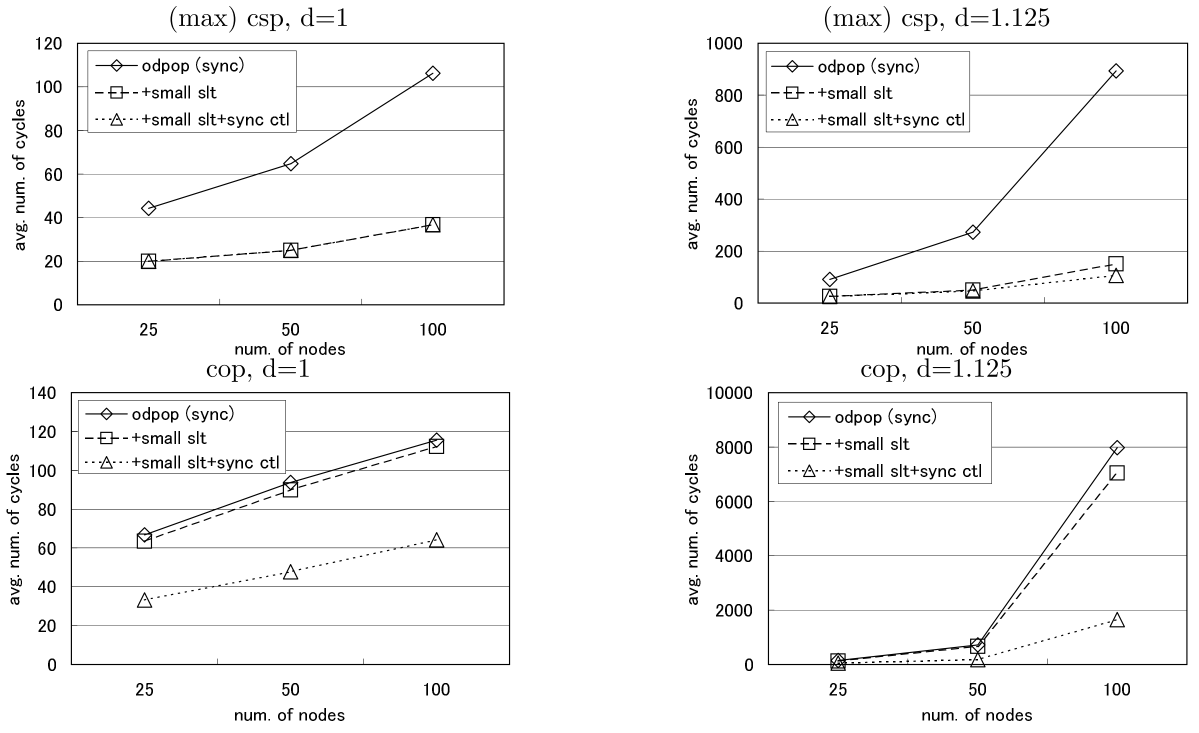

Figure 2: Average number of cycles

tial. To reduce computation and memory use, we apply error bounds of utility. Let b denote error bounds for each node. Solution $s^{*\prime}$ and utility $u'(s^{*\prime})$, which include error of utility, are defined as follows.

$$u(s^{*\prime}) \geq u'(s^{*\prime}) \geq \max\{u(s^{*\prime}) - b, 0\} \qquad (11)$$

The error bounds may reduce size of $s^{*\prime}$ and number of elements of $S_{i,j}$.

4 Property of proposed methods

Time complexity, space complexity and soundness of modified ODPOPs, which use proposed methods in subsection 3.1 and 3.2, are similar to original algorithm.

Termination of modified ODPOPs, which use proposed methods in subsection 3.1, 3.2 and 3.3, are similar to original algorithm.

5 Evaluation

We evaluated proposed methods using computational experiments. In the experiments, message passing and processing are simulated as follows.

1. Each node has message queues for sending and receiving message.
2. The simulator iterates message cycles. A message cycle consists of receive-process-send phase and exchange phase.
3. In receive-process-send phase, each node gets messages from receiving queue and process the messages. Then each node puts messages to sending queue, if it is necessary.
4. In exchange phase, messages in sending queue of source nodes are moved into receiving queue of destination nodes.

We use two types of graph coloring problems with three colors. In one type of problem, utility functions are set as a (maximum) constraint satisfaction problem ((max) csp). In the other type of problem it is set as constraint optimization problem (cop). Utility functions are set using random integer numbers between $[1, 10]$.

Scale of problem is decided using number of nodes n and link density d. Please note that these problems are rather *loose* constrained problems. Our purpose of the experiment is to evaluate the effect of proposed methods for best-first search.

Synchronized ODPOP and proposed methods are evaluated. Parameter L for Derivation of small partial solution is set to 2. Parameter b for error bounds are set to 1 or 2.

5.1 Effect of small solution and synchronization control

Average number of cycles are shown in figure 2. In the case of (max) csp (d=1), both proposed methods take least number of cycles. On the other hand, in case of cop

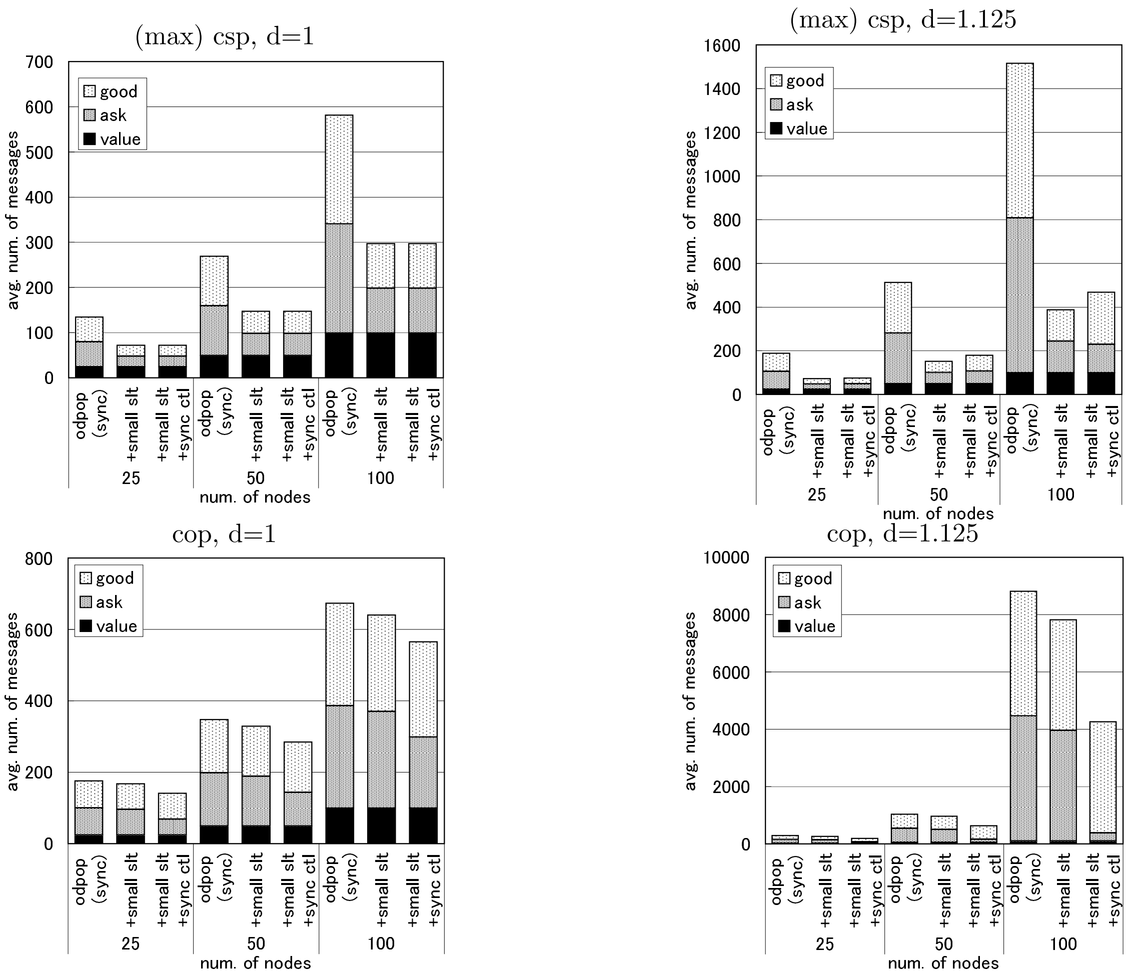

Figure 3: Average number of messages

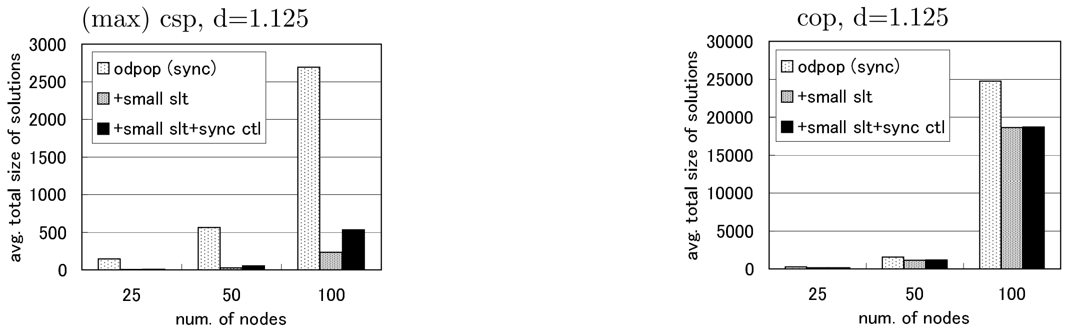

Figure 4: Total size of solutions

(d=1), only synchronization control takes less cycles than other methods. In the case of (max) csp (d=1.125), both proposed methods take less cycles than odpop (sync).

Total number of messages are shown in figure 3. In any case, proposed methods reduce total number of messages. In the case of (max) csp and cop, synchronization control reduces number of ASK messages. That increases number of GOOD messages. However, total number of messages are not significantly increased.

Total size of solutions, which are sent with GOOD messages, are shown in figure 4. The results show that small solution reduces the size of solutions.

5.2 Effect of error bounds

Results of experiments with error bounds are shown in Figure 5. In the case of cop(d=1.125), error bounds reduce number of cycles, number of messages and total size of utilities.

Maximum error of solutions are shown in Table 1. The results show that error ratio increases according to error bounds. However, the error ratio is less than upper limit

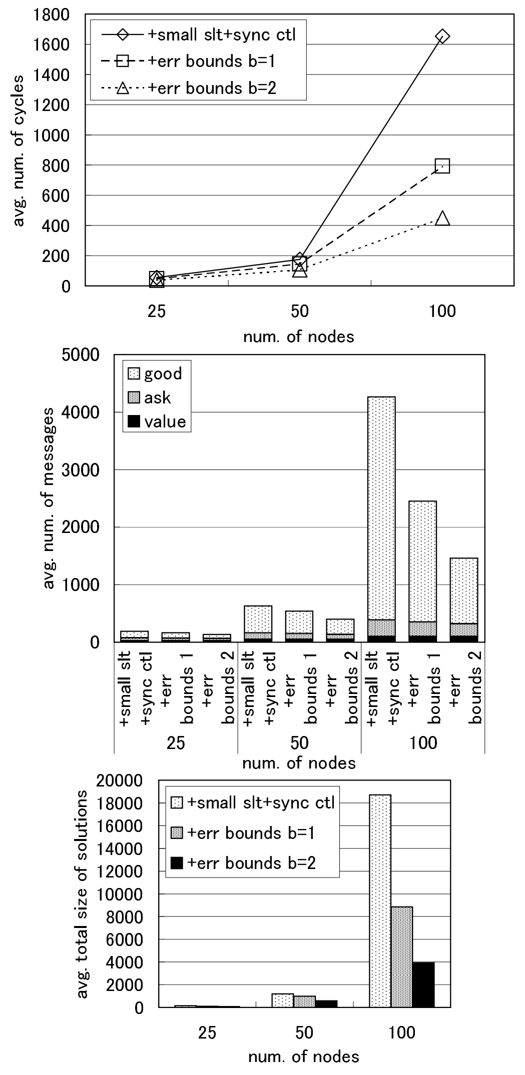

Figure 5: Using error bounds (cop, d=1.125)

Table 1: Maximum error of solutions (%)

problem	n	d	error bounds	
			1	2
(max)csp	25	1	0	0
		1.125	0.7	1.1
	50	1	0	0
		1.125	0.7	0.7
	100	1	0	0
		1.125	1.1	1.1
cop	25	1	0.1	0.5
		1.125	0.1	0.5
	50	1	0.3	1.0
		1.125	0.1	0.6
	100	1	0.2	0.8
		1.125	0.2	0.6

of error ratio. In the case of maxcsp, upper limit of error is equal to $n \times b$.

6 Conclusions and Future Work

In this paper, we proposed heuristic methods for pseudo-tree based distributed constraint optimization method. Derivation of partial solution is introduced to decrease number of backtracking among agents, Synchronization control method is applied to decrease number of communication message cycles. Error bounds is applied to obtain quasi optimal solution within less number of message cycles. Results of experiments show efficiency of proposed methods.

More detailed evaluation, integration with other algorithms and applying to practical problems are future work for the proposed methods.

References

[1] J. Liu and K. Sycara, "Exploiting problem structure for distributed constraint optimization", *Proc. 1st Int. Conf. on Multiagent Systems*, San Francisco, CA, USA, pp.246-253, 6/95.

[2] K. Hirayama and M. Yokoo, "Distributed Partial Constraint Satisfaction Problem", *Proc. 3rd Int. Conf. on Principles and Practice of Constraint Programming*,

[3] M. Lemaître and G. Verfaillie, "An Incomplete Method for Solving Distributed Valued Constraint Satisfaction Problems", *In Proc. AAAI97 Workshop on Constraints and Agents*, Providence, Rhode Island, 7/97.

[4] P. J. Modi, W. Shen, M. Tambe and M. Yokoo, "Adopt: asynchronous distributed constraint optimization with quality guarantees", *Artif. Intell.*, V161, N1-2, pp.149-180, 1/05.

[5] S. M. Ali, S. Koenig and M. Tambe, "Preprocessing techniques for accelerating the DCOP algorithm ADOPT", *Proc. 4rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, Utrecht, The Netherlands, pp.1041-1048, 7/05.

[6] A. Petcu and B. Faltings, "A distributed, complete method for multi-agent constraint optimization", *Proc. 5th Int. Workshop on Distributed Constraint Reasoning*, Toronto, Canada, pp.1041-1048, 9/04.

[7] Adrian Petcu, Boi Faltings, "A Scalable Method for Multiagent Constraint Optimization", *Proc. 9th Int. Joint Conf. on Artificial Intelligence*, Edinburgh, Scotland, pp.266-271, 8/05.

[8] A. Petcu and B. Faltings, "O-DPOP: An algorithm for Open/Distributed Constraint Optimization", *Proc. Int. Proceedings of the National Conference on Artificial Intelligence*, Boston, USA, pp.703-708, 7/06.