Pseudo-tree-based Incomplete Algorithm for Distributed Constraint Optimization with Quality Bounds

Tenda Okimoto, Yongjoon Joe, Atsushi Iwasaki, and Makoto Yokoo

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

Abstract. A Distributed Constraint Optimization Problem (DCOP) is a fundamental problem that can formalize various applications related to multi-agent cooperation. Since it is NP-hard, considering faster incomplete algorithms is necessary for large-scale applications. Most incomplete algorithms generally do not provide any guarantees on the quality of solutions. Some notable exceptions are DALO, the bounded max-sum algorithm, and ADPOP.

In this paper, we develop a new solution criterion called p-optimality and an incomplete algorithm for obtaining a p-optimal solution. The characteristics of this algorithm are as follows: (i) it can provide the upper bounds of the absolute/relative errors of the solution, which can be obtained a priori/a posteriori, respectively, (ii) it is based on a pseudo-tree, which is a widely used graph structure in complete DCOP algorithms, (iii) it is a one-shot type algorithm, which runs in polynomial-time in the number of agents n, and (iv) it has adjustable parameter p, so that agents can trade-off better solution quality against computational overhead. The evaluation results illustrate that this algorithm can obtain better quality solutions and bounds compared to existing bounded incomplete algorithms, while the run time of this algorithm is shorter.

1 Introduction

A Distributed Constraint Optimization Problem (DCOP) is a fundamental problem that can formalize various applications related to multi-agent cooperation. A DCOP consists of a set of agents, each of which needs to decide the value assignment of its variables so that the sum of the resulting rewards is maximized. Many application problems in multi-agent systems can be formalized as DCOPs, in particular, distributed resource allocation problems including distributed sensor networks [1] and meeting scheduling [2]. Various complete algorithms have been developed for finding globally optimal solution to DCOPs, e.g., DPOP [2], ADOPT [1], and OptAPO [3]. However, finding optimal DCOP solutions is NP-hard, so considering faster incomplete algorithms is necessary for large-scale applications. Various incomplete algorithms have been developed, e.g., DSA [4], MGM/DBA [5, 6], and ALS-DisCOP [7].

Most incomplete algorithms generally do not provide any guarantees on the quality of the solutions they compute. Notable exceptions are DALO [8], the bounded max-sum algorithm [9], and ADPOP [10]. Among these algorithms, DALO is unique since it can provide the bound of a solution a priori, i.e., the error bound is obtained before actually running the algorithm. Also, the obtained bound is independent of problem instances. On the other hand, the bounded max-sum algorithm and ADPOP can only provide the bound of a solution a posteriori, i.e., the error bound is obtained only after we actually run the algorithm and obtain an approximate solution. Having a priori bound is desirable, but a posteriori bound is usually more accurate.

In this paper, we develop an incomplete algorithm based on a new solution criterion called p-optimality. This algorithm can provide the upper bounds of the absolute/relative errors of the solution, which can be obtained a priori/a posteriori, respectively. These bounds are based on the induced width of a constraint graph and the maximal value of each reward function, but they are independent of problem instances. Induced width is a parameter that determines the complexity of many constraint optimization algorithms. This algorithm utilizes a graph structure called a pseudo-tree, which is widely used in complete DCOP algorithms such as ADOPT and DPOP. This algorithm can obtain an approximate solution with reasonable quality, while it is a one-shot type algorithm and runs in polynomial-time in the number of agents n. Thus, it is suitable for applications that need to obtain reasonable quality solutions (with quality guarantees) very quickly. Furthermore, in this algorithm, agents can adjust parameter p so that they can trade-off better solution quality against computational overhead.

DALO is an anytime algorithm based on the criteria of local optimality called k-size/t-distance optimality [8, 11] and has adjustable parameters k/t. Compared to this algorithm, our algorithm is a one-shot type algorithm, while DALO is an anytime algorithm, which repeatedly obtains new local optimal solutions until the deadline and returns the best solution obtained so far. Also, our algorithm can provide tighter bounds a priori. Furthermore, in our algorithm, the increase of computation/communication costs by increasing parameter p is more gradual compared to those for k-size/t-distance-optimality.

The bounded max-sum algorithm is a one-shot type algorithm. Compared to this algorithm, our algorithm has adjustable parameter p, while this algorithm has no adjustable parameter. Also, our algorithm can obtain a priori bound. Thus, agents can adjust parameter p before actually running the algorithm to obtain a solution with a desirable bound.

Our proposed algorithm is quite similar to ADPOP, which also eliminates edges among variables to bound the size of messages. ADPOP is also one-shot type algorithm and has an adjustable parameter. However, ADPOP uses a heuristic method to determine which edges to eliminate. As a result, it cannot obtain a priori bound. We can consider p-optimality gives a simple but theoretically well-founded method to determine which edges to eliminate in ADPOP.

Another advantage of our algorithm is that it can be used for a preprocessing phase before running a complete algorithm. Since our algorithm utilizes a pseudo-tree, it would be well-suited with pseudo-tree based complete algorithms.

The rest of this paper is organized as follows. Section 2 formalizes DCOP and provides basic terms related to the graphs. Section 3 introduces our incomplete algorithm and provides methods for estimating the error bound obtained by our algorithm. Section 4 evaluates the solution quality and the accuracy of the error bounds obtained by our algorithm. Section 5 concludes this paper.

2 Preliminaries

In this section, we briefly describe the formalization of Distributed Constraint Optimization Problems (DCOPs) and the basic terms for graphs.

Definition 1 (DCOP). *A distributed constraint optimization problem is defined by a set of agents S, a set of variables X, a set of binary constraint relations C, and a set of binary reward functions F. An agent i has its own variable x_i. A variable x_i takes its value from a finite, discrete domain D_i. A binary constraint relation (i, j) means there exists a constraint relation between x_i and x_j. For x_i and x_j, which have a constraint relation, the reward for an assignment $\{(x_i, d_i), (x_j, d_j)\}$ is defined by a binary reward function $r_{i,j}(d_i, d_j) : D_i \times D_j \rightarrow \mathbb{R}$. For a value assignment to all variables A, let us denote*

$$R(A) = \sum_{(i,j)\in C, \{(x_i,d_i),(x_j,d_j)\}\subseteq A} r_{i,j}(d_i, d_j).$$

Then, an optimal assignment A^* is given as $\arg\max_A R(A)$, i.e., A^* is an assignment that maximizes the sum of the value of all reward functions.

Since there exists a one-to-one relationship between an agent and its variable, for notation simplicity, we occasionally don't distinguish an agent and its variable. For example, we define a constraint optimization problem by a tuple $\langle X, C, F\rangle$, where X is a set of agents/variables. In this paper, we assume all reward values are non-negative and that the maximal value of each binary reward function is bounded, i.e., we assume $\forall i, \forall j$, where $(i, j) \in C$, $\forall d_i \in D_i, \forall d_j \in D_j, 0 \leq r_{i,j}(d_i, d_j) \leq r_{max}$ holds.

A DCOP problem can be represented using a constraint graph, in which a node represents an agent/variable and an edge represents a constraint.

Definition 2 (Constraint Graph). *For a distributed constraint optimization problem $\langle X, C, F\rangle$, we say $G = (X, C)$ as a constraint graph of $\langle X, C, F\rangle$. More specifically, a constraint graph is obtained by assuming each agent/variable as a node, and each binary constraint relation as an edge.*

In this paper, we consider subgraphs that are obtained by removing several edges/constraints from the original DCOP/constraint graph. We define a reward obtained in a subgraph as follows.

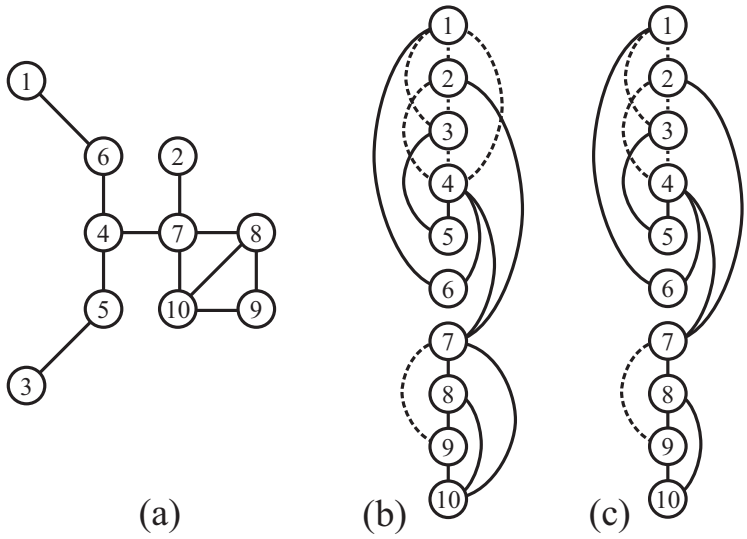

Fig. 1. (a) shows a constraint graph with ten nodes. (b) shows the induced chordal graph of (a) based on $o = 1 \prec \ldots \prec 10$. Induced width of (b) is three. (c) shows the subgraph of (b) obtained by removing edges $(1, 4)$ and $(7, 10)$. This graph is not chordal.

Definition 3 (Rewards in a subgraph). *For a distributed constraint optimization problem $\langle X, C, F\rangle$, its constraint graph $G = (X, C)$, and $G' = (X, C')$, which is a subgraph of G, i.e., $C' \subseteq C$, we define the rewards of an assignment A in subgraph $G' = (X, C')$ (denoted as $R_{C'}(A)$) as*

$$R_{C'}(A) = \sum_{(i,j)\in C', \{(x_i,d_i),(x_j,d_j)\}\subseteq A} r_{i,j}(d_i, d_j).$$

Let us introduce several basic terms for graphs.

Definition 4 (Undirected graph). *An undirected graph $G = (V, E)$ consists of a set of nodes $V = \{1, \ldots, n\}$ and a set of edges between nodes E. We denote $e \in E$ using nodes connected by the edge as $e = (i, j)$.*

Definition 5 (Connected graph). *We say an undirected graph $G = (V, E)$ is connected if any two nodes $i, j \in V$ are reachable via edges in E.*

In the rest of this paper, we assume a graph is connected.

Definition 6 (Neighboring nodes). *For a graph $G = (V, E)$ and a node $i \in V$, we call $Nb(E, i) = \{j \mid (i, j) \in E\}$ as i's neighboring nodes.*

Definition 7 (Total ordering among nodes). *A total ordering among nodes o is a permutation of a sequence of nodes $\langle 1, 2, \ldots, n\rangle$. We say node i precedes*

node j (denoted as $i \prec j$), if i occurs before j in o. We also denote ord(i) for the i-th node in a total ordering o.

Definition 8 (Ancestors). *For a graph $G = (V, E)$, a total ordering o, and a node $i \in V$, we call $A(E, o, i) = \{j \mid (i, j) \in E \ \wedge \ j \prec i\}$ as i's ancestors.*

Definition 9 (Chordal graph based on total ordering). *For a graph $G = (V, E)$ and a total ordering o, we say G is a chordal graph based on total ordering o when the following condition holds:*

- *$\forall i, \forall j, \forall k \in V$, if $j, k \in A(E, o, i)$, then $(j, k) \in E$.*

Definition 10 (Induced chordal graph based on total ordering). *For a graph $G = (V, E)$ and a total ordering o, we say a chordal graph $G' = (V, E')$ based on total ordering o, which is obtained by the following procedure, as an induced chordal graph [1] of G based on total ordering o.*

1. *Set E' to E.*
2. *Choose each node $i \in V$ from the last to the first based on o and apply the following procedure.*
   - *if $\exists j, \exists k \in A(E', o, i)$ s.t. $(j, k) \notin E'$, then set E' to $E' \cup \{(j, k)\}$.*
3. *Return $G' = (V, E')$.*

Next, we introduce a parameter called *induced width*, which can be used as a measure for checking how close a given graph is to a tree. For example, if the induced width of a graph is one, it is a tree. Also, the induced width of a complete graph with n variables is $n - 1$.

Definition 11 (Width based on total ordering). *For a graph $G = (V, E)$, a total ordering o, and a node $i \in V$, we call $|A(E, o, i)|$ as the width of node i based on total ordering o. Furthermore, we call $\max_{i \in V} |A(E, o, i)|$ as the width of graph G based on total ordering o and is denoted as $w(G, o)$.*

Definition 12 (Induced width based on total ordering). *For a graph $G = (V, E)$ and a total ordering o, we call $w(G', o)$ as the induced width of G based on total ordering o, where $G' = (V, E')$ is the induced chordal graph of G based on total ordering o.*

We show a simple example of the induced chordal graph of a graph and its induced width.

Example 1 (Induced width of induced chordal graph). Figure 1-(a) shows a constraint graph with ten nodes. (b) presents the induced chordal graph based on total ordering $o = 1 \prec \ldots \prec 10$. The ancestors of node 10 are nodes 7, 8, and 9. Since no edge exists between ancestors 7 and 9, edge $(7, 9)$ is added. Similarly, several new edges are added (shown as broken lines). The induced width of (b) is three.

[1] In constraint reasoning literature [12], such a graph is simply called an *induced* graph. However, the term *induced* is used in a more general meaning in graph theory. Thus, we use a more specific term, i.e., *induced chordal graph* in this paper.

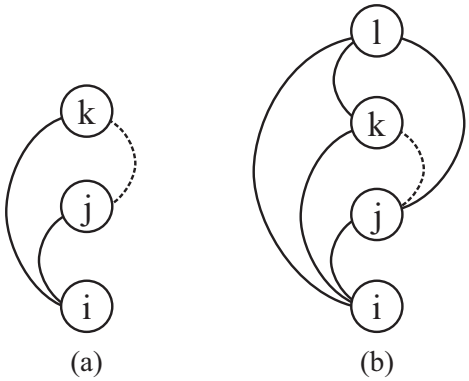

Fig. 2. (a) presents a part of p-reduced graph $G'' = (V, E'')$, where $j, k \in A(E'', o, i)$, and $(j, k) \notin E''$. (b) presents a situation where (j, k) is not j's first back-edge in G, i.e., there exists node l s.t. $l \prec k$, $(j, l) \in E$, and $(j, l) \notin E''$.

A pseudo-tree is a special graph structure, where a unique root node exists and each non-root node has a parent node.

Definition 13 (Pseudo-tree representation of chordal graph based on total ordering). *A chordal graph $G = (V, E)$ based on total ordering o can be assumed as a pseudo-tree as follows: (i) the node that appears first in o is the root node, and (ii) for each non-root node i, i's parent is node j, where $j \in A(E, o, i)$ and $\forall k \in A(E, o, i)$ and $k \neq j$, $k \prec j$ holds.*

Definition 14 (Back-edge). *When assuming a chordal graph $G = (V, E)$ based on total ordering o as a pseudo-tree, we say an edge (i, j) is a back-edge of i, if $j \in A(E, o, i)$ and j is not i's parent. Also, when $(i, j_1), (i, j_2), \ldots, (i, j_k)$ are all back-edges of i, and $j_1 \prec j_2 \prec \ldots \prec j_k$ holds, we call $(i, j_1), (i, j_2), \ldots,$ (i, j_k) as first back-edge, second back-edge, ..., k-th back-edge, respectively.*

Clearly, a node has at most $w(G, o) - 1$ back-edges.

3 Bounded Incomplete Algorithm based on Induced Width

In this section, we describe our new incomplete algorithm based on the induced width of a constraint graph. The basic idea of this algorithm is that we remove several edges from a constraint graph, so that the induced width of the remaining graph is bounded. Then we compute the optimal solution of the remaining graph, which is used as the approximate solution of the original graph.

3.1 Incomplete Algorithm and p-optimality

Our proposed incomplete algorithm has two phases:

Phase 1: Generate a subgraph from the induced chordal graph based on the total ordering by removing several edges, so that the induced width of the induced chordal graph obtained from the subgraph is bounded by parameter p.

Phase 2: Find an optimal solution to the graph obtained in Phase 1 using any complete DCOP algorithms.

First, let us describe Phase 1. Our goal is to obtain a subgraph so that the induced width of the induced chordal graph obtained from the subgraph equals p. At the same time, we want to bound the number of removed edges. This is not easy. One might imagine that we can easily obtain such a subgraph by just removing the back-edges so that all nodes have at most $p-1$ back-edges. However, by this simple method, we cannot guarantee that the remaining graph is chordal and we might need to add some edges to make it chordal. As a result, the induced width of the induced chordal graph can be more than p.

Let us show an example where the simple method does not work.

Example 2 (Simple method does not work). Figure 1-(c) presents the subgraph of (b) in Example 1. If we simply remove edges $(1,4)$ and $(7,10)$, each node has at most two edges with its ancestors (in (c)). However, the graph shown in (c) is not chordal, i.e., edge $(1,4)$ is missing, while there exist edges $(1,6)$ and $(4,6)$.

We develop a method for Phase 1 as follows. We call the obtained subgraph a *p-reduced graph*.

Definition 15 (p-reduced graph). *For a induced chordal graph $G=(V,E)$ based on total ordering o, we say a graph $G'=(V,E')$ obtained by the following procedure as p-reduced graph of G (where $1 \leq p \leq w(G,o)$):*

1. *Set E' to E.*
2. *Repeat the following procedure $w(G,o)-p$ times*
   - *For each $i \in V$ where $p+1 \leq ord(i) \leq w(G,o)$ remove the first back-edge in $G'=(V,E')$ from E' if there is one.*
3. *Return $G'=(V,E')$.*

Assuming that the agents know the pseudo-tree among them, running this procedure by these agents is quite simple. For obtaining the p-reduced graph, each agent i ($p+1 \leq ord(i) \leq w(G,o)$) simply removes its first back-edge, second back-edge, ..., $(w(G,o)-p)$-th back-edge.

Theorem 1. *For a induced chordal graph $G=(V,E)$ based on total ordering o, for any $1 \leq p \leq w(G,o)$, and G's p-reduced graph $G'=(V,E')$, the following conditions hold:*

1. *G' is a chordal graph based on total ordering o.*

2. $w(G', o)$ *is* p*.*

Proof. When obtaining p-reduced graph G', for each node i ($p+1 \leq ord(i) \leq w(G,o)$), its first back-edge is repeatedly removed $w(G,o) - p$ times. Since the number of back-edges is at most $w(G,o)-1$, the number of remaining back-edges is at most $p-1$. Also, there exists at least one node who has exactly $w(G,o)-1$ back-edges. Thus, since the remaining back-edges for the node are $p-1$, $w(G', o)$, i.e., the width of G' based on o, is p.

Next, we show that G' is a chordal graph based on total ordering o. Since p-reduced graph G' is obtained by repeatedly removing first back-edges for each node, it suffices to show that graph $G'' = (V, E'')$, which is obtained by removing first back-edges for each node in G, is a chordal graph based on total ordering o. We prove this fact by contradiction, i.e., we derive a contradiction by assuming that $\exists i \in V, \exists j, \exists k \in A(E'', o, i)$, s.t., $(j,k) \notin E''$. Without loss of generality, we can assume $k \prec j$ (Fig. 2-(a)).

Since $G = (V, E)$ is a chordal graph based on total ordering o, $(j,k) \in E$ holds. Furthermore, since $(j,k) \notin E''$, (j,k) must be the first back-edge of j in G. Also, since $k \in A(E'', o, i)$, $(i,k) \in E''$ holds. Thus, there exists node l s.t. $l \prec k$, $(i,l) \in E$, and $(i,l) \notin E''$ holds, i.e., (i,l) is i's first back-edge in G and is removed in G''. Furthermore, since $G = (V, E)$ is a chordal graph based on total ordering o, and $(i,l) \in E$ and $(i,j) \in E$ hold, $(j,l) \in E$ must hold (Fig. 2-(b)). However, since $l \prec k$, (j,k) cannot be j's first back-edge in G. This is a contradiction. Thus, $G'' = (V, E'')$ must be a chordal graph based on total ordering o.

We introduce a new criterion for approximated solutions.

Definition 16 (p-optimality). *We say an assignment A is p-optimal for a distributed constraint optimization problem $\langle X, C, F \rangle$ and a total ordering o, when A maximizes the total rewards in $G'' = (X, C'')$, where $G' = (X, C')$ is an induced chordal graph of $G = (X, C)$ based on total ordering o, and $G'' = (X, C'')$ is the p-reduced graph of G'. More specifically, $\forall A', R_{C''}(A) \geq R_{C''}(A')$ holds.*

Next, let us describe Phase 2. To find a p-optimal solution, we can use any complete DCOP algorithms. We use the obtained p-optimal solution as an approximate solution of the original graph. In particular, since we already obtained a pseudo-tree whose induced width is bounded, using pseudo-tree-based DCOP algorithms would be convenient.

3.2 Quality Guarantees

We provide two methods for estimating the error of the solution obtained by our algorithm. One method estimates absolute error which can be obtained a priori, i.e., agents can obtain the estimate before obtaining an approximate solution. Thus, agents can choose parameter p based on the estimation before actually obtaining an approximate solution.

Theorem 2. *For a distributed constraint optimization problem $\langle X, C, F\rangle$, its constraint graph $G = (X, C)$, and a total ordering o, if A is p-optimal, then the following condition holds among $R(A^*)$ and $R(A)$, where A^* is an optimal assignment:*

$$R(A^*) - R(A) \leq r_{max} \times \sum_{k=1}^{w(G,o)-p} (|X| - (k+1))$$

Proof. Since A is p-optimal, for $G' = (X, C')$, which is an induced chordal graph of G based on total ordering o, and $G'' = (X, C'')$, which is a p-reduced graph of G', the following condition holds:

$$R_{C''}(A^*) \leq R_{C''}(A).$$

Furthermore, C' is obtained by adding edges to C, and C'' is obtained from C' by removing at most $\sum_{k=1}^{w(G,o)-p}(|X| - (k+1))$ edges. Since the maximal reward of each edge is bounded by r_{max}, the following condition holds:

$$R(A^*) \leq R_{C''}(A^*) + r_{max} \times \sum_{k=1}^{w(G,o)-p} (|X| - (k+1)).$$

Furthermore, it is clear that the following condition holds:

$$R_{C''}(A) \leq R(A).$$

Thus, we obtain

$$R(A^*) - R(A) \leq r_{max} \times \sum_{k=1}^{w(G,o)-p} (|X| - (k+1)).$$

Intuitively, the absolute error is given by the product of r_{max} and the maximal number of removed back-edges.

Furthermore, we can compute the upper bound of the relative error using a method similar to ADPOP [10]. Note that this error bound can be obtained only a posteriori, i.e., we first need to obtain an approximate solution, then, we know the upper-bound of the relative error. Intuitively, if we remove a back-edge connecting i and j, we add an edge that connects i and j', where j' is a copy of j but it is connected only to i and has no unary reward. If we add an equality constraint between j and j', this problem is equivalent to the original problem. By ignoring such a constraint, we obtain a relaxed problem. Note that the induced width of this relaxed problem is p. This method, which ignores some dependencies among variables, is similar to minibucket elimination scheme [12].

4 Experimental Evaluation

In this section, we evaluate the solution quality and the accuracy of the error bounds obtained by our algorithm and show comparisons with DALO-t [13] and

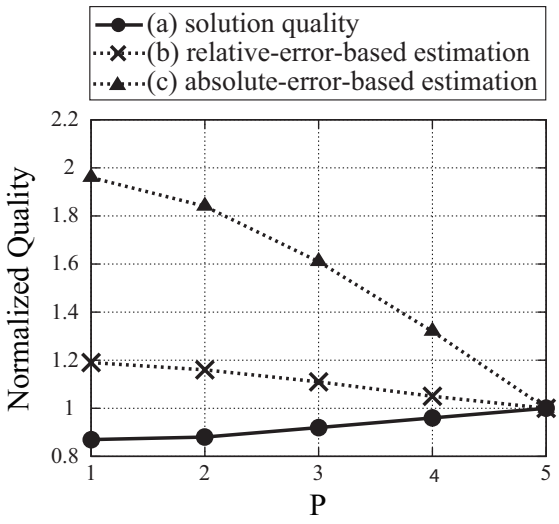

Fig. 3. (a), (b), and (c) in p-optimal algorithm for graphs with 20 nodes, induced width 5, and density 0.4. Value closer to 1 is desirable.

the bounded max-sum algorithm [9]. In our evaluations, we use the following problem instances. The domain size of each variable is three, and the reward of each binary constraint is in the range [0,. . . ,99]. Each data point in a graph represents an average of 30 problem instances. We generate random graphs with a fixed induced width. For Phase 2 of our p-optimal algorithm, we use the DPOP algorithm with FRODO [14](version 2.7.1). For comparison, we use the DALO-t algorithm that obtains t-distance-optimal solutions, since [8] shows that the error bounds for t-distance-optimality are usually better than that for k-size optimality. In our comparison, we mostly use settings p=1 and t=1.

First, we show (a) the quality of an obtained solution, (b) the estimated quality of an optimal solution based on the relative error bound, and (c) the estimated quality of an optimal solution based on the absolute error bound for the p-optimal algorithm. The results of (a), (b), and (c) are normalized by the quality of an actual optimal solution, where (a) should be less than 1, and (b) and (c) should be more than 1. For all of them, a value closer to 1 is desirable. Figure 3 shows these values for graphs with 20 nodes, induced width 5, and the density of the binary constraints 0.4. We vary parameter p from 1 to 5. Note that when we set the number of nodes to 20 and the induced width to 5, we cannot create a graph whose density is greater than 0.6. We can see that the obtained solution quality and estimation are reasonable for most cases, except that (c) becomes rather inaccurate when $p = 1$. This is because the number of removed edges is large. In such a case, we need to increase p to obtain a better estimation.

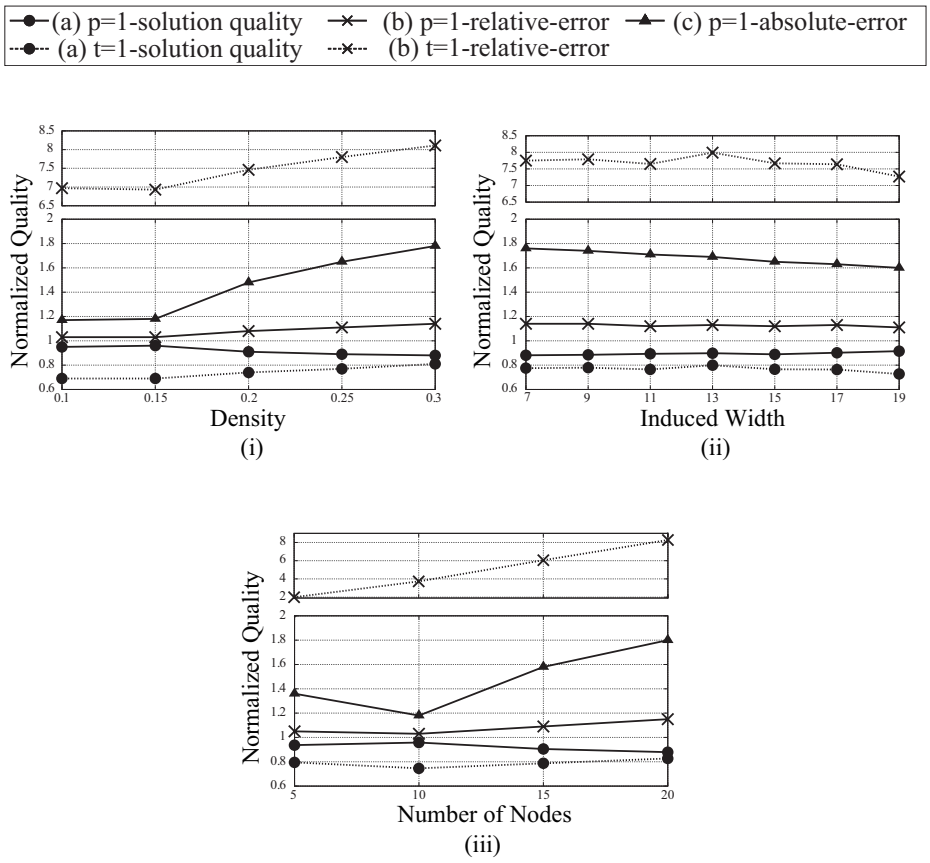

Fig. 4. (a), (b), and (c) in p=1-optimal algorithm and DALO-t=1 for graphs with (i) 20 nodes, induced width 5, (ii) 20 nodes, density 0.3, and (iii) density 0.3, induced width 3. Broken line indicates results for DALO-t=1. Value closer to 1 is desirable.

Next, we compare our algorithm for p=1-optimality and DALO-t=1 for t=1-distance-optimality. Usually, DALO-t is used as an anytime algorithm, i.e., it continuously obtains t-optimal solutions. In this paper, we stop DALO-t when the first t-optimal solution is found. Figure 4(i) shows (a), (b), and (c) in the p=1-optimal algorithm and in DALO-t=1 for graphs with 20 nodes and induced width 5, varying the density. A value closer to 1 is desirable. The broken lines indicate the results for DALO-t=1. We can see (a), (b), and (c) are better/more accurate in the p=1-optimal algorithm compared with DALO-t=1. Results (b) and (c) for the p=1-optimal algorithm become less accurate when the density increases. This is because the number of removed edges becomes large in the high density region. Figure 4(ii) shows the results for graphs with 20 nodes and density 0.3, varying the induced width. We can see even the induced width is

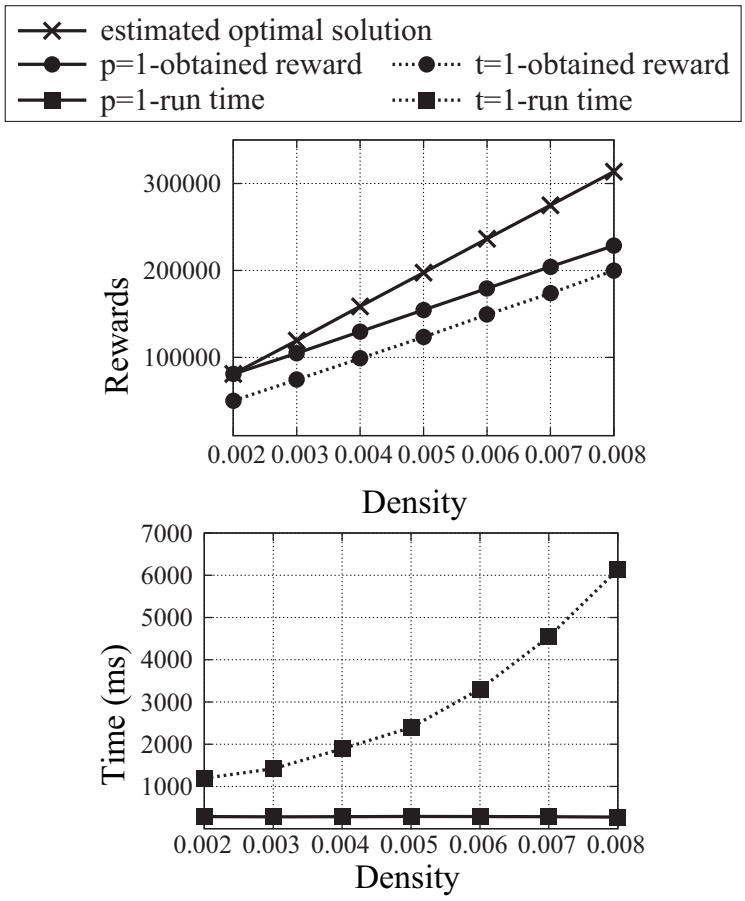

Fig. 5. Obtained rewards (not normalized) and run time (ms) for graphs with 1000 nodes and induced width 5. Broken line indicates results for DALO-t=1.

increased, (a), (b), and (c) for p=1-optimal algorithm are better/more accurate compared with DALO-t=1. Figure 4(iii) shows the results for graphs with density 0.3 and induced width 3, varying the number of nodes. The obtained results are similar to Fig. 4(i), i.e., (b) and (c) for the p=1-optimal algorithm become less accurate when the number of nodes increases.

Moreover, we show the results for large-scale problem instances. For them, obtaining an optimal solution is infeasible. Figure 5 shows the results for graphs with 1000 nodes and induced width 5, varying the density. Since we cannot obtain optimal solutions for these problem instances, we show the values of the obtained reward (which are not normalized). By setting the induced width to 5, we cannot create a graph whose density is greater than 0.01. We can see the

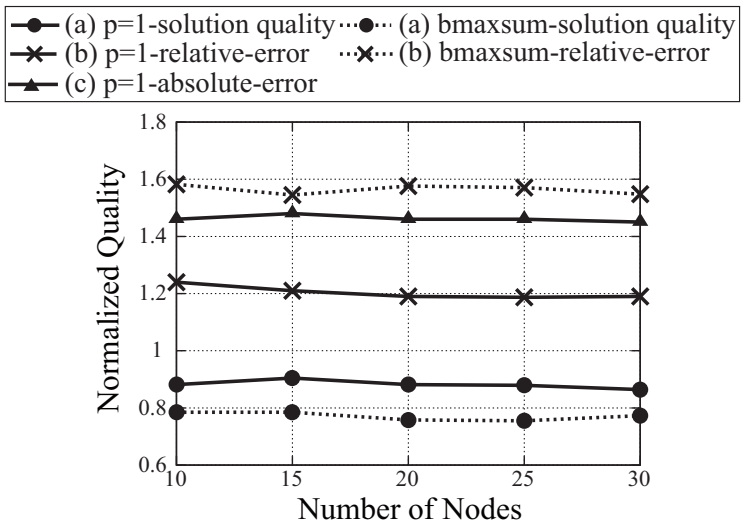

Fig. 6. (a), (b), and (c) in p=1-optimal algorithm and bounded max-sum algorithm for graphs with induced width 2. Broken line indicates results for bounded max-sum (bmaxsum) algorithm. Value closer to 1 is desirable.

rewards/run time for the p=1-optimal algorithm are greater/shorter compared to those for DALO-t=1.

Finally, we compare our algorithm for p=1-optimality and the bounded max-sum (bmaxsum) algorithm. We used graph coloring problems in the same settings presented in [9], except that the reward of each binary constraint is in the range [0,. . . ,6]. Figure 6 shows the results for graphs with induced width 2, varying the number of nodes. A value closer to 1 is desirable. Broken lines indicate the results for the bounded max-sum algorithm. We can see (a) and (b) (also (c)) are better in the p=1-optimal algorithm compared with the bounded max-sum algorithm.

One might imagine that the relative error of bounded-max-sum can be higher than the absolute error of p-optimality with p=1, since bounded max-sum algorithm uses some information about the real cost of removed edges. Since our algorithm and the bounded maxsum algorithm use different graph structures (i.e., a standard constraint graph and a factor graph, respectively), we cannot simply say that an informed/heuristic method for eliminating edges should work better. Our speculation is that the set of eliminated edges chosen by the bounded maxsum algorithm is somewhat far away from an optimal overall choice, since the elimination method is greedy/local.

We show the results for large-scale problem instances (graphs with 1000 nodes and induced width 2) in Fig. 7. Similar to the problem instances used in Fig. 5, obtaining an optimal solution is infeasible for these problem instances. By setting

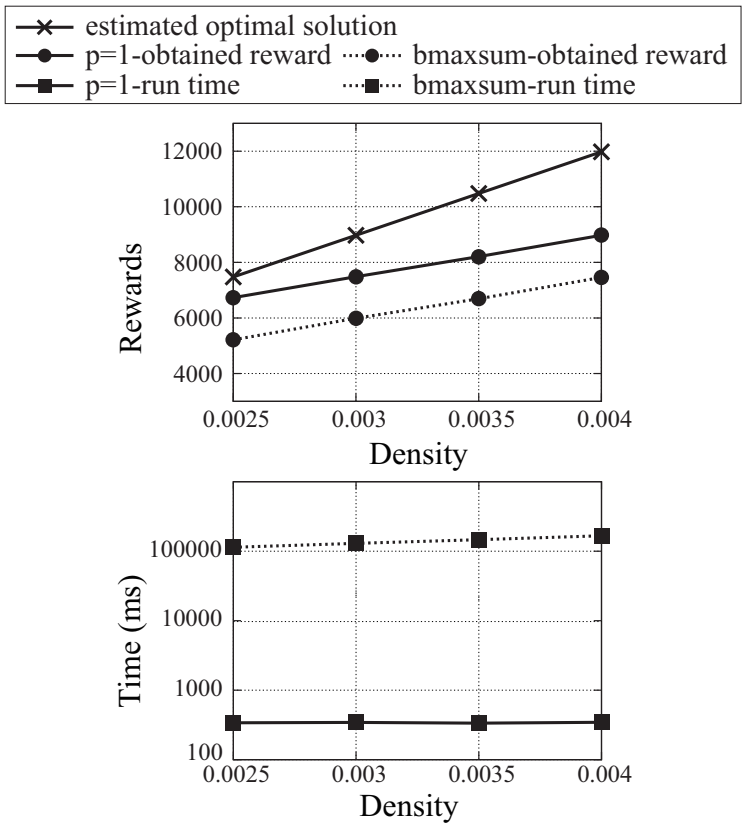

Fig. 7. Obtained rewards (not normalized) and run time (ms) for graphs with 1000 nodes and induced width 2. Broken line indicates results for bounded max-sum (bmax-sum) algorithm.

the induced width to 2, we cannot create a graph whose density is greater than 0.004. We show the values of the obtained reward (which are not normalized) as in Fig. 5. We can see the rewards/run time for the p=1-optimal algorithm are greater/shorter compared to those for the bounded max-sum algorithm.

In summary, these experimental results reveal that (i) the quality of the obtained solution of the p=1-optimal algorithm is much better compared with DALO-t=1 and bounded max-sum algorithms, (ii) the estimated quality of an optimal solution based on the absolute/relative error bounds for the p=1-optimal algorithm is more accurate than the other algorithms, and (iii) the run time of our algorithm is much shorter.

Although we did not show the results of ADPOP, they are basically similar to our algorithms, since these two algorithms differ only in the methods to de-

termine which edges to eliminate. The advantage of our algorithm is that it can provide the bound of a solution a priori.

Let us speculate why our algorithm can obtain better results compared to DALO-t and the bounded max-sum algorithm. These algorithms obtain approximate solutions of the original problem, while our algorithm obtains an *optimal* solution for a relaxed problem. If the relaxed problem is not so different from the original problem (e.g., the induced width is small), our algorithm can find a better solution quickly.

It must be mentioned that we require knowledge of the induced width and r_{max} to obtain a priori bound based on p-optimality. On the other hand, the error bound obtained by k/t-optimality is independent from problem instances. If r_{max} can be extremely large, while the average of the binary rewards is rather small compared to r_{max}, the absolute error bound of p-optimality becomes less informative.

5 Conclusion

We developed a new solution criterion called p-optimality and an incomplete algorithm for obtaining a p-optimal solution. This algorithm utilizes a graph structure called a pseudo-tree, which is widely used in complete DCOP algorithms. We provided the upper bounds of the absolute/relative errors of the solution, which can be obtained a priori/a posteriori, respectively. We showed that our algorithm for p=1-optimality can obtain better quality solutions and estimate more accurate error bounds compared with DALO-t for t=1-distance-optimality and the bounded max-sum algorithm. Furthermore, we showed that the run time for our algorithm for p=1-optimality is much shorter compared to these existing algorithms. Our future works include developing an anytime/complete algorithm that utilizes our algorithm as a preprocessing phase. A similar idea, i.e., using ADPOP as a preprocessing for ADOPT, is presented in [15].

References

[1] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[2] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 266–271, 2005.

[3] Roger Mailler and Victor Lesser. Using cooperative mediation to solve distributed constraint satisfaction problems. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, pages 446–453, 2004.

[4] Stephen Fitzpatrick and Lambert Meertens. Distributed coordination through anarchic optimization. In Victor Lesser, Charles Ortiz, and Milind Tambe, editors, *Distributed Sensor Networks: A Multiagent Perspective*, pages 257–295. Kluwer Academic Publishers, 2003.

[5] Jonathan Pearce, Milind Tambe, and Rajiv Maheswaran. Solving multiagent networks using distributed constraint optimization. In *AI Magazine, 29(3)*, pages 47–66, 2008.

[6] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, 2005.

[7] Roie Zivan. Anytime local search for distributed constraint optimization. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 1449–1452, 2008.

[8] Christopher Kiekintveld, Zhengyu Yin, Atul Kumar, and Milind Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 133–140, 2010.

[9] Alessandro Farinelli, Alex Rogers, and Nicholas Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *Proceedings of the 12th International Workshop on Distributed Constraint Reasoning*, pages 46–59, 2009.

[10] Adrian Petcu and Boi Faltings. Approximations in distributed optimization. In *Proceedings of the Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming*, pages 802–806, 2005.

[11] Jonathan Pearce and Milind Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1446–1451, 2007.

[12] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

[13] Zhengyu Yin. USC dcop repository. University of Southern California, Department of Computer Science, 2008.

[14] Thomas Léauté, Brammert Ottens, and Radoslaw Szymanek. FRODO 2.0: An open-source framework for distributed constraint optimization. In *Proceedings of the 12th International Workshop on Distributed Constraint Reasoning*, pages 160–164, 2009.

[15] James Atlas, Matt Warner, and Keith Decker. A memory bounded hybrid approach to distributed constraint optimization. In *Proceedings of the 11th International Workshop on Distributed Constraint Reasoning*, pages 37–51, 2008.