HS-CAI: A Hybrid DCOP Algorithm via Combining Search with Context-Based Inference

Dingding Chen,[1] **Yanchen Deng,**[2] **Ziyu Chen,**[1,*] **Wenxing Zhang,**[1] **Zhongshi He**[1]
[1]College of Computer Science, Chongqing University, Chongqing, China
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
{dingding, chenziyu, zshe}@cqu.edu.cn, ycdeng@ntu.edu.sg, wenxinzhang18@163.com

Abstract

Search and inference are two main strategies for optimally solving Distributed Constraint Optimization Problems (DCOPs). Recently, several algorithms were proposed to combine their advantages. Unfortunately, such algorithms only use an approximated inference as a one-shot preprocessing phase to construct the initial lower bounds which lead to inefficient pruning under the limited memory budget. On the other hand, iterative inference algorithms (e.g., MB-DPOP) perform a context-based complete inference for all possible contexts but suffer from tremendous traffic overheads. In this paper, (i) hybridizing search with context-based inference, we propose a complete algorithm for DCOPs, named HS-CAI where the inference utilizes the contexts derived from the search process to establish tight lower bounds while the search uses such bounds for efficient pruning and thereby reduces contexts for the inference. Furthermore, (ii) we introduce a context evaluation mechanism to select the context patterns for the inference to further reduce the overheads incurred by iterative inferences. Finally, (iii) we prove the correctness of our algorithm and the experimental results demonstrate its superiority over the state-of-the-art.

Introduction

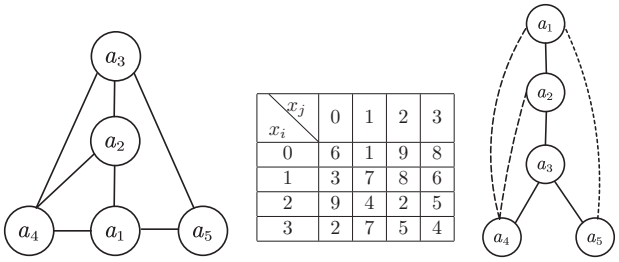
Distributed Constraint Optimization Problems (DCOPs) (Hirayama and Yokoo 1997; Fioretto, Pontelli, and Yeoh 2018) are an elegant model for representing Multi-Agent Systems (MAS) where agents coordinate with each other to optimize a global objective. Due to their ability to capture essential MAS aspects, DCOPs can formalize various applications in the real world such as sensor network (Farinelli, Rogers, and Jennings 2014), task scheduling (Maheswaran et al. 2004; Fioretto, Yeoh, and Pontelli 2017), smart grid (Fioretto et al. 2017) and so on.

Incomplete algorithms for DCOPs (Zhang et al. 2005; Maheswaran, Pearce, and Tambe 2006; Farinelli et al. 2008; Ottens, Dimitrakakis, and Faltings 2017) aim to rapidly find solutions at the cost of sacrificing optimality. On the contrary, complete algorithms guarantee the optimal solution and can be generally classified into inference-based and search-based algorithms. DPOP (Petcu and Faltings 2005b) and Action_GDL (Vinyals, Rodriguez-Aguilar, and Cerquides 2009) are typical inference-based complete algorithms which employ a dynamic programming paradigm to solve DCOPs. However, they require a linear number of messages of exponential size with respect to the induced width. Accordingly, ODPOP (Petcu and Faltings 2006) and MB-DPOP (Petcu and Faltings 2007) were proposed to trade the message number for smaller memory consumption by propagating the dimension-limited utilities with the corresponding contexts iteratively. That is, they iteratively perform a context-based inference to solve DCOPs optimally when the memory budget is limited.

Search-based complete algorithms like SBB (Hirayama and Yokoo 1997), AFB (Gershman, Meisels, and Zivan 2009), PT-FB (Litov and Meisels 2017), ADOPT (Modi et al. 2005) and its variants (Yeoh, Felner, and Koenig 2010; Gutierrez, Meseguer, and Yeoh 2011) perform distributed backtrack searches to exhaust the search space. They have a linear size of messages but an exponential number of messages. Furthermore, these algorithms only use local knowledge to update the lower bounds, which exerts a trivial effect on pruning and makes them infeasible for solving large-scale problems. Then, PT-ISABB (Deng et al. 2019), DJAO (Kim and Lesser 2014) and ADPOT-BDP (Atlas, Warner, and Decker 2008) came out to attempt to hybridize search with inference, where an approximated inference is used to construct the initial lower bounds for the search process. More specifically, PT-ISABB and ADOPT-BDP use ADPOP (Petcu and Faltings 2005a) to establish the lower bounds, while DJAO employs a function filtering technique (Brito and Meseguer 2010) to get them. Here, ADPOP is an approximate version of DPOP by dropping the exceeding dimensions to ensure that each message size is below the memory limit. However, given the limited memory budget, the lower bounds obtained in a one-shot preprocessing phase are still inefficient for pruning since such bounds cannot be tightened by considering the running contexts. That is, the existing hybrid algorithms use only a context-free approximated inference as a one-shot phase to construct the initial lower bounds.

In this paper, we investigate the possibility of combining

*Corresponding author.



x_i \ x_j	0	1	2	3
0	6	1	9	8
1	3	7	8	6
2	9	4	2	5
3	2	7	5	4

(a) Constraint graph (b) Constraint matrix (c) Pseudo tree

Figure 1: An example of a DCOP and its pseudo tree

search with context-based inference to solve DCOPs efficiently. Specifically, our main contributions are listed as follows:

- We propose a novel complete DCOP algorithm by hybridizing search with context-based inference, called HS-CAI where the search adopts a tree-based SBB to find the optimal solution and provides contexts for the inference, while the inference iteratively performs utility propagation for these contexts to construct the tight lower bounds to speed up the search process.
- We introduce a context evaluation mechanism to extract the context patterns for the inference from the contexts derived from the search process so as to further reduce the number of context-based inferences.
- We theoretically show the completeness of HS-CAI and prove that the lower bounds produced by the context-based inference are at least as tight as the ones established by the context-free approximated inference under the same memory budget. Moreover, the experimental results demonstrate HS-CAI outperforms state-of-the-art complete DCOP algorithms.

Background

In this section, we expound the preliminaries including DCOPs, pseudo tree, MB-DPOP and ODPOP.

Distributed Constraint Optimization Problems

A distributed constraint optimization problem (Modi et al. 2005) can be defined by a tuple $\langle A, X, D, F\rangle$ where

- $A = \{a_1, a_2, \ldots, a_n\}$ is a set of agents.
- $X = \{x_1, x_2, \ldots, x_m\}$ is a set of variables.
- $D = \{D_1, D_2, \ldots, D_m\}$ is a set of finite, discrete domains. Each variable x_i takes a value in D_i.
- $F = \{f_1, f_2, \ldots, f_q\}$ is a set of constraint functions. Each function $f_i : D_{i1} \times \cdots \times D_{ik} \to \mathbb{R}_{\geq 0}$ specifies the non-negative cost for each combination of $x_{i1}, \cdots, x_{ik}$.

For the sake of simplicity, we assume that each agent holds exactly one variable (and thus the term *agent* and *variable* could be used interchangeably) and all constraints are binary (i.e., $f_{ij} : D_i \times D_j \to \mathbb{R}_{\geq 0}$). A solution to a DCOP is the assignments to all the variables such that the total cost is minimized. That is,

$$X^* = \underset{d_i \in D_i, d_j \in D_j}{\arg\min} \sum_{f_{ij} \in F} f_{ij}(x_i = d_i, x_j = d_j)$$

A DCOP can be represented by a constraint graph where a vertex denotes a variable and an edge denotes a constraint. Fig. 1 (a) presents a DCOP with five variables and seven constraints. For simplicity, the domain size of each variable is four and all constraints are identical as shown in Fig. 1(b).

Pseudo Tree

A depth first search (Freuder and Quinn 1985; Dechter, Cohen, and others 2003) arrangement of a constraint graph is a pseudo tree with the property that different branches are independent, and categorizes its constraints into tree edges and pseudo edges (i.e., non-tree edges). Thus, the neighbors of agent a_i can be classified into its parent $P(a_i)$, children $C(a_i)$, pseudo parents $PP(a_i)$ and pseudo children $PC(a_i)$ based on their positions in the pseudo tree and the type edges they connect with a_i. For clarity, we denote all the (pseudo) parents of a_i as $AP(a_i) = PP(a_i) \cup \{P(a_i)\}$ and the set of ancestors who share constraints with a_i and its descendants as its separators $Sep(a_i)$ (Petcu and Faltings 2005b). Fig. 1(c) presents a possible pseudo tree deriving from Fig. 1(a).

MB-DPOP and ODPOP

MB-DPOP and ODPOP apply an iterative context-based utility propagation to aggregate the optimal global utility. Specifically, MB-DPOP first uses a cycle-cuts idea (Dechter, Cohen, and others 2003) on a pseudo tree to determine cycle-cut variables and groups these variables into clusters. Within the cluster, MB-DPOP preforms a bounded-memory exploration; anywhere else, the utility propagation from DPOP applies. Specifically, agents in each cluster propagate memory-bounded utilities for all the contexts of cycle-cut variables. As for ODPOP, each agent adopts an incremental and best-first fashion to propagate the context-based utility. Specifically, an agent repeatedly asks its children for their suggested context-based utilities until it can calculate a suggested utility for its parent during the utility propagation phase. The phase finishes after the root agent receiving enough utilities to determine the optimal global utility.

Proposed Method

In this section, we present a novel complete DCOP algorithm which utilizes both search and inference interleaved.

Motivation

It can be concluded that search can exploit bounds to prune the solution space but the pruning efficiency is closely related to the tightness of bounds. However, most of search-based complete algorithms can only use local knowledge to compute the initial lower bounds, which leads to inefficient pruning. On the other hand, inference-based complete algorithms can aggregate the global utility promptly, but their memory consumption is exponential in the induced width. Therefore, it is natural to combine both search and inference

by performing memory-bounded inferences to construct efficient lower bounds for search. Unfortunately, the existing hybrid algorithms only perform an approximated inference to construct one-shot bounds in the preprocessing phase, which would lead to inefficient pruning given a limited memory budget. In fact, the bounds can be tightened by the context-based inference. That is, instead of dropping a set of dimensions, named *approximated* dimensions, to stay below the memory budget, we explicitly consider the running-context assignments to a subset of approximated dimensions (i.e., the context patterns) and compute tight bounds w.r.t. the context patterns. Here, we denote an assigned subset of approximated dimensions as *decimated* dimensions.

More specifically, we aim to combine the advantages of search and context-based inference to optimally solve DCOPs. Different from the existing hybrid methods, we compute tight bounds for the running contexts by performing the context-based inference for the context patterns chosen from the contexts derived from the search process.

Proposed Algorithm

Now, we present our proposed algorithm which consists of a preprocessing phase and a hybrid phase.

Preprocessing Phase performs a bottom-up dimension and utility propagation to accumulate the approximated dimensions and establish the initial lower bounds based on these propagated utilities for search. Accordingly, we employ a tailored version of ADPOP with the limit k to propagate the approximated dimensions and incomplete utilities (we omit the pseudo code of this phase due to the limited space). Particularly, during the propagation process, each agent a_i selects the dimensions S_i of its highest ancestors to approximate to make the dimension size of each outgoing utility below k. Then, the approximated dimensions $SList_i$ (i.e., the dimensions approximated by a_i and its descendants) and utility $preUtil^i_{P(a_i)}$ sent from a_i to its parent can be computed as follows:

$$SList_i = S_i \cup (\bigcup_{a_c \in C(a_i)} SList^c_i) \tag{1}$$

$$preUtil^i_{P(a_i)} = \min_{S_i \cup \{x_i\}} (localUtil_i \otimes (\bigotimes_{a_c \in C(a_i)} preUtil^c_i)) \tag{2}$$

Here, $SList^c_i$ and $preUtil^c_i$ are the approximated dimensions and utility received from its child $a_c \in C(a_i)$, respectively. $localUtil_i$ denotes the combination of the constraints between a_i and its (pseudo) parents, i.e.,

$$localUtil_i = \bigotimes_{a_i \in AP(a_i)} f_{ij} \tag{3}$$

Taking Fig. 1(c) for example, if we set $k = 1$, the dimensions approximated by a_4 are $\{x_1, x_2\}$. Thus, the approximated dimensions and utility sent from a_4 to a_3 are $SList^4_3 = \{x_1, x_2\}$ and $preUtil^4_3 = \min_{\{x_1,x_2,x_4\}} (f_{41} \otimes f_{42} \otimes f_{43})$, respectively.

Hybrid Phase consists of the search and context-based inference part. The search part uses a variant of SBB on a pseudo tree (i.e., a simple version of NCBB (Chechetka and Sycara 2006)) to expand any feasible partial assignments and provides contexts for the inference part. By using such contexts, the inference part propagates context-based utilities iteratively to produce tight lower bounds for the search part.

Traditionally, the context-based inference is implemented by considering the assignments to the approximated dimensions (that is, the decimated dimensions are equal to the approximated dimensions). For example, MB-DPOP performs an iterative context-based inference by considering each assignment combination of cycle-cut variables. However, the approach is not a good choice for our case due to the following facts. First, the number of the assignments of cycle-cut variables is exponential, which would incur unacceptable traffic overheads. Moreover, the propagated utilities w.r.t. a specific combination may go out of date since another inference is required as long as the assignments change, which would be very common in the search process. Therefore, it's unnecessary to perform inference for each assignment combination of the approximated dimensions.

Therefore, we consider to reduce the number of context-based inferences by making the propagated context-based utilities compatible with more contexts. Specifically, for agent a_i, we consider the decimated dimensions $PList_i \subseteq SList_i$. Then, the specific assignments to $PList_i$ serve as a context pattern and the propagated utilities will cover all the partial assignments with the same context pattern. That is, a_i's context pattern can be defined by:

$$ctxt_i = \{(x_j, Cpa_i(x_j)) | \forall x_j \in PList_i\} \tag{4}$$

where Cpa_i refers to a_i's received current partial assignment, and $Cpa_i(x_j)$ is the current assignment of x_j.

Taking agent a_3 in Fig. 1(c) as an example, given the limit $k = 1$, we have $SList_3 = \{x_1, x_2\}$. Assume that $PList_3 = \{x_1\}$. Thus, only the assignment of x_1 will be considered and x_4 (i.e., $SList_3 \backslash PList_3$) will be still dropped during the context-based inference part. Further, assume that $ctxt_3 = \{(x_1, 0)\}$. Then, $ctxt_3$ covers four contexts ($\{(x_1, 0), (x_2, 0)\}$,$\{(x_1, 0), (x_2, 1)\}$,$\{(x_1, 0), (x_2, 2)\}$ and $\{(x_1, 0), (x_2, 3)\}$). Therefore, a_3 only need to perform inference for $ctxt_3$ rather than the four contexts above in this case.

Selecting a context pattern is challenging as it offers a trade-off between the tightness of lower bounds and the number of compatible partial assignments. Thus, a good context pattern should comply with the following requirements. First, the good context pattern should be compatible with more contexts so as to avoid unnecessary context-based inference. In other words, these assignments are not likely to change in a short term. Second, the context pattern should result in tight lower bounds. Therefore, we propose a *context evaluation mechanism* to select the context pattern according to the frequency of an assignment of each dimension in the approximated dimensions. In more detail, we consider the context pattern consisting of the assignments whose frequency is greater than a specified threshold

Algorithm 1: Hybrid phase for agent a_i

When `Initialize()`:
1 perform ADPOP as a preprocessing phase
2 **if** a_i *is the root* **then**
3 $eval_i \leftarrow true$
4 $d_i \leftarrow$ the first element in D_i, $Cpa \leftarrow (x_i, d_i)$
5 send CPA($Cpa, eval_i$) to $a_c \in C(a_i)$

When `received CPA`$(Cpa, eval)$ *from* $P(a_i)$:
6 $preCpa_i \leftarrow Cpa_i$,$Cpa_i \leftarrow Cpa$,$eval_i \leftarrow eval$
7 **InitBounds**()
8 **UpdateSCounter**()
9 **if** $eval_i$ **then**
10 $ctxt_i \leftarrow \{(x_j, Cpa_i(x_j)) | Cnt_i(\langle x_j, Cpa_i(x_j)\rangle) > t, \forall x_j \in SList_i\}$
11 **if** $|ctxt_i| > 0$ **then**
12 **AllocateContext**()
13 $eval_i \leftarrow flase$
14 $ctxt_i \leftarrow \emptyset$
15 //mark a_i as the starter of the context-based inference part
16 //original CPA message handler

When `received CTXT` $(ctxt_i)$ *from* $P(a_i)$:
17 **AllocateContext**()
18 **if** $|inferChild_i| = 0$ **then**
19 **SendCtxtUtil**()

When `received CTXTUTIL` $(ctxtUtil_c)$ *from* $a_c \in inferChild_i$:
20 $ctxtUtil_i^c \leftarrow ctxtUtil_c$
21 **if** *received all CTXTUTIL from* $inferChild_i$ *and* $|ctxt_i| > 0$ **then**
22 **SendCtxtUtil**()

Function `InitBounds()`:
23 **foreach** $a_c \in C(a_i)$ **do**
24 **if** *received a CTXTUTIL that is compatible with* Cpa_i *from* a_c **then**
25 $lb_i^c(d_i) \leftarrow ctxtUtil_i^c(x_i = d_i, Cpa_i(Sep(a_c)))$
26 **else**
27 $lb_i^c(d_i) \leftarrow preUtil_i^c(x_i = d_i, Cpa_i(Sep(a_c)))$

Function `UpdateSCounter()`:
28 **foreach** $x_j \in SList_i \wedge (x_j, d_j) \in Cpa_i$ **do**
29 **if** $preCpa_i(x_j) \neq Cpa_i(x_j)$ **then**
30 $Cnt_i(\langle x_j, Cpa_i(x_j)\rangle) \leftarrow 1$
31 **else**
32 $Cnt_i(\langle x_j, Cpa_i(x_j)\rangle) \leftarrow Cnt_i(\langle x_j, Cpa_i(x_j)\rangle) + 1$

Function `AllocateContext()`:
33 $ctxt_i^c \leftarrow \{(x_j, d_j) | (x_j, d_j) \in ctxt_i, \forall x_j \in SList_i^c\}, \forall a_c \in C(a_i)$
34 $inferChild_i \leftarrow \{a_c || ctxt_i^c| > 0, \forall a_c \in C(a_i)\}$
35 send CTXT($ctxt_i^c$) to a_c,$\forall a_c \in inferChild_i$

Function `SendCtxtUtil()`:
36 $F \leftarrow localUtil_i \cup \{ctxtUtil_i^c | \forall a_c \in inferChild_i\}$
$\cup \{preUtil_i^c | \forall a_c \in C(a_i) \backslash inferChild_i\}$
37 $S_i' \leftarrow \{x_j | \forall x_j \in S_i \wedge (x_j, d_j) \notin ctxt_i\}$
38 $join \leftarrow \bigotimes_{f \in F} f\,(ctxt_i)$
39 $ctxtUtil_i \leftarrow \min_{S_i' \cup \{x_i\}} join$
40 send CTXTUTIL($ctxtUtil_i$) to $P(a_i)$

t. Given t, we have $PList_i = \{x_j | Cnt_i(\langle x_j, Cpa_i(x_j)\rangle) > t, \forall x_j \in SList_i\}$ where $Cnt_i(\langle x_j, Cpa_i(x_j)\rangle)$ refers to the frequency of $Cpa_i(x_j)$ for x_j. With an appropriate t, the assignments in the context pattern could not change in a short term. On the other hand, if a partial assignment is hard for pruning, then there would be more assignments included in the context pattern, which guarantees the lower bound tightness.

In addition, we introduce the variable $eval_i$ to ensure that the descendants of a_i perform an inference only for a context pattern at a time. Once a_i has found or received a context pattern, $eval_i$ is set to false to stop the context evaluation. And when the context pattern is incompatible with Cpa_i, $eval_i$ is set to true to indicate that a_i can find a new context pattern.

Next, we will detail how to implement the context evaluation mechanism and context-based inference part. Algorithm 1 presents the sketch of these procedures. We ignore the search part since it is an analogy to the tree-based branch and bound search algorithm PT-ISABB.

After the preprocessing phase (line 1), the root agent starts the context evaluation by initializing $eval_i$ with true (line 2-3). Besides, it also starts the search part via CPA messages with its first assignment to its children (line 4-5). Upon receipt of a CPA message, a_i first holds the previous Cpa_i, and stores the received Cpa_i and $eval_i$ (line 6). Afterwards, it initializes the lower bounds for each child $a_c \in C(a_i)$ according to its received utilities (line 7, 23-27). Concretely, the lower bound for a_c is established by the context-based utility compatible with Cpa_i received from a_c (line 24-25). Otherwise, the bound is computed by the utility received from a_c in the preprocessing phase (line 26-27). Next, a_i updates Cnt_i based on Cpa_i and the previous one (line 8, 28-32). Specifically, for each dimension in $SList_i$, a_i clears its counter if its assignment differs from its previous one (line 29-30). Otherwise, a_i increases that counter (line 31-32). Then, a_i finds a context pattern $ctxt_i$ if the pattern for the context-based inference has not been determined (line 9-10). After finding one, it allocates $ctxt_i$ and sends the allocated patterns via CTXT messages to its children who need to execute the context-based inference (i.e., $inferChild_i$) (line 11-15, 33-35). Here, $ctxt_i^c$, the allocated pattern for a_c, is a partition of $ctxt_i$ based on a_c's approximated dimensions $SList_i^c$ (line 33).

When receiving a CTXT message, a_i allocates the received pattern if there is any child who needs to perform the context-based inference (line 17, 33-35). Otherwise, it sends its context-based utility to its parent (line 18-19, 36-40). Here, its context-based utility is computed by the following steps. Firstly, it joins its local utility with the context-based utilities from $inferChild_i$ and the utilities from the other children (line 36). Next, it applies $ctxt_i$ to fix the values of the partial dimensions in S_i so as to improve the completeness of the utility (line 38). Finally, it drops the dimensions of the utility to stay below the limit k (line 39). After all the context-based utilities from $inferChild_i$ have arrived, a_i sends its context-based utility to its parent if it is not the starter of the context-based inference (line 21-22).

Considering a_3 in Fig. 1(c), assume that the context pattern has not been determined and $Cnt_3 = \{(\langle x_1, 0\rangle, 1), (\langle x_2, 0\rangle, 1)\}$. Given $t = 1$, we have $Cnt_3 = \{(\langle x_1, 0\rangle, 2), (\langle x_2, 1\rangle, 1)\}$ and $ctxt_3 = \{(x_1, 0)\}$ after a_3 receives a CPA with $\{(x_1, 0), (x_2, 1)\}$. Since the approximated dimensions for its child a_4 are $\{x_1, x_2\}$, a_3 sends a CTXT message with the context pattern $\{(x_1, 0)\}$ to a_4. When receiving the pattern $\{(x_1, 0)\}$, a_4 sends the context-based utility $ctxtUtil_3^4 = \min_{\{x_2, x_4\}} (f_{41}(x_1 = 0) \otimes f_{42} \otimes f_{43})$ to a_3. Then, a_3 uses $ctxtUtil_3^4$ to compute the lower bound for a_4 after receiving the CPA message with $\{(x_1, 0), (x_2, 2)\}$ or $\{(x_1, 0), (x_2, 3)\}$.

Theoretical Results

In this section, we first prove the effectiveness of the context-based inference on HS-CAI, and further establish the completeness of HS-CAI. Finally, we give the complexity anal-

ysis of the proposed method.

Lower Bound Tightness

Lemma 1. *For a given Cpa_i, the lower bound $lb_i^c(d_i)$ of $a_c \in C(a_i)$ for d_i produced by the context-based utility ($ctxtUtil_i^c$) is at least as tight as the one established by the utility ($preUtil_i^c$). That is, $ctxtUtil_i^c(PA) \geq preUtil_i^c(PA)$, where $PA = Cpa_i(Sep(a_c)) \cup \{(x_i, d_i)\}$.*

Proof. Directly from the pseudo code, S'_c, the dimensions dropped by a_c in the context-based inference part (line 37), can be defined by:

$$S'_c = \{x_j | (x_j, d_j) \notin ctxt_c, \forall x_j \in S_c\}$$

where $ctxt_c$ is the context pattern for a_c's context-based inference, and S_c is the dimensions dropped by a_c in the preprocessing phase. Since a_i has received $ctxtUtil_i^c$ from a_c, we have $|ctxt_c| > 0$ (line 34-35, 20-21). Thus, $S'_c \subset S_c$ is established.

Next, we will prove Lemma 1 by induction.

Base case. a_i's children are leaf agents. For each child $a_c \in C(a_i)$, we have

$$\begin{aligned}
&ctxtUtil_i^c(PA) = (\min_{S'_c \cup \{x_c\}} localUtil_c)(PA) \\
&= \min_{x_c}(\sum_{x_j \in AP(a_c) \setminus S'_c} f_{cj}(x_c, d_j) + \sum_{x_j \in S'_c} \min_{x_j} f_{cj}(x_c, x_j)) \\
&= \min_{x_c}(\sum_{x_j \in AP(a_c) \setminus S_c} f_{cj}(x_c, d_j) + \sum_{x_j \in S_c} \min_{x_j} f_{cj}(x_c, x_j) \\
&+ \sum_{x_j \in S_c \setminus S'_c} (f_{cj}(x_c, d_j) - \min_{x_j} f_{cj}(x_c, x_j))) \\
&\geq \min_{x_c}(\sum_{x_j \in AP(a_c) \setminus S_c} f_{cj}(x_c, d_j) + \sum_{x_j \in S_c} \min_{x_j} f_{cj}(x_c, x_j)) \\
&= (\min_{S_c \cup \{x_c\}} localUtil_c)(PA) = preUtil_i^c(PA)
\end{aligned}$$

where d_j is the assignment of x_j in PA. The equation in the third to the fourth step holds since $S'_c \subset S_c$. Thus, we have proved the basis.

Inductive hypothesis. Assume that the lemma holds for all a_i's children. Next, we are going to show the lemma holds for a_i as well. For each $a_c \in C(a_i)$, we have

$$\begin{aligned}
&ctxtUtil_i^c(PA) = (\min_{S'_c \cup \{x_c\}} (localUtil_c + \sum_{a_{c'} \in inferChild_c} ctxtUtil_c^{c'} \\
&+ \sum_{a_{c'} \in C(a_c) \setminus inferChild_c} preUtil_c^{c'}))(PA) \\
&\geq (\min_{S_c \cup \{x_c\}} (localUtil_c + \sum_{a_{c'} \in inferChild_c} ctxtUtil_c^{c'} \\
&+ \sum_{a_{c'} \in C(a_c) \setminus inferChild_c} preUtil_c^{c'}))(PA) \\
&\geq (\min_{S_c \cup \{x_c\}} (localUtil_c + \sum_{a_{c'} \in C(a_c)} preUtil_c^{c'}))(PA) \\
&= preUtil_i^c(PA)
\end{aligned}$$

where $inferChild_c$ are a_c's children who need to perform the context-based inference. Thus, Lemma 1 is proved. □

Correctness

Lemma 2. *Given the optimal solution X^*, $cost_c(X^*)$, the cost to the sub-tree rooted at $a_c \in C(a_i)$ is no less than the lower bound $lb_i^c(X^*(x_i))$. That is, $cost_c(X^*) \geq lb_i^c(X^*(x_i))$.*

Proof. Since we have proved the lower bounds constructed by the context-based inference part are at least as tight as the ones established by the preprocessing phase in Lemma 1, to prove the lemma, it is sufficient to show that $cost_c(X^*) \geq ctxtUtil_i^c(X^*)$.

Next, we will prove Lemma 2 by induction as well.

Base case. a_i's children are leaf agents. For each child $a_c \in C(a_i)$, we have

$$\begin{aligned}
&cost_c(X^*) = \sum_{a_j \in AP(a_c)} f_{cj}(d_c^*, d_j^*) \\
&= localUtil_c(X^*) \\
&\geq (\min_{S'_c \cup \{x_c\}} localUtil_c)(X^*) \\
&= ctxtUtil_i^c(X^*)
\end{aligned}$$

where d_l^* is the assignment of x_l in X^*, and S'_c is the dimensions dropped by a_c in the context-based inference part. Thus, the basis is proved.

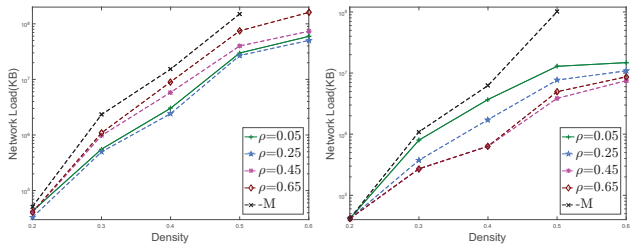
Inductive hypothesis. Assume the lemma holds for all $a_c \in C(a_i)$. Next, we will prove the lemma also holds for a_i. For each child $a_c \in C(a_i)$, we have

$$\begin{aligned}
&cost(Spa_c^*) = \sum_{a_j \in AP(a_c)} f_{cj}(d_c^*, d_j^*) + \sum_{a_{c'} \in C(a_c)} cost_{c'}(X^*) \\
&= localUtil_c(X^*) + \sum_{a_{c'} \in C(a_c)} cost_{c'}(X^*) \\
&\geq localUtil_c(X^*) + \sum_{a_{c'} \in C(a_c)} ctxtUtil_c^{c'}(X^*) \\
&= (localUtil_c + \sum_{a_{c'} \in C(a_c)} ctxtUtil_c^{c'})(X^*) \\
&\geq (\min_{S'_c \cup \{x_c\}} (localUtil_c + \sum_{a_{c'} \in C(a_c)} ctxtUtil_c^{c'}))(X^*) \\
&\geq (\min_{S'_c \cup \{x_c\}} (localUtil_c + \sum_{a_{c'} \in inferChild_c} ctxtUtil_c^{c'} \\
&+ \sum_{a_{c'} \in C(a_c) \setminus inferChild_c} preUtil_c^{c'}))(X^*) \\
&= ctxtUtil_i^c(X^*)
\end{aligned}$$

Thus, the lemma is proved. □

Theorem 1. *HS-CAI is complete.*

Proof. Immediately from Lemma 2, the optimal solution will not be pruned in HS-CAI. Furthermore, it has been proved that each agent will not receive two identical Cpas in the search part from PT-ISABB (Deng et al. 2019), and the termination of HS-CAI relies on the search part. Thus, HS-CAI is complete. □



(a) $k = 6$ (b) $k = 10$

Figure 2: Network load of varying ρ on different densities

Complexity

When it performs a context-based inference, a_i needs to store the context-based utilities and the utilities received from all its children. Thus, the overall space complexity is $O(|C(a_i)|d_{max}^k)$ where $d_{max} = \max_{a_j \in Sep(a_i)} |D_j|$, and k is the maximum dimension limit. Since a CTXTUTIL message only contains a context-based utility, its size is $O(d_{max}^k)$. For a CPA message, it is composed of the assignment of each agent and a context evaluation flag. Thus, the size of a CPA message is $O(|A|)$. Other messages like CTXT only carry an assignment combination of the approximated dimensions, it only requires $O(|A|)$ space.

The preprocessing phase in HS-CAI only requires $|A| - 1$ messages, since only the utility propagation are performed. For the search part and context-based inference part in the hybrid phase, the message number of the search part grows exponentially to the agent number with the same as the search-based complete algorithms. And the message number of the context-based inference part is proportional to the number of the context patterns selected by the context evaluation mechanism.

Empirical Evaluation

In this section, we first investigate the effect of the parameter t in the context evaluation mechanism on HS-CAI. Then, we present the experimental comparisons of HS-CAI with state-of-the-art complete DCOP algorithms.

Configuration and Metrics

We empirically evaluate the performance of HS-CAI and state-of-the-art complete DCOP algorithms including PT-FB, DPOP and MB-DPOP on random DCOPs. Besides, we consider HS-CAI without the context-based inference part as HS-AI and HS-CAI without the context evaluation mechanism as HS-CAI(-M). Here, HS-AI is actually a variant of PT-ISABB in DCOP settings. All evaluated algorithms are implemented in DCOPSovler[1], the DCOP simulator developed by ourselves. Besides, we consider the parameter t in HS-CAI related to both h and d_{max}, where $d_{\max} = \max_{a_i \in A} |D_i|$ and h is the height of a pseudo tree. Therefore, we set $t = (d_{max})^{\rho h}$. Moreover, we choose $k = 6$ and $k = 10$ as the low and high memory budget for MB-DPOP, HS-AI, HS-CAI(-M) and HS-CAI. In our experiments, we use the message number and network load (i.e., the size of total information exchanged) to measure the communication overheads, and the NCLOs (Netzer, Grubshtein, and Meisels 2012) to measure the hardware-independent runtime where the logical operations in the inference and the search are accesses to utilities and constraint checks, respectively. For each experiment, we generate 50 random instances and report the average of over all instances.

[1]https://github.com/czy920/DCOPSovler

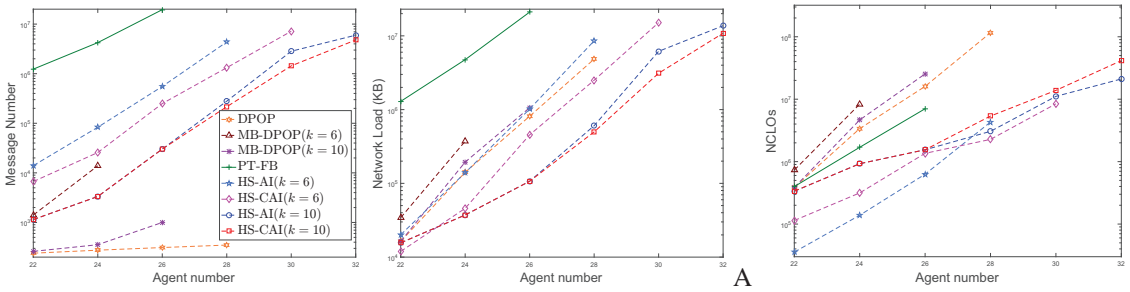
Parameter Tuning

Firstly, we aim to examine the effect of different ρ on the performance of HS-CAI to verify the effectiveness of the context evaluation mechanism. Specifically, we consider the DCOPs with 22 agents and the domain size of 3. The graph density varies from 0.2 to 0.6 and ρ varies from 0.05 to 0.65. Here, we do not show the experiment results of ρ greater than 0.65 since the larger ρ leads to the exact same results as ρ with 0.65. Fig. 2 presents the network load of HS-CAI(-M) and HS-CAI with different ρ. The average induced widths in this experiment are $8 \sim 16$. It can be seen from the figure that HS-CAI requires much less network load than HS-CAI(-M). That is because HS-CAI performs inference only for the context patterns selected by the context evaluation mechanism rather than all the contexts as HS-CAI(-M) does.

Besides, given the memory budget limit k, it can be observed that HS-CAI does not decrease all the time with the increase of ρ. This is due to the fact that increasing ρ which leads to large t can decrease the number of context-based inferences but also loose the tightness of lower bounds to some degree. Exactly as mentioned above, the context pattern selection offers a trade-off between the tightness of lower bounds and the number of compatible partial assignments. Moreover, it can be seen that the best value of ρ is close to 0.25 in HS-CAI($k = 6$) while the one in HS-CAI($k = 10$) is near to 0.45. Thus, we choose ρ to 0.25 for HS-CAI($k = 6$) and 0.45 for HS-CAI($k = 10$) in the following comparison experiments.

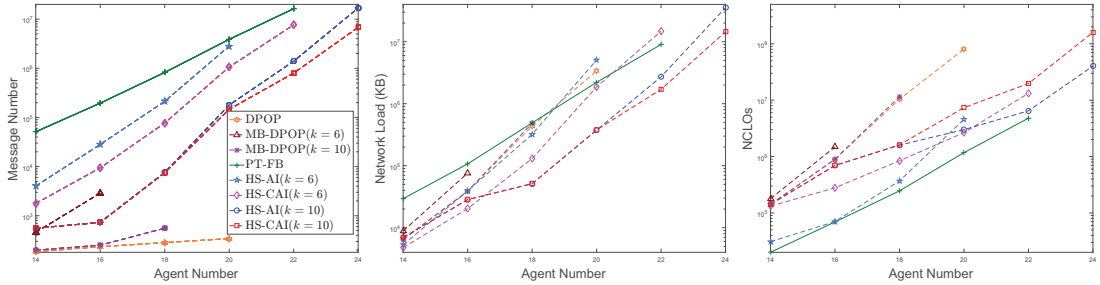
Performance Comparisons

Fig. 3 gives the experimental results under different agent numbers on the sparse configuration where we consider the graph density to 0.25, the domain size to 3 and vary the agent number from 22 to 32. Here, the average induced widths are $9 \sim 17$. It can be seen from Fig. 3(a) and (b) that although the hybrid complete algorithms (e.g., HS-AI and HS-CAI) and PT-FB all use the search strategy to find the optimal solution, HS-AI and HS-CAI are superior to PT-FB in terms of the network load and message number. This is because the lower bounds in PT-FB cannot result in effective pruning by only considering the constraints related to the assigned agents. Also, given a fixed k, HS-CAI requires fewer messages than HS-AI since the lower bounds produced by the context-based inference are tighter than the ones established by the context-free approximated inference. Besides, it can be seen that HS-CAI($k = 6$) can solve larger problems than HS-AI($k = 6$) and the inference-based complete algorithms like DPOP and MB-DPOP, which demonstrates the superior-



(a) Message Number (b) Network Load (c) NCLOs

Figure 3: Performance comparison under different agents on sparse configuration



(a) Message Number (b) Network Load (c) NCLOs

Figure 4: Performance comparison under different agents on dense configuration

ity of hybridizing search with context-based inference when the memory budget is relatively low.

Although inference requires larger messages than search, it can be observed from Fig. 3(b) that HS-CAI incurs less network load than HS-AI, which indicates that HS-CAI can find the optimal solution with fewer messages owing to the effective pruning and context evaluation mechanism. Moreover, we can see from Fig. 3(c) that when solving problems with the agent number to 28, HS-CAI($k = 6$) requires fewer NCLOs than HS-AI($k = 6$) in spite of the exponential computation overheads incurred by the iterative inferences. This is because that HS-CAI($k = 6$) can provide tight lower bounds to speed up the search so as to greatly reduce the constraint checks when solving the large scale problems under the limited memory budget.

Besides, we consider the DCOPs with the domain size of 3 and graph density of 0.6 as the dense configuration. The agent number varies from 14 to 24 and the average induced widths are $8 \sim 18$. Fig. 4 presents the performance comparison. It can be seen from Fig. 4(a) that DPOP and MB-DPOP cannot solve the problems with the agent number greater than 20 due to the large induced widths. Furthermore, since the inference-based complete algorithms have to perform inference on the entire solution space, these algorithms require much more NCLOs than the other competitors as Fig. 4(c) shows. Additionally, although they both perform the context-based inference, it can be seen from Fig. 4(b) and (c) that HS-CAI exhibits great superiority over MB-DPOP in terms of the network load and NCLOs. That is because HS-CAI only performs inference for the context patterns extracted by the context evaluation mechanism, while MB-DPOP needs to iteratively perform inference for all the contexts of cycle-cut variables. As for HS-CAI with different k, it can be seen from Fig. 4(a) and (c) that HS-CAI($k = 10$) requires fewer messages but more NCLOs than HS-CAI($k = 6$). That is because HS-CAI($k = 10$) can produce tighter lower bounds but will incur more computation overheads than HS-CAI($k = 6$).

Conclusion

By analyzing the feasibility of hybridizing search and inference, we propose a complete DCOP algorithm, named HS-CAI which combines search with context-based inference for the first time. Different from the existing hybrid complete algorithms, HS-CAI constructs tight lower bounds to speed up the search by executing context-based inference iteratively. Meanwhile, HS-CAI only needs to perform inference for a part of the contexts obtained from the search process by means of a context evaluation mechanism, which reduces the huge traffic overheads incurred by iterative context-based inferences. We theoretically prove that the context-based inference can produce tighter lower bounds compared to the context-free approximated inference under the same memory budget. Moreover, the experimental results show that HS-CAI can find the optimal solution faster with less traffic overheads than the state-of-the-art.

In the future, we will devote to further accelerating the search process by arranging the search space with the inference results. In addition, we will also work for reducing the

overheads caused by a context-based utility propagation.

Acknowledgments

This work is funded by the Chongqing Research Program of Basic Research and Frontier Technology (No.:cstc2017jcyjAX0030), Fundamental Research Funds for the Central Universities (No.:2019CDXYJSJ0021) and Graduate Research and Innovation Foundation of Chongqing (No.:CYS17023).

References

Atlas, J.; Warner, M.; and Decker, K. 2008. A memory bounded hybrid approach to distributed constraint optimization. In *Proceedings 10th International Workshop on DCR*, 37–51.

Brito, I., and Meseguer, P. 2010. Improving DPOP with function filtering. In *Proceedings of the 9th AAMAS*, 141–148.

Chechetka, A., and Sycara, K. 2006. No-commitment branch and bound search for distributed constraint optimization. In *Proceedings of the 5th AAMAS*, 1427–1429.

Dechter, R.; Cohen, D.; et al. 2003. *Constraint processing*. Morgan Kaufmann.

Deng, Y.; Chen, Z.; Chen, D.; Jiang, X.; and Li, Q. 2019. PT-ISABB: A hybrid tree-based complete algorithm to solve asymmetric distributed constraint optimization problems. In *Proceedings of the 18th AAMAS*, 1506–1514.

Farinelli, A.; Rogers, A.; Petcu, A.; and Jennings, N. R. 2008. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th AAMAS*, volume 2, 639–646.

Farinelli, A.; Rogers, A.; and Jennings, N. R. 2014. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous agents and multi-agent systems* 28(3):337–380.

Fioretto, F.; Yeoh, W.; Pontelli, E.; Ma, Y.; and Ranade, S. J. 2017. A distributed constraint optimization (DCOP) approach to the economic dispatch with demand response. In *Proceedings of the 16th AAMAS*, 999–1007.

Fioretto, F.; Pontelli, E.; and Yeoh, W. 2018. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research* 61:623–698.

Fioretto, F.; Yeoh, W.; and Pontelli, E. 2017. A multiagent system approach to scheduling devices in smart homes. In *Proceedings of the 16th AAMAS*, 981–989.

Freuder, E. C., and Quinn, M. J. 1985. Taking advantage of stable sets of variables in constraint satisfaction problems. In *Proceedings of the 9th IJCAI*, volume 85, 1076–1078.

Gershman, A.; Meisels, A.; and Zivan, R. 2009. Asynchronous forward bounding for distributed COPs. *Journal of Artificial Intelligence Research* 34:61–88.

Gutierrez, P.; Meseguer, P.; and Yeoh, W. 2011. Generalizing ADOPT and BnB-ADOPT. In *Proceedings of the 22nd IJCAI*, 554–559.

Hirayama, K., and Yokoo, M. 1997. Distributed partial constraint satisfaction problem. In *International Conference on Principles and Practice of Constraint Programming*, 222–236.

Kim, Y., and Lesser, V. 2014. DJAO: a communication-constrained DCOP algorithm that combines features of ADOPT and Action-GDL. In *Proceedings of the 28th AAAI*, 2680–2687.

Litov, O., and Meisels, A. 2017. Forward bounding on pseudo-trees for DCOPs and ADCOPs. *Artificial Intelligence* 252:83–99.

Maheswaran, R. T.; Tambe, M.; Bowring, E.; Pearce, J. P.; and Varakantham, P. 2004. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proceedings of the 3rd AAMAS*, volume 1, 310–317.

Maheswaran, R. T.; Pearce, J. P.; and Tambe, M. 2006. A family of graphical-game-based algorithms for distributed constraint optimization problems. In *Coordination of large-scale multiagent systems*. Springer. 127–146.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2005. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161(1-2):149–180.

Netzer, A.; Grubshtein, A.; and Meisels, A. 2012. Concurrent forward bounding for distributed constraint optimization problems. *Artificial Intelligence* 193:186–216.

Ottens, B.; Dimitrakakis, C.; and Faltings, B. 2017. DUCT: An upper confidence bound approach to distributed constraint optimization problems. *ACM Transactions on Intelligent Systems and Technologyc* 8(5):69.

Petcu, A., and Faltings, B. 2005a. Approximations in distributed optimization. In *International Conference on Principles and Practice of Constraint Programming*, 802–806.

Petcu, A., and Faltings, B. 2005b. A scalable method for multiagent constraint optimization. In *Proceedings of the 19th IJCAI*, 266–271.

Petcu, A., and Faltings, B. 2006. ODPOP: An algorithm for open/distributed constraint optimization. In *Proceedings of the 21st AAAI*, 703–708.

Petcu, A., and Faltings, B. 2007. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In *Proceedings of the 20th IJCAI*, 1452–1457.

Vinyals, M.; Rodriguez-Aguilar, J. A.; and Cerquides, J. 2009. Generalizing DPOP: DPOP, a new complete algorithm for DCOPs. In *Proceedings of the 8th AAMAS*, 1239–1240.

Yeoh, W.; Felner, A.; and Koenig, S. 2010. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. *Journal of Artificial Intelligence Research* 38:85–133.

Zhang, W.; Wang, G.; Xing, Z.; and Wittenburg, L. 2005. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161(1-2):55–87.