Leximin Multiple Objective Optimization for Preferences of Agents

Toshihiro Matsui[1], Marius Silaghi[2], Katsutoshi Hirayama[3], Makoto Yokoo[4], and Hiroshi Matsuo[1]

[1] Nagoya Institute of Technology, Gokiso-cho Showa-ku Nagoya 466-8555, Japan
{matsui.t, matsuo}@nitech.ac.jp
[2] Florida Institute of Technology, Melbourne FL 32901, United States of America
msilaghi@fit.edu
[3] Kobe University, 5-1-1 Fukaeminami-machi Higashinada-ku Kobe 658-0022, Japan
hirayama@maritime.kobe-u.ac.jp
[4] Kyushu University, 744 Motooka Nishi-ku Fukuoka 819-0395, Japan
yokoo@is.kyushu-u.ac.jp

Abstract. We address a variation of Multiple Objective Distributed Constraint Optimization Problems (MODCOPs). In the conventional MODCOPs, a few objectives are globally defined and agents cooperate to find the Pareto optimal solution. On the other hand, in several practical problems, the share of each agent is important. Such shares are represented as preference values of agents. This class of problems is defined as the MODCOP on the preferences of agents. Particularly, we focus on the optimization problems based on the leximin ordering (Leximin AMODCOPs), which improves the equality among agents. The solution methods based on pseudo trees are applied to the Leximin AMODCOPs.

Keywords: leximin, preference, multiple objectives, Distributed Constraint Optimization, multiagent, cooperation

1 Introduction

The Distributed Constraint Optimization Problem (DCOP) [3, 10, 15, 21] lies at the foundations of multiagent cooperation. With DCOPs, the optimization in distributed resource allocation uses the representation of a single objective function. The Multiple Objective Distributed Constraint Optimization Problem (MODCOP) [2] is an extension of the DCOP framework, where agents cooperatively have to optimize simultaneously multiple objective functions. For the case of multiple objectives, evaluation values are defined as vectors of objective values. Agents cooperate to find the Pareto optimal solution. In [2], a bounded Max-Sum algorithm for MODCOPs has been proposed. A solution method based on tree-search and dynamic programming has also been applied to MODCOPs [7]. In conventional MODCOPs, a few objectives are globally defined for the whole system. However, such models do not capture the interests of each agent. In several practical problems, the share of each agent is important. Such shares are represented as preference values of agents. This point of view recently has been addressed

in the context of DCOPs which are designed for dedicated resource allocation problems [12, 6, 13, 14]. These problems define multiple objective functions, optimizing the preferences for all the agents.

In this work, we address a class of MODCOPs on the preferences of agents. Particularly, we focus on problems where the importance of objective functions is based on the leximin ordering (referred to as Leximin AMODCOPs). Since the optimization based on the leximin ordering improves the equality among agents, this class of problems is important. The solution methods based on pseudo trees are applied to the Leximin AMODCOPs. Also, the investigated search methods employ the concept of boundaries of the sorted vectors.

2 Preliminary

2.1 Distributed Constraint Optimization Problem

A Distributed Constraint Optimization Problem (DCOP) is defined as follows.

Definition 1 (Distributed Constraint Optimization Problem). *A Distributed Constraint Optimization Problem is defined by (A, X, D, F) where A is a set of agents, X is a set of variables, D is a set of domains of variables, and F is a set of objective functions. Variable $x_i \in X$ represents a state of agent $i \in A$. Domain $D_i \in D$ is a discrete finite set of values for x_i. An objective function $f_{i,j}(x_i, x_j) \in F$ defines a utility extracted for each pair of assignments to x_i and x_j. The objective value of assignment $\{(x_i, d_i), (x_j, d_j)\}$ is defined by the binary function $f_{i,j} : D_i \times D_j \to \mathbb{R}$. For an assignment $\mathcal{A}$ of variables, the global objective function $F(\mathcal{A})$ is defined as $F(\mathcal{A}) = \sum_{f_{i,j} \in F} f_{i,j}(\mathcal{A}_{|x_i}, \mathcal{A}_{|x_j})$. The value of x_i is controlled by agent i. Agent i locally knows the objective functions that relate to x_i in the initial state. The goal is to find a global optimal assignment $\mathcal{A}^*$ that maximizes the global objective value.*

The computation to find the optimal solution is a distributed algorithm. We assume that each pair of agents has a communication route on an overlay network. For the sake of simplicity, we assume that all the objective functions are binary. Also, the state of each agent is represented by only one variable. However, the proposal can be generalized for n-ary functions and agent states represented by multiple variables.

2.2 Multiple objective problem

Multiple objective DCOP [2] (MODCOP) is a generalization of the DCOP framework. With MODCOPs, multiple objective functions are defined over the variables. The objective functions are simultaneously optimized based on appropriate criteria. The tuple with the values of all the objective functions for a given assignment is called *objective vector*.

Definition 2 (Objective vector). *An objective vector $\mathbf{v}$ is defined as $[v_0, \cdots, v_K]$. Here, v_k is an objective value. The Vector $\mathbf{F}(X)$ of objective functions is defined as $[F^0(X^0), \cdots, F^K(X^K)]$, where X^k is the subset of X on which F^k is defined.*

$F^k(X^k)$ is an objective function for objective k. For assignment $\mathcal{A}$, the vector $\mathbf{F}(\mathcal{A})$ of the functions returns an objective vector $[v_0, \cdots, v_K]$. Here, $v_k = F^k(\mathcal{A}^k)$ for each objective k.

Objective vectors are compared based on Pareto dominance. For maximization problems, the dominance between two vectors is defined as follows: Vector $\mathbf{v}$ dominates $\mathbf{v}'$ if and only if $\mathbf{v} \geq \mathbf{v}'$, and $v_k > v'_k$ for at least one objective k. Similarly, Pareto optimality on the assignments is defined as follows: Assignment $\mathcal{A}^*$ is Pareto optimal if and only if there is no other assignment $\mathcal{A}$, such that $\mathbf{F}(\mathcal{A}) \geq \mathbf{F}(\mathcal{A}^*)$, and $F^k(\mathcal{A}) > F^k(\mathcal{A}^*)$ for at least one objective k. In previous studies of MODCOPs [2], each objective function $f_{i,j}(x_i, x_j)$ in the original DCOPs is extended to a vector $[f^0_{i,j}(x_i, x_j), \cdots, f^K_{i,j}(x_i, x_j)]$. $F^k(\mathcal{A}^k)$ is therefore defined as $\sum_{f^k_{i,j} \in F^k} f^k_{i,j}(\mathcal{A}^k_{|x_i}, \mathcal{A}^k_{|x_j})$ for each objective k. Also, all the objectives are evaluated for the same assignment. Namely, $\mathcal{A}^0 = \mathcal{A}^1 = \cdots = \mathcal{A}^K$. Multiple objective problems generally have a set of Pareto optimal solutions that form a Pareto front. With an appropriate social welfare that defines an order on objective vectors, traditional solution methods for single objective problems find a Pareto optimal solution.

2.3 Social welfare

There are several criteria of social welfare [17] and scalarization methods [5]. A well-known social welfare function is defined as the summation $\sum_{k=0}^{K} F^k(\mathcal{A}^k)$ of objectives. The maximization of this summation ensures Pareto optimality. This summation is a 'utilitarian' criterion since it represents the total value of the objectives while it does not capture the equality on these objectives. On the other hand, the minimization $\min_{k=0}^{K} F^k(\mathcal{A}^k)$ on objectives emphasizes the objective of the worst value. Although the maximization of the minimum objective (maximin) reduces the worst complaint among all the objectives, the optimal assignment on the maximin is not Pareto (but weak Pareto) optimal. To improve maximin, the summation welfare function is additionally employed. A social welfare is defined as a vector $[\min_{k=0}^{K} F^k(\mathcal{A}^k), \sum_{k=0}^{K} F^k(\mathcal{A}^k)]$ with an appropriate definition of dominance. When the maximization on the minimization part dominates that on the summation part, it can be considered as a (partial) lexicographical ordering that yields the Pareto optimal solution, similar to the lexicographic weighted Tchebycheff method [5].

Another social welfare, called *leximin* [11, 1], is defined with a lexicographic order on objective vectors whose values are sorted in ascending order.

Definition 3 (Sorted vector). *A sorted vector based on vector $\mathbf{v}$ is the vector where all the values of $\mathbf{v}$ are sorted in ascending order.*

Definition 4 (Leximin). *Let $\mathbf{v}$ and $\mathbf{v}'$ denote vectors of the same length $K+1$. Let $[v_0, \cdots, v_K]$ and $[v'_0, \cdots, v'_K]$ denote sorted vectors of $\mathbf{v}$ and $\mathbf{v}'$, respectively. Also, let $\prec_{leximin}$ denote the relation of the leximin ordering. $\mathbf{v} \prec_{leximin} \mathbf{v}'$ if and only if $\exists t, \forall t' < t, v_{t'} = v'_{t'} \wedge v_t < v'_t$.*

The maximization on the leximin ordering ensures Pareto optimality. The leximin is an 'egalitarian' criterion since it reduces the inequality on objectives. It is also considered

as an improved version of maximin similar to a variation with the summation. The above property of the leximin is important for the preferences of agents. Further we focus on the leximin social welfare.

2.4 Preferences of agents

While previous studies address MODCOPs [2, 7], their goal is to optimize a few global objectives. Agents cooperate with each other to optimize those global objectives. On the other hand, in practical resource allocation problems, such as power supply networks, each agent has a strong interest for its share of the result. Hence there is the need for a more appropriate model where the objectives represent the preferences of agents. This class of problems has two key characteristics: 1) Each agent individually has its set of objective functions whose aggregated value represents its preferences, while several agents are related since subsets of their variables are in the scope of the same function. 2) The problem is a MODCOP where a solution is characterized by an objective vector consisting of objective values that are individually aggregated for different agents.

In [6], a resource constrained DCOP, which is designed for resource allocation on power supply networks, is extended to a MODCOP on the preferences of agents. In that study, min-max as well as min-max with the additional summation was introduced for minimizing problems. In addition, to reduce inequality among agents, a few first methods that consider the variance of objective values were shown. A general representation of the objectives of individual agents has been proposed as Asymmetric DCOP (ADCOP) [4]. In the ADCOP, two different objective functions are asymmetrically defined for a pair of two agents. Here, each objective function represents the valuation for one of the agents. Several classes of ADCOPs with multiple objectives for individual agents have been proposed in [12–14]. We focus on a class of ADCOPs optimizing the leximin social welfare. Since the leximin is known to reduce the inequality among agents, it helps define an important class of MODCOPs on preferences of agents.

3 Leximin multiple objective optimization on preferences of agents

3.1 Problem definition

A Leximin MODCOP on preferences of agents (Leximin AMODCOP) is defined as follows.

Definition 5 (Leximin MODCOP on preferences of agents). *A leximin MODCOP on preferences of agents is defined by (A, X, D, F), where A, X and D are similarly defined as for the DCOP in Definition 1. Agent $i \in A$ has its local problem defined on $X_i \subseteq X$. Here, $\exists(i,j), i \neq j \wedge X_i \cap X_j \neq \emptyset$. F is a set of objective functions $f_i(X_i)$. The function $f_i(X_i) : D_{i_0} \times \cdots \times D_{i_k} \rightarrow \mathbb{R}$ represents the objective value for agent i based on the variables in $X_i = \{x_{i_0}, \cdots, x_{i_k}\}$. For an assignment $\mathcal{A}$ of variables, the global objective function $\mathbf{F}(\mathcal{A})$ is defined as $[f_0(\mathcal{A}_0), \cdots, f_{|A|-1}(\mathcal{A}_{|A|-1})]$. Here, $\mathcal{A}_i$ denotes the projection of the assignment $\mathcal{A}$ on X_i . The goal is to find the assignment $\mathcal{A}^*$ that maximizes the global objective function based on the leximin ordering.*

As shown in Definition 5, each agent i has a function $f_i(X_i)$ that represents i's local problem. In a simple case, the local problem is defined as a part of an ADCOP where $f_i(X_i)$ is the summation of the corresponding functions in the ADCOP. In an ADCOP, variable x_i of agent i relates to other variables by objective functions. When x_i relates to x_j, agent i evaluates an objective function $f_{i,j}(x_i, x_j)$. On the other hand, j evaluates another function $f_{j,i}(x_j, x_i)$. Based on this ADCOP, a local problem is represented as $f_i(X_i) = \sum_{j \in Nbr_i} f_{i,j}(x_i, x_j)$ for agent i, aggregating objective functions among i and its neighborhood agents Nbr_i. While we will discuss our solution methods based on this ADCOP for the sake of simplicity, we address several motivated domains below.

Example 1 (Resource allocation on a power supply network). In a resource allocation problem on a power supply network [9, 6], each agent represents a node of the network. An agent i has several input links, output links and its resource. Given the amount $x^l_{i,j}$ of transferred resource on each input/output link (i, j) and x^r_i of its own resource, the total amount must satisfy resource constraint $c_i : \sum_{x^l_{j,i} \in X^{in}_i} x^l_{j,i} = x^r_i + \sum_{x^l_{i,k} \in X^{out}_i} x^l_{i,k}$. Here, X^{in}_i and X^{out}_i corresponds to input and output links, respectively. In addition, agent i has an objective function $f^r_i(x^r_i)$ of its own resource use x^r_i. Using a sufficiently small objective value for the violation of hard constraint c_i, this problem is represented by $f_i(X_i)$ for agent i, where X_i consists of $\{x^i_r\} \cup X^{in}_i \cup X^{out}_i$. The value of $f_i(X_i)$ is $f^r_i(x^r_i)$ if assignments for X_i satisfy c_i. Otherwise, $f_i(X_i)$ takes the sufficiently small value. Each agent desires to improve its local objective value under the resource constraints and preferences of other agents.

Example 2 (Variation of Coalition Structure Generation). A Coalition Structure Generation problem is represented as a DCOP [18]. An agent i has two variables x_i and x^g_i. x^g_i represents a group to which agent i belongs. x_i represents i's decision. Depending on x^g_i, utility values that relate to x_i are defined as follows. $f^v_{i,j}(x_i, x_j, x^g_i, x^g_j) = v_{i,j}(x_i, x_j)$ if $x^g_i \neq \text{`}alone' \wedge x^g_i = x^g_j$. Otherwise, $f^v_{i,j}(x_i, x_j, x^g_i, x^g_j) = 0$. $f^v_i(x_i, x^g_i) = v_i(x_i)$ if $x^g_i = \text{`}alone'$. Otherwise, $f^v_i(x_i, x^g_i) = 0$. Based on this DCOP, a local problem is represented as $f_i(X_i) = f^v_i(x_i, x^g_i) + \sum_{j \in Nbr_i} f^v_{i,j}(x_i, x_j, x^g_i, x^g_j)$ for agent i aggregating utility functions among i and its neighborhood agents Nbr_i.

4 Solution method based on pseudo tree

4.1 Pseudo tree for local problems

Several solution methods for DCOPs are based on pseudo trees on constraint networks [10, 15]. A pseudo tree of the problem is a depiction of its constraint network (adding directions to edges and levels for the nodes), based on a spanning tree in which there are no edges between different sub-trees of the corresponding spanning tree. Such pseudo trees can be generated using several algorithms, including the depth-first traversal on the constraint network. Edges of the spanning tree are called tree-edges while other edges are called back-edges. Based on the pseudo tree, the following notations are defined for each agent i: parent agent (p_i), set of child agents (Ch_i), the set of lower neighborhood agents, i.e. the child and pseudo child nodes ($Nbrs^l_i$), and the set

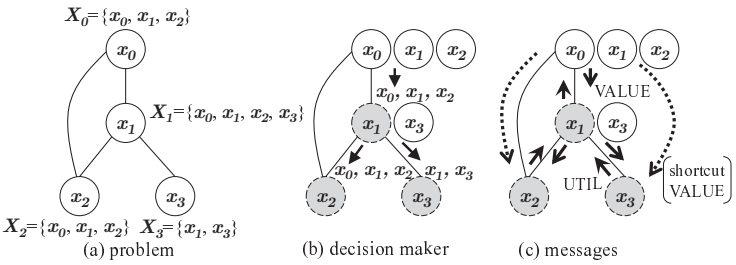

Fig. 1. Pseudo tree for local problems

of upper neighborhood agents, i.e. the parent and pseudo parent nodes ($Nbrs_i^u$). A partial order on a set of agents is defined based on the tree edges of a pseudo tree. The priorities induced by this order are used for breaking ties during decision making.

Figure 1(a) shows a pseudo tree for a problem. In the figure, four nodes represent agents/variables while four edges represent functions. In our problem, each edge stands for a pair of two asymmetric objective functions. Since an objective function is evaluated by only one related agent, each agent has to evaluate all the related objective functions. Namely, each agent has to manage all the assignments for its local problem. Therefore, the value of a variable x_i is decided by the highest neighborhood agent whose variable relates to the variable x_i with an edge. Hence a modification of pseudo trees is necessary. Figure 1(b) shows the pseudo tree modified from (a). The priority on decisions of assignments is represented as shown in (b).

To set up the data structures need for this pseudo tree, agent i computes the following information. XX_i^{upr}: A set of pairs of variables to compute the related agent in the highest level of the pseudo tree. X_i^{dcd}: The set of variables whose values are determined by agent i. X_i^{sep}: The set of *separator* variables that are shared between the sub-tree rooted at i and another part of the problem. Except at the root agent in the pseudo tree, the information is recursively computed as follows.

$$XX_i^{upr} = \bigcup_{h \in Nbrs_i^u} \{(x_i, x_h)\} \cup \{(x_a, x_b) | (x_a, x_b) \in \bigcup_{j \in Ch_i} XX_j^{upr} \wedge x_b \neq x_i\} \quad (1)$$

$$X_i^{dcd} = \{x_a | (x_a, x_i) \in \bigcup_{j \in Ch_i} XX_j^{upr} \wedge \nexists b, (x_a, x_b) \in XX_i^{upr}\} \quad (2)$$

$$X_i^{sep} = \left(\bigcup_{h \in Nbrs_i^u} \{x_h\} \cup \bigcup_{j \in Ch_i} X_j^{sep} \right) \setminus X_i^{dcd} \quad (3)$$

Equation (1) enables defining the agent assigning x_i as the highest placed agent in the set of those having a relation with some node in the sub-tree rooted as x_i (upper neighbors of i and upper neighbors of variables in sub-trees defined by its children, and found above i). Equation (2) defines the variables assigned by agent i as the lower neighbors of x_i in sub-trees defined by children, and which do not have upper neighbors above i. Equation (3) defines the separator variables as those in the upper neighbors of

x_i and its sub-tree, and of that are not controlled by agent i or its children. Note that $x_i \in X_i^{sep}$ and $x_i \notin X_i^{dcd}$, unlike the standard definition of separators on pseudo trees.

On the other hand, in the root agent, $XX_i^{upr} = \emptyset$, $X_i^{dcd} = \{x_a | (x_a, x_i) \in \bigcup_{j \in Ch_i} XX_j^{upr}\}$ and $X_i^{sep} = \emptyset$. Note that the root agent also determines the value of its own variable. The actual computation is performed as a distributed processing, after the preprocessing of generating a pseudo tree. Each non-root agent i sends XX_i^{upr}, X_i^{dcd} and X_i^{sep} to its parent agent p_i in a bottom-up manner.

4.2 Computation of the optimal objective vector

We apply a computation of the optimal objective value, which is employed in the solution method DPOP [15], to the Leximin AMODCOP. The computation is performed on the modified pseudo tree shown in Subsection 4.1. For the aggregation of objective values, we define an addition on vectors that is different from the common definition. The addition is the operator concatenating all the values.

Definition 6 (Addition on vectors). *Let* $\mathbf{v}$ *and* $\mathbf{v}'$ *denote vectors* $[v_0, \cdots, v_K]$ *and* $[v'_0, \cdots, v'_{K'}]$. *The addition* $\mathbf{v} \oplus \mathbf{v}'$ *of the two vectors gives a vector* $\mathbf{v}'' = [v''_0, \cdots v''_{K+K'+1}]$ *where each value in* $\mathbf{v}''$ *is a distinct value in* $\mathbf{v}$ *or* $\mathbf{v}'$. *Namely,* $\mathbf{v}''$ *consists of all values in* $\mathbf{v}$ *and* $\mathbf{v}'$. *As a normalization, the values in* $\mathbf{v}''$ *are sorted in ascending order.*

The computation of the optimal objective vector is recursively defined. The optimal objective vector $g_i^*(\mathcal{A}_i^{sep})$ for assignment $\mathcal{A}_i^{sep}$ of variables X_i^{sep} whose values are determined by i's ancestor nodes and parent node is represented as follows.

$$g_i^*(\mathcal{A}_i^{sep}) = \max_{\mathcal{A}_i^{dcd} \text{ for } X_i^{dcd}} g_i(\mathcal{A}_i^{sep} \cup \mathcal{A}_i^{dcd}) \tag{4}$$

$$g_i(\mathcal{A}) = [f_i(\mathcal{A}_{|X_i})] \oplus \bigoplus_{j \in Ch_i, \mathcal{A}_j^{sep} \subseteq \mathcal{A}} g_j^*(\mathcal{A}_j^{sep}) \tag{5}$$

Here, $\mathcal{A}_i^{dcd}$ denotes an assignment of the variables in X_i^{dcd} whose values are determined by i. The operator $\oplus$ denotes aggregation of objective values. While the summation operator is used in common DCOPs, we aggregate objective vectors using the operator shown in Definition 6. Similarly, $\max$ denotes the maximization on the leximin ordering. This computation is a dynamic programming based on the following proposition.

Proposition 1 (Invariance on leximin relation). *Let* $\mathbf{v}$ *and* $\mathbf{v}'$ *denote vectors of the same length. Also, let* $\mathbf{v}''$ *denote another vector. If* $\mathbf{v} \prec_{leximin} \mathbf{v}'$, *then* $\mathbf{v} \oplus \mathbf{v}'' \prec_{leximin} \mathbf{v}' \oplus \mathbf{v}''$.

Proof. Let $[v_0, \cdots, v_K]$ and $[v'_0, \cdots, v'_K]$ denote values in the sorted vectors of $\mathbf{v}$ and $\mathbf{v}'$, respectively. From the definition of leximin, there is a value t such that $\forall t' < t, v_{t'} = v'_{t'} \wedge v_t < v'_t$. Let t'' denote the value such that $v_{t''} < v_t \wedge v_{t''+1} = v_t$. Namely, $v_{t''}$ is the value just before the sequence of values equal to v_t. Note that $t'' + 1 \leq t$. In the case of $t = 0$, the value of t'' is generalized using -1. Consider the values in the sorted vectors

of $\mathbf{v} \oplus \mathbf{v}''$ and $\mathbf{v}' \oplus \mathbf{v}''$. When vector $\mathbf{v}''$ contains k values smaller than v_t, then there are $t''+k$ such values in both sorted vectors of $\mathbf{v} \oplus \mathbf{v}''$ and $\mathbf{v}' \oplus \mathbf{v}''$. Namely, the sequences of values less than v_t are the same in both of the sorted vectors. When vector $\mathbf{v}''$ contains k' values equal to v_t, $\mathbf{v} \oplus \mathbf{v}''$ contains a sequence of at least $(t - t'') + k'$ values equal to v_t. On the other hand, $\mathbf{v}' \oplus \mathbf{v}''$ contains a sequence of $(t - 1 - t'') + k'$ values equal to v_t. The above property also holds in the cases where $k = 0$ and/or $k' = 0$. Now, we can conclude that the sequences of the first $(t'' + k) + (t - 1 - t'') + k'$ values are the same in both sorted vectors of $\mathbf{v} \oplus \mathbf{v}''$ and $\mathbf{v}' \oplus \mathbf{v}''$, while the next values are the value equal to v_t and a value greater than v_t, respectively. Therefore, $\mathbf{v} \oplus \mathbf{v}'' \prec_{leximin} \mathbf{v}' \oplus \mathbf{v}''$.

The maximization in Expression (4) compares objective vectors for the same assignment $\mathcal{A}_i^{sep}$ that will produce the same partial objective vector. The above computation therefore correctly calculates the globally optimal objective vector.

After the computation of the optimal objective vector, the root agent i determine its optimal assignment $\mathcal{A}_i^{dcd*}$ such that $g_i(\emptyset \cup \mathcal{A}_i^{dcd*}) = g_i^*(\emptyset)$. $\mathcal{A}_j^{sep*} \subseteq \mathcal{A}_i^{dcd*}$ is then computed for each child $j \in Ch_i$. Similarly, non-root agent i computes $\mathcal{A}_i^{dcd*}$ such that $g_i(\mathcal{A}_i^{sep*} \cup \mathcal{A}_i^{dcd*}) = g_i^*(\mathcal{A}_i^{sep*})$, and $\mathcal{A}_j^{sep*} \subseteq \mathcal{A}_i^{sep*} \cup \mathcal{A}_i^{dcd*}$ for each child $j \in Ch_i$. The protocol of the modified version of DPOP is basically the same as the original one. The DPOP employs two types of messages UTIL and VALUE shown in Figure 1(c). After the processing of the modified pseudo tree, agents compute the optimal objective vector. In this computation, UTIL messages are propagated in a bottom-up manner. Each agent i sends $g_i^*(\mathcal{A}_i^{sep})$ to its parent p_i using UTIL message. Then the optimal assignment is computed propagating VALUE messages in a top-down manner. Each agent i sends $\mathcal{A}_j^{sep*}$ to its child agents $j \in Ch_i$ using VALUE message. The protocol of DPOP is quite simple. However, the size of UTIL messages and memory use to store $g_i^*(\mathcal{A}_i^{sep})$ of all the assignments exponentially increases with the size $|X_i^{sep}|$ of i's separator.

4.3 Search method

We apply solution methods based on tree search and partial dynamic programming to the Leximin AMODCOPs. The methods are variations of ADOPT [10, 20, 6], therefore needing less memory and employing messages of relatively smaller size. First, we show a simple search method, which is basically a time division of DPOP. While this method employs messages named VALUE and UTIL shown in Figure 1(c), they are different from those of DPOP. Similar to DPOP, the method consists of two phases of computations.

In the first phase, the optimal objective vector is computed in a manner of tree search. The root agent i chooses an assignment $\mathcal{A}_{i,j}^{dcd}$ for variables in $X_i^{dcd} \cap X_j^{sep}$ for its child $j \in Ch_i$. Then the root agent sends the current assignment $\mathcal{A}_j^{sep} = \mathcal{A}_{i,j}^{dcd}$ to its child node j using a VALUE message. When non-root agent i receives $\mathcal{A}_i^{sep}$ from its parent p_i, agent i chooses an assignment $\mathcal{A}_{i,j}^{dcd}$ for variables in $X_i^{dcd} \cap X_j^{sep}$ for its child j. Agent i then sends $\mathcal{A}_j^{sep} \subseteq \mathcal{A}_i^{sep} \cup \mathcal{A}_{i,j}^{dcd}$ for variables in X_j^{sep} to its child j. Namely, an assignment is expanded for all children of a node in a pseudo tree, in the same time. The current assignment $\mathcal{A}_i^{sep}$ is called *current context*. In the root agent, the current context is always $\emptyset$.

For the current context $\mathcal{A}_i^{sep}$, each agent computes $g_i^*(\mathcal{A}_i^{sep})$. Then $g_i^*(\mathcal{A}_i^{sep})$ is sent to i's parent p_i using a UTIL message. When agent i receives $g_j^*(\mathcal{A}_j^{sep})$ from its child j, $g_i^*(\mathcal{A}_j^{sep})$ is stored in the agent, if $\mathcal{A}_j^{sep}$ is compatible with $\mathcal{A}_i^{sep}$. When the current context changes to new assignment $\mathcal{A}^{sep\prime}{}_i$, objective vector $g_j^*(\mathcal{A}_j^{sep})$ whose $\mathcal{A}_j^{sep}$ is incompatible with $\mathcal{A}^{sep\prime}{}_i$ is deleted.

While the computation of $g_i^*(\mathcal{A}_i^{sep})$ is based on Equations (4) and (5), the computation is generalized to the case where agent i has not received $g_j^*(\mathcal{A}_j^{sep})$ from child j. In such cases, the lower and upper limit values of unknown objective values are introduced. With the limit values, the objective values are separated into lower and upper bound values. For the leximin ordering, we define the upper and lower bounds of objective vectors.

Definition 7 (Boundaries of unknown vector). *For an objective vector* $\mathbf{v}$ *of* K *unknown values, lower bound* $\mathbf{v}^{\perp}$ *and upper bound* $\mathbf{v}^{\top}$ *are vectors of* K *values, whose values are* $-\infty$ *and* ∞*, respectively.*

These boundaries are obviously reasonable since they are the minimum vector and the maximum vector on the leximin ordering. Operators $\oplus$ and $\prec_{leximin}$ are applied to the boundaries of vectors without any modifications. For a vector $\mathbf{v} = [v_0, \cdots, v_K]$ and the lower bound $\mathbf{v}'^{\perp} = [-\infty, \cdots, -\infty]$ of unknown vector $\mathbf{v}'$, the vector $\mathbf{v} \oplus \mathbf{v}'^{\perp}$ consists of $-\infty, \cdots, -\infty$ and $v_0, \cdots, v_K$. Similarly, $\mathbf{v} \oplus \mathbf{v}'^{\top}$ consists of $v_0, \cdots, v_K$ and $\infty, \cdots, \infty$. We consider these vectors as $(\mathbf{v} \oplus \mathbf{v}')^{\perp}$ and $(\mathbf{v} \oplus \mathbf{v}')^{\top}$, respectively.

Proposition 2 (Lower bound of partially unknown vector). *Let* $\mathbf{v}^{\perp}$ *denote a vector whose values are* $v_0, \cdots, v_K$ *and* K' *values of* $-\infty$*. For any vector* $\mathbf{v}$ *whose values are* $v_0, \cdots, v_K$ *and* K' *values greater than* $-\infty$*,* $\mathbf{v}^{\perp} \prec_{leximin} \mathbf{v}$.

Proof. While the first value in the sorted vector of $\mathbf{v}^{\perp}$ is $-\infty$, that of $\mathbf{v}$ is greater than $-\infty$. Therefore, $\mathbf{v}^{\perp} \prec_{leximin} \mathbf{v}$.

Proposition 3 (Upper bound of partially unknown vector). *Let* $\mathbf{v}^{\top}$ *denote a vector whose values are* $v_0, \cdots, v_K$ *and* K' *values of* ∞*. For any vector* $\mathbf{v}$ *whose values are* $v_0, \cdots, v_K$ *and* K' *values less than* ∞*,* $\mathbf{v} \prec_{leximin} \mathbf{v}^{\top}$.

Proof. Consider a vector $\mathbf{v}^{\top[v'_0]}$ where one of values ∞ in $\mathbf{v}^{\top}$ is replaced by a value v'_0 less than ∞. Both sorted vectors of $\mathbf{v}^{\top[v'_0]}$ and $\mathbf{v}^{\top}$ contain the same sequence of k values less than v'_0, since v'_0 does not affect this sequence. When $\mathbf{v}^{\top}$ contains k' values of v'_0, $\mathbf{v}^{\top[v'_0]}$ contains $k'+1$ values of v'_0. We can conclude that the sequences of the first $k+k'$ values are the same in both sorted vectors of $\mathbf{v}^{\top[v'_0]}$ and $\mathbf{v}^{\top}$, while the next values are the value equal to v'_0 and a value greater than v'_0, respectively. Therefore, $\mathbf{v}^{\top[v'_0]} \prec_{leximin} \mathbf{v}^{\top}$. Consider a vector $\mathbf{v}^{\top[v'_0,v'_1]}$ where one of values ∞ in $\mathbf{v}^{\top[v'_0]}$ is replaced by a value v'_1 less than ∞. Similar to $\mathbf{v}^{\top[v'_0]} \prec_{leximin} \mathbf{v}^{\top}$, we can conclude $\mathbf{v}^{\top[v'_0,v'_1]} \prec_{leximin} \mathbf{v}^{\top[v'_0]}$. Based on the mathematical induction, we can conclude that $\mathbf{v} = \mathbf{v}^{\top[v'_0,\cdots v'_{K'-1}]} \prec_{leximin} \cdots \prec_{leximin} \mathbf{v}^{\top[v'_0]} \prec_{leximin} \mathbf{v}^{\top}$ for any combination $[v'_0, \cdots v'_{K'-1}]$ of values that replace the values of ∞ in $\mathbf{v}^{\top}$.

In addition, with a bottom-up preprocessing, the lower and upper limit values for each function $f_i(X_i)$ can be aggregated to vectors of limit values instead of the vectors of $-\infty$ and ∞.

$g_i^*(\mathcal{A}_i^{sep})$ is extended to a pair of $g_i^{*\perp}(\mathcal{A}_i^{sep})$ and $g_i^{*\top}(\mathcal{A}_i^{sep})$ that are simultaneously computed. To introduce the boundaries, an agent has to know the number of descendants of each sub-tree rooted at each child. The information of the descendants is additionally computed in the preprocessing. When the number of descendants for a child j is dcd_j, $g_j^{*\perp}(\mathcal{A}_j^{sep})$ for unknown $g_j^*(\mathcal{A}_j^{sep})$ is a vector of dcd_j values of $-\infty$. Similarly, $g_j^{*\top}(\mathcal{A}_j^{sep})$ is a vector of dcd_j values of ∞.

Based on the boundaries, agents complete the tree search for sub problems. When $g_j^{*\perp}(\mathcal{A}_j^{sep}) = g_j^{*\top}(\mathcal{A}_j^{sep})$ for child $j \in Ch_i$, agent i completes the tree search for the assignment $\mathcal{A}_j^{sep}$. Then i chooses another assignment $\mathcal{A}^{sep\prime}{}_j$ such that $g_j^{*\perp}(\mathcal{A}^{sep\prime}{}_j) \prec_{leximin} g_j^{*\top}(\mathcal{A}^{sep\prime}{}_j)$. While there are several search strategies on the assignments, we employ a depth-first search based on the pseudo tree.

Now, a UTIL message carries a pair of vectors for the both boundaries. Since the boundaries are narrowed with the true objective values that are propagated in a bottom-up manner on the pseudo tree, agents repeatedly send UTIL messages. When agent i receives new vectors of $g^{*\perp}{}_j(\mathcal{A}_j^{sep})$ and $g^{*\top}{}_j(\mathcal{A}_j^{sep})$ from child $j \in Ch_i$, those vectors update the previous vectors. While $g^{*\perp}{}_j(\mathcal{A}_j^{sep})$ is maximized, $g^{*\top}{}_j(\mathcal{A}_j^{sep})$ is minimized with the new vectors based on the leximin ordering.

When $g_i^{*\perp}(\emptyset) = g_i^{*\top}(\emptyset)$ in the root agent i, agent i compute the optimal assignment $\mathcal{A}_i^{dcd*}$ such that $g_i^{\perp}(\emptyset \cup \mathcal{A}_i^{dcd*}) = g_i^{\top}(\emptyset \cup \mathcal{A}_i^{dcd*}) = g_i^{*\perp}(\emptyset) = g_i^{*\top}(\emptyset)$. $\mathcal{A}_j^{sep*} \subseteq \mathcal{A}_i^{dcd*}$ is then sent to each child $j \in Ch_i$ using a VALUE message with a flag of the termination. When $g_i^{*\perp}(\mathcal{A}_i^{sep*}) = g_i^{*\top}(\mathcal{A}_i^{sep*})$ in non-root agent i, the agent similarly computes the optimal assignment $\mathcal{A}_i^{dcd*}$ such that $g_i^{\perp}(\mathcal{A}_i^{sep*} \cup \mathcal{A}_i^{dcd*}) = g_i^{\top}(\mathcal{A}_i^{sep*} \cup \mathcal{A}_i^{dcd*}) = g_i^{*\perp}(\mathcal{A}_i^{sep*}) = g_i^{*\top}(\mathcal{A}_i^{sep*})$, and $\mathcal{A}_j^{sep*} \subseteq \mathcal{A}_i^{sep*} \cup \mathcal{A}_i^{dcd*}$ for each child $j \in Ch_i$ under $\mathcal{A}_i^{sep*}$. As a result, all the agents determine their optimal assignment.

4.4 Pruning

Next, we introduce the pruning based on the global lower bound of objective vectors. The global lower bound is $g_r^{*\perp}(\emptyset)$ in the root agent r. $g_r^{*\perp}(\emptyset)$ is propagated in a top-down manner using VALUE messages. An assignment $\mathcal{A}_j^{sep}$ for agent j is pruned if $g_r^{*\perp}(\emptyset) \not\prec_{leximin} g_j^{*\top}(\mathcal{A}_j^{sep})$. However, the length of $g_j^{*\top}(\mathcal{A}_j^{sep})$ is the number of agents in the sub-tree rooted at j while the length of $g_r^{*\perp}(\emptyset)$ equals the number of all the agents $|A|$. In this case, $\prec_{leximin}$ is applied as follows. Since $g_j^{*\top}(\mathcal{A}_j^{sep})$ is an upper bound, unknown objective values are represented by ∞. Therefore, with padding of ∞, $g_j^{*\top}(\mathcal{A}_j^{sep})$ and $g_r^{*\perp}(\emptyset)$ can be compared as the same length of vectors. Let $g_j^{*\top\top}(\mathcal{A}_j^{sep})$ denote the vector $g_j^{*\top}(\mathcal{A}_j^{sep})$ with the padding of ∞. In actual computation, the padding can be omitted since the sequence of ∞ is the last part of vectors. When $g_j^{*\perp}(\mathcal{A}_j^{sep}) = g_j^{*\top}(\mathcal{A}_j^{sep}) \vee g_r^{*\perp}(\emptyset) \not\prec_{leximin} g_j^{*\top\top}(\mathcal{A}_j^{sep})$ for child $j \in Ch_i$, agent i completes the tree search for the assignment $\mathcal{A}_j^{sep}$.

Moreover, to improve effects of the pruning, the upper bound for other parts of the problem is introduced. Namely, for each child agent $j \in Ch_i$, agent i computes the upper bound of objective vector $h_j^{+\top}(\mathcal{A}_j^{sep})$ for sub-trees except one rooted at j.

$$h_j^{+\top}(\mathcal{A}_j^{sep}) = h_i^{+\top}(\mathcal{A}_i^{sep}) \oplus \max_{\mathcal{A}^{dcd\prime}{}_i \text{ for } X_i^{dcd} \setminus X_j^{sep}} h_i^{\top}(\mathcal{A}_i^{sep} \cup \mathcal{A}^{dcd\prime}{}_i \cup \mathcal{A}_j^{sep}) \qquad (6)$$

$$h_i^{\top}(\mathcal{A}) = [\delta_i(\mathcal{A})] \oplus \bigoplus_{j \in Ch_i \setminus \{j\}, \mathcal{A}_j^{sep} \subseteq \mathcal{A}} g_j^{*\top}(\mathcal{A}_j^{sep}) \tag{7}$$

Note that the maximization in Equation (6) is not the maximization of objective values but the selection of the widest boundary. Since $\mathcal{A}_j^{sep}$ is a part of an assignment for $X_i^{sep} \cup X_j^{dcd}$, there are several assignments compatible with $\mathcal{A}_j^{sep}$. For such compatible assignments, the widest boundary prevents an over estimation. $h_j^{+\top}(\mathcal{A}_j^{sep})$ is sent from agent i to its child j using VALUE messages. When $g_j^{*\bot}(\mathcal{A}_j^{sep}) = g_j^{*\top}(\mathcal{A}_j^{sep}) \vee g_r^{*\bot}(\emptyset) \not\prec_{leximin} h_j^{+\top}(\mathcal{A}_j^{sep}) \oplus g_j^{*\top}(\mathcal{A}_j^{sep})$ for child $j \in Ch_i$, agent i completes the tree search for the assignment $\mathcal{A}_j^{sep}$.

4.5 Shortcut VALUE messages for modified pseudo tree

In several search methods [10, 20], additional VALUE messages are sent from ancestor agents to descendant agents taking shortcut paths. The shortcut VALUE messages directly carry assignments to deep levels of the pseudo tree. Then the assignments are propagated in a bottom-up manner using extended UTIL messages to update contexts. In our solution methods, the shortcut messages are particularly important to reduce the delay in updating the contexts since the decision makers of most variables are the agents in higher levels of the pseudo tree. In the conventional methods, the paths of shortcut VALUE messages are back edges. On the other hand, in our cases, back edges may not directly connect the decision maker and the deepest agent which relate to the same variable. In the example of Figure 1(c), the root agent sends x_0, x_1 and x_2 to the agent of x_2, and sends x_1 to the agent of x_3, respectively. Note that the root agent and the agent of x_3 are not directly connected. Therefore, we compute the deepest related agent for each variable in a bottom-up preprocessing, which is integrated to the preprocessing. The information on the deepest agent is stored in the corresponding decision maker. Agent i knows a set Sc_i of agents, to which shortcut VALUE messages are sent. For each agent $k \in Sc_i$, i computes $\mathcal{A}_k^{sc}$ containing assignments for k based on $\mathcal{A}_j^{sep}$, where child $j \in Ch_i$ is an ancestor of k. In addition, we employ timestamps based on the logical clock of the assignment for each variable, to compare the freshness of the assignment.

4.6 Pseudo code of search method

Algorithm 1 shows the pseudo code of the search method for agent i. Here ${}^i*$ denotes agent i's copy of $*$. Also, $*^{\bot/\top}$ denotes a pair of $*^{\bot}$ and $*^{\top}$. $\overrightarrow{-\infty}$ and $\overrightarrow{\infty}$ denote the vectors consisting of $-\infty$ and ∞, respectively. The length of these vectors is the same as the length of the vectors to be assigned. After the initialization (lines 2-4), agents repeatedly receive messages and maintain their status (lines 5-8). Note that the message passing is initiated by the root agent when it first enters the `Maintenance` state (line 8). When an agent receives a message, the agent updates its status based on the type of messages (lines 9-19). Then the agent maintains other data structures (lines 21-23). The root agent updates the global lower bound ${}^i g^{*\bot}(\emptyset)$ (line 21). If the termination condition is achieved, the agent determines its optimal assignment (line 22-23). Based on the updated status, messages are sent to other agents (line 24-29).

Algorithm 1: Distributed search for leximin AMODCOP (agent i)

```
1  Main:
2    if p_i = null then { A_i^sep ← ∅. h_i^{+⊤}(A_i^sep) ← [ ]. ptrm_i ← true. }
3    else { A_i^sep ← null. ptrm_i ← false. }
4    {}^i g_r^{*⊥}(∅) = −→∞. trm_i ← false.
5    forever do {
6      until receive loop exits do
7        if ¬trm_i then { receive a message. } else { purge all messages. }
8      if A_i^sep ≠ null ∧ ¬trm_i then Maintenance. }
9  Receive(VALUE, A, g, h, trm):
10   update A_i^sep by A. if A_i^sep=A then { h_i^{+⊤}(A_i^sep) ← h. } else { h_i^{+⊤}(A_i^sep) ← ⇉∞. }
11   {}^i g_r^{*⊥}(∅) ← g. ptrm_i ← trm. Consistent. return.
12 Receive(VALUE, A):
13   if A_i^sep ≠ null then {
14     update A_i^sep by A. if A_i^sep is updated then h_i^{+⊤}(A_i^sep) ← ⇉∞.
15     Consistent. } return.
16 Receive(UTIL, A, g^{⊥/⊤}):
17   update A_i^sep by A. if A_i^sep is updated then h_i^{+⊤}(A_i^sep) ← ⇉∞.
18   if A_i^sep is compatible with A then store/update {}^i g_j^{*⊥/⊤}(A) by g^{⊥/⊤}.
19   Consistent. return.
20 Maintenance:
21   if p_i = null ∧ {}^i g_r^{*⊥}(∅) ≺_leximin g_i^{*⊥}(∅) then {}^i g_r^{*⊥}(∅) ← g_i^{*⊥}(∅).
22   if ptrm_i ∧ g_i^{*⊥}(A_i^sep) = g_i^{*⊤}(A_i^sep) then {
23     determine A_i^{*dcd} corresponding to the termination condition. trm_i ← true. }
24   foreach j ∈ Ch_i do {
25     if trm_i then { determine A_j^sep from A_i^{*dcd}. } else { choose A_j^sep with a strategy. }
26     send (VALUE, A_j^sep, {}^i g_r^{*⊥}(∅), h_j^{+⊤}(A_j^sep), trm_i) to j. }
27   foreach k ∈ Sc_i do {
28     determine A_k^sc from A_j^sep of k's ancestor j. send (VALUE, A_k^sc) to j. }
29   if ¬ptrm_i then send (UTIL, A_i^sep, g_i^{*⊥/⊤}(A_i^sep)) to p_i.
30   return.
31 Consistent:
32   foreach A incompatible with A_i^sep do delete {}^i g_j^{*⊥/⊤}(A).
33   return.
```

4.7 Representation of objective vectors

In the whole computation of objective vectors, sorted vector can be employed. With the sorted vectors, the objective values of individual agents are not directly identified. The length of the objective vectors is upper bounded by the number of agents $|A|$. On the other hand, the sorted vector is compressed with run-length encoding, as a sequence of pairs $(objective\ value,\ length)$. This reduces both the size of the representation and the computation of $\prec_{leximin}$, when there are a number of the same objective values.

4.8 Correctness and Complexity

The both of the extended DPOP and the search method are variations of the previous solution methods [10, 15] while we use a representation without any subtraction. Therefore, their correctness is proven with the same reasoning as for the previous methods,

replacing the assignment concept with the proposed vectors (since we proved above that it satisfies the same additive properties). We have addressed how the computation is extended to Leximin AMODCOPs. Propositions 1, 2 and 3 shows that the monotonicity in the computation resembles the conventional solution methods based on addition. The properties on the computational/communication complexity of the proposed methods are also the same as those of the previous methods. On the other hand, the modified pseudo tree implicitly increases the induced width [15], which is $\prod_{x_i \in X_i^{sep}} |D_i|$ for agent i. The worst case of the basic tree search is as follows. 1) The tree is a single sequence of agents. 2) The decision maker is only the root node. 3) The evaluation is only made in the single leaf node. 4) No pruning works. Therefore, the maximum number of message cycles is $2(|A|-1)\prod_{x_j \in X} |D_j|$. However, this is an inherent property of the AMODCOPs. One can address large size problems using approximation methods. The maximum length of objective vector is the same as the number of agent $|A|$. With the representation using pairs of a value and its length, the size of the representation is between 2 and $2|A|$. This representation can be implemented with several tree structures, including Red-Blacks, tree whose major operations are performed in $O(\log n)$ time. The size of messages increases since their scalar values are replaced by the vectors.

5 Evaluation

The proposed method was experimentally evaluated. In our experiments with Leximin AMODCOPs (see Subsection 3.1) each problem consists of n ternary variables and c pairs of asymmetric objective functions. The constraint network is randomly generated by first creating a spanning trees and then adding additional edges. For each assignment, the objective function $f_{i,j}(x_i, x_j)$ returns an integer value w from $[0, 1]$ or $[0, 10]$ based on a uniform distribution. Note that we treat the aggregated function $f_i(X_i)$ as a black-box which cannot be decomposed. For each type of problem, the results are averaged over 25 instances. As the first experiment, we focused on the effects of search methods on the modified pseudo trees and the leximin ordering. The following solution methods were evaluated. `b`: the basic search method shown in Subsection 4.3. `gl`: `b` with the pruning based on the global lower bound shown in Subsection 4.4. `glou`: `bl` with the upper bound for other part of the problem shown in Subsection 4.4. `glousv`: `glou` with shortcut VALUE messages shown in Subsection 4.5. `lvb`, `lvgl`, `lvglou`, `lvglousv`: solution methods with the vectors of lower and upper limit values for each function $f_i(X_i)$ addressed in Subsection 4.3 [5]. The experiments were performed using simulation programs based on message cycles. In each message cycle, each agent receives messages from its message queue. Then the agent updates its status and sends messages if necessary. A simulation is interrupted after a number of 50000 cycles. Additionally, the number of non-concurrently performed operations (ncops) relating to objective functions and assignments is also evaluated. While it resembles ncccs [8], we also consider several operations that involve a (partial) assignment.

[5] In this case, to avoid over estimation, we modified the condition of the pruning in the second phase using a flag. Agent i completes the tree search for the assignment $\mathcal{A}_j^{sep}$ when $g_j^{*\perp}(\mathcal{A}_j^{sep}) = g_j^{*\top}(\mathcal{A}_j^{sep}) \vee h_j^{+\top}(\mathcal{A}_j^{sep}) \oplus g_j^{*\top}(\mathcal{A}_j^{sep}) \prec_{leximin} g_r^{*\perp}(\emptyset)$ for child $j \in Ch_i$.

Table 1. Number of iterations ($w = [0, 10]$) (trm.: number of completed instances)

n, c	10, 9			10, 12			10, 15			20, 19			20, 22			40, 39		
alg.	msg. cyc.	ncop. (10^3)	trm.	msg. cyc.	ncop. (10^3)	trm.	msg. cyc.	ncop. (10^3)	trm.	msg. cyc.	ncop. (10^3)	trm.	msg. cyc.	ncop. (10^3)	trm.	msg. cyc.	ncop. (10^3)	trm.
b	781	125	25	16312	6582	22	42125	45183	8	12092	1535	23	38866	16147	11	38499	8303	9
gl	660	118	25	6617	4349	25	28349	36721	18	8413	1320	23	29768	12341	16	35431	7847	11
glou	332	300	25	3169	4622	25	20553	42786	23	3602	3930	25	19774	25064	21	19922	28369	19
glousv	212	268	25	2140	5268	25	17692	52776	24	1561	3068	25	12813	25179	24	11267	22692	24
lvb	511	96	25	15584	6342	23	41998	44905	8	9295	1314	24	37743	15529	13	33538	8009	13
lvgl	434	92	25	6001	4206	25	25216	36259	19	5903	1095	24	27612	11706	19	30150	7372	15
lvglou	214	231	25	2473	4409	25	16461	41398	25	1046	2829	25	7645	18228	25	12305	23290	23
lvglousv	146	216	25	1787	5065	25	13970	51438	25	605	2465	25	4758	19189	25	4974	17074	25

Table 2. Size of pseudo tree (no dcd.: $|X_i^{dcd}| = 0$)

n, c	max. depth	max. $\|Ch_i\|$	max. $\|X_i^{sep}\|$	max. $\|X_i\|$	max. $\|X_i^{dcd}\|$	#agent no dcd.	max. $\|Sc_i\|$
10, 9	5	4	2	5	5	4	5
10, 12	6	2	5	5	5	6	5
10, 15	7	2	7	6	6	6	6
20, 19	8	4	2	5	5	9	5
20, 22	9	3	5	6	6	10	6
40, 39	11	5	2	6	6	17	6

Table 3. Size of vector

w	[0, 1]						[0, 10]					
n, c	20, 22			40, 39			20, 22			40, 39		
alg.	len.	sz.	2sz.	len.	sz.	2sz.	len.	sz.	2sz.	len.	sz.	2sz.
lvb	5	2	4	4	2	4	5	4	8	4	3	7
lvgl	9	3	5	15	2	5	9	6	13	15	7	14
lvglou	11	3	6	20	3	6	11	7	15	20	9	19
lvglousv	10	3	6	20	3	6	11	7	14	20	9	18

Table 1 shows the number of iterations. The efficient methods reduce the number of message cycles. Particularly, `glou` is effective since it prunes branches with full information of boundaries. Although `glou` employs the limit values $-\infty$ and ∞, the pruning works. The effect comes from the property that leximin partially compares values in two vectors. In addition, the lower and upper limit values for each function $f_i(X_i)$ are effective in the case of trees and less effective for cyclic networks. This reveals the need for better bounding methods, as available with conventional DCOP solvers. Such methods are, however, domain specific since the decomposition of $f_i(X_i)$ and the identification of the preferences of the agents will be necessary. Advanced methods need more ncops than basic methods. Also, the additional shortcut VALUE messages are necessary, similar to ADOPT [10, 20]. Therefore, there are several trade-offs between computation and communication. On the other hand, there are opportunities to reduce ncops in our implementation. Table 2 shows the size of the pseudo trees. There are a number of agents with an empty X_i^{dcd}. These agents only evaluate their objective values. While there are opportunities to reduce this redundancy by revealing the objective functions of the agents, it will also be domain specific. Table 3 shows the size of the vectors. The actual size (2sz.) of the representation of the vectors is relatively smaller than the length (len.) of the vectors in the case of $w = [0, 1]$. In these results, the computation of leximin is reduced since the number of pairs (sz.) to be enumerated is less than the length of vectors. Table 4 shows the comparison between leximin (max-leximin) and other optimization criteria. The other optimization criteria are summation (max-sum), maximin (max-min), and maximin with additional summation (max-LWT). These criteria were also applied to the solvers based on pseudo trees, similar to the previous solvers [6]. Each cell shows the number of cases of dominance ($\prec$ or $\succ$) or tie ($=$). On the summation of objective values, max-sum and max-LWT are never dominated by max-leximin. Max-leximin, max-min and max-LWT give the same minimum objective value. For max-sum and max-LWT, max-leximin relatively decreases the variance of objective values. Max-min is not Pareto optimal while the other criteria are Pareto optimal.

Table 4. Comparison between max-leximin and other optimization criteria ($w = [0, 10]$)

comparison	sum						min						max					
optimization	max-sum			max-min			max-LWT			max-sum			max-min			max-LWT		
n, c	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$
10, 15	24	1	0	3	3	19	19	6	0	0	6	19	0	25	0	0	25	0	21	2	2	14	3	8	18	7	0
20, 22	25	0	0	2	0	23	23	2	0	0	2	23	0	25	0	0	25	0	23	2	0	11	3	11	18	6	1
40, 39	25	0	0	0	0	25	24	1	0	0	0	25	0	25	0	0	25	0	22	1	2	10	3	12	18	5	2
comparison	variance									leximin									Pareto								
optimization	max-sum			max-min			max-LWT			max-sum			max-min			max-LWT			max-sum			max-min			max-LWT		
n, c	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$	$\prec$	=	$\succ$
10, 15	24	1	0	14	2	9	20	5	0	0	1	24	0	2	23	0	5	20	0	25	0	0	22	3	0	25	0
20, 22	25	0	0	11	0	14	24	1	0	0	0	25	0	0	25	0	1	24	0	25	0	0	24	1	0	25	0
40, 39	25	0	0	12	0	13	24	0	1	0	0	25	0	0	25	0	0	25	0	25	0	0	25	0	0	25	0

6 Related works and discussions

In ADCOP [4], each value of a function is defined by a pair of values that correspond to different directions on an edge of a constraint graph. Therefore, an agent has its local view based on the direction of connected edges. However, its optimal solution corresponds to the maximum summation over all functions and directions. In [12–14, 6], resource allocation problems similar to ones in this study have been addressed. On the other hand, we addressed an extension of ADCOPs based on the leximin social welfare. While Theil based social welfare has been addressed in [12], that solution method is a local search. In our proposed search methods, the high induced width exponentially increases the number of search iterations. For addressing this issue, a promising direction is to investigate more aggressive modifications of graphs [19]. Also, there are opportunities to approximate the problems [16, 2]. While existing efficient techniques including forward-bounding may improve the efficiency of solution methods [13], it needs several assumptions such that each preference function is additive and can be decomposed to sub-functions in exchange for the privacy of the agents. While several solution methods for a centralized constraint optimization problem on the leximin ordering have been proposed, for example [1], they are dedicated extensions of centralized solvers.

7 Conclusions

In this work, we presented a multiple objective DCOP that considers preferences of agents, and its solution method based on the leximin ordering on multiple objectives. Our future work will include improvements to reduce redundant computations, evaluations in practical domains, and analysis on various types of problems.

References

1. Bouveret, S., Lemaître, M.: Computing leximin-optimal solutions in constraint networks. Artificial Intelligence 173(2), 343–364 (2009)
2. Delle Fave, F.M., Stranders, R., Rogers, A., Jennings, N.R.: Bounded decentralised coordination over multiple objectives. In: 10th International Conference on Autonomous Agents and Multiagent Systems. vol. 1, pp. 371–378 (2011)
3. Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: 7th International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 639–646 (2008)

4. Grinshpoun, T., Grubshtein, A., Zivan, R., Netzer, A., Meisels, A.: Asymmetric distributed constraint optimization problems. Journal of Artificial Intelligence Research 47, 613–647 (2013)
5. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26, 369–395 (2004)
6. Matsui, T., Matsuo, H.: Considering equality on distributed constraint optimization problem for resource supply network. In: 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology. vol. 2, pp. 25–32 (2012)
7. Matsui, T., Silaghi, M., Hirayama, K., Yokoo, M., Matsuo, H.: Distributed search method with bounded cost vectors on multiple objective dcops. In: Principles and Practice of Multi-Agent Systems - 15th International Conference. pp. 137–152 (2012)
8. Meisels, A., Kaplansky, E., Razgon, I., Zivan, R.: Comparing performance of distributed constraints processing algorithms. In: 3rd International Workshop on Distributed Constraint Reasoning. p. (no page numbers) (2002)
9. Miller, S., Ramchurn, S.D., Rogers, A.: Optimal decentralised dispatch of embedded generation in the smart grid. In: 11th International Conference on Autonomous Agents and Multiagent Systems. vol. 1, pp. 281–288 (2012)
10. Modi, P.J., Shen, W., Tambe, M., Yokoo, M.: Adopt: Asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence 161(1-2), 149–180 (2005)
11. Moulin, H.: Axioms of Cooperative Decision Making. Cambridge : Cambridge University Press (1988)
12. Netzer, A., Meisels, A.: SOCIAL DCOP - Social Choice in Distributed Constraints Optimization. In: 5th International Symposium on Intelligent Distributed Computing. pp. 35–47 (2011)
13. Netzer, A., Meisels, A.: Distributed Envy Minimization for Resource Allocation. In: 5th International Conference on Agents and Artificial Intelligence. vol. 1, pp. 15–24 (2013)
14. Netzer, A., Meisels, A.: Distributed Local Search for Minimizing Envy. In: 2013 IEEE/WIC/ACM International Conference on Intelligent Agent Technology. pp. 53–58 (2013)
15. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: IJCAI05. pp. 266–271 (2005)
16. Rogers, A., Farinelli, A., Stranders, R., Jennings, N.R.: Bounded approximate decentralised coordination via the Max-Sum algorithm. Artificial Intelligence 175(2), 730–759 (2011)
17. Sen, A.K.: Choice, Welfare and Measurement. Harvard University Press (1997)
18. Ueda, S., Iwasaki, A., Yokoo, M., Silaghi, M., Hirayama, K., Matsui, T.: Coalition structure generation based on distributed constraint optimization (2010)
19. Vinyals, M., Rodriguez-Aguilar, J.A., Cerquides, J.: Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. Autonomous Agents and Multi-Agent Systems 22(3), 439–464 (2011)
20. Yeoh, W., Felner, A., Koenig, S.: Bnb-adopt: an asynchronous branch-and-bound dcop algorithm. In: 7th International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 591–598 (2008)
21. Zivan, R.: Anytime local search for distributed constraint optimization. In: AAAI08. pp. 393–398 (2008)