考査者についての詳細は以下の通りです。

<table>
<thead>
<tr>
<th>姓氏 (英)</th>
<th>ドメイン名1 域名2 域名3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ツシホロウ</td>
<td>マツシウ</td>
</tr>
</tbody>
</table>

著作権 ©2012 IEEE. 本材料の個人利用は許可されているが、商用利用や他の目的での利用は、IEEEに事前に許可が必要です。
Considering Equality on Distributed Constraint Optimization Problem for Resource Supply Network

Toshihiro Matsui and Hiroshi Matsuo
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 Japan
Email: {matsui.t, matsuo}@nitech.ac.jp

Abstract—Distributed resource allocation is an important application of multiagent systems. In this work, we focus on a resource allocation problem that is motivated from a power supply network that contains distributed sources. In the supply network, resources that are initially distributed among source nodes have to be shared among all nodes. The problem is formalized as a resource constrained distributed constraint optimization problem that is an extended class of distributed constraint optimization problems (DCOPs). In the formalization, cost functions represent preferences of agents on resource use. We specifically consider allocating the cost values to agents as evenly as possible. We present several methods to select optimal assignment in consideration of the equality. Characteristics of the proposed methods are experimentally evaluated. Employing histograms and distinguishing types are effective for reducing the variance while requiring high computational costs. The number of histograms is able to be limited without significant lack of the effects. The minimum-maximum cost value as the main objective reduces the number of iterations and histograms.

Keywords—resource allocation; distributed constraint optimization; multi-agent; equality

I. INTRODUCTION

Distributed resource allocation is an important application of multiagent systems. While there are a number of types of resource allocation, including sensor networks and power supply networks, resource allocation essentially contains optimization problems. Therefore, distributed optimization methods are necessary to solve the allocation.

As a basic framework of cooperative problem solving in multiagent systems, Distributed Constraint Optimization Problems (DCOPs) have been studied [1], [2], [3], [4]. With DCOPs, the states of agents and the relationships between agents are formalized into a constraint optimization problem that is solved by distributed search algorithms. These studies focus on the optimization problems and distributed search algorithms that are essentially contained in cooperative protocols of the multiagent systems. Several cooperative problems including distributed resource scheduling and sensor networks are represented as DCOPs [4], [5]. For more special cases, the representation of DCOPs can be extended to meet a particular problem. A solver is then also modified for the problem. Resource Constrained DCOPs (RCDCOPs) [6] make up a dedicated class of problems that explicitly represents shared resources as global constraints that can be decomposed into agents.

We focus on a resource allocation problem motivated from the power supply networks of smart grid systems. In supply networks, resources that are initially distributed among source nodes have to be shared among all nodes. In a related work [7], a dedicated class of DCOPs and a solver for the problem of power supply restoration [8] have been proposed. The goal in the study is to generate feeder trees under resource constraints.

In this study, we address sharing of resources in a feeder tree. Optimization for the supply and consumption of resources in a supply network that contains distributed sources is considered an important domain. We define an example problem motivated from resource sharing as a variation of RCDCOPs in which the unary cost functions represent agents’ preferences on resource use. The optimal solution of the problem represents the appropriate assignment of amounts of the resource that is supplied and consumed in each node of the network.

We particularly focus on allocating cost values to agents as evenly as possible. That is, the preferences of agents should be evenly satisfied. While the equality of agents is important in practical problems, it has not been well addressed in studies of (RC)DCOPs. We present several methods to select optimal assignment in consideration of the equality. Characteristics of the proposed methods are experimentally evaluated.

The rest of our paper is organized as follows: In Section II, we address a resource allocation problem that is motivated from a power supply network. The problem is then formalized as an RCDCOP and a basic solver is shown. In III, several methods to reduce inequality in optimal assignments among agents are proposed. The proposed model and solvers are experimentally evaluated in Section IV. Related works are addressed in Section V and we set forth our conclusions in Section VI.

II. PRELIMINARY

A. Motivated problem

We define a resource allocation problem on a network that is motivated from power supply networks containing
distributed resources. In the network, amounts of the resource in the sources are allocated to sinks. For the sake of simplicity, we limit the structure of the networks to trees. Such feeder-trees are common in actual power networks.

We also assume that the feeder-trees are constructed in pre-processing.

The network consists of the following elements:

- nodes: each node supplies or consumes an amount of resource. Nodes that supply resources are called sources. Nodes that consume other nodes’ resources are called sinks. There are limitations on the amount of supply and consumption. The node also has a preference on the amount of the resource.
- links: each link connects two nodes. The links and nodes form paths to transfer an amount of resource. There are limitations on the amount of resource that is transferred in a link. We do not consider the loss of the resource when it is transferred.

In addition to the limitations on the amount of resource, there is the constraint that the total amount of the resource that is supplied and consumed must be zero. Basically, the goal of the problem is to globally optimize an aggregation of preference under the constraints.

Formally, the problem is defined by \(<N, L, R, F, \mathcal{L}>\), where \(N\), \(L\), \(R\), \(F\), and \(\mathcal{L}\) are a set of nodes, a set of links, a family set of amounts of resources on nodes, a set of cost functions, and a family set of amounts of resources on links, respectively.

For node \(i \in N\), the preference of the supply and the consumption on the amounts of the resource is represented as follows:

- \(R_i; R_i \in R\) is a finite set of amounts of resource that are supplied or consumed by node \(i\). Where amount \(r \in R_i\) is a negative value, \(r\) represents an amount of the supplied resource. Where amount \(r\) is a positive value, it represents an amount of consumed resources. Node \(i\) chooses a value of the amount from \(R_i\).
- \(f_i(r); f_i(r) \in F\) is a cost function from amount \(r \in R_i\) of the resource to a non-negative value. We use cost functions to represent the preferences of the nodes because solution methods are defined for minimizing problems.

Each link is defined for a pair \((i, j)\) of nodes. For link \((i, j) \in L\), the transfer of the resource is represented as follows:

- \(L_{i,j}; L_{i,j} \in \mathcal{L}\) is a finite set of amounts of resource that is transferred through link \((i, j)\). Amount \(l \in L_{i,j}\) is a finite value such that \(-l_{i,j} \leq l \leq l_{i,j}\), where \(l_{i,j}\) is the capacity of link \((i, j)\). The positive and negative values in \(L_{i,j}\) are symmetrical. The sign of value \(l\) represents the direction of the transfer. \(l\) takes a positive value when the corresponding link transfers an amount of resource on a downward path of a feeder-tree rooted at a node.

In each node \(i \in N\), the summation of \(r_i\) and \(l_{i,j}\) for all links \((i, j)\) that connect node \(i\) must always be zero. The constraint is defined using set \(L^i\) of links that connect node \(i\).

\[r_i + \sum_{(i,j)\in L^i} l_{i,j} = 0 \quad (1)\]

For allocation \(\mathcal{R}\) of amounts of the resource for all nodes, the global cost \(f(\mathcal{R})\) is defined as follows:

\[f(\mathcal{R}) = \sum_{i\in N} f_i(r_i) \quad (2)\]

Here \(r_i\) takes a corresponding value in \(\mathcal{R}\). The goal of the problem is to find the optimal allocation \(\mathcal{R}^*\) that minimizes \(f(\mathcal{R})\) under the constraints.

B. Formalization as resource constrained DCOP

DCOP is a framework of multiagent cooperation. With the representation of DCOPs, an optimization problem in a multiagent system is defined as a constraint optimization problem whose variables, constraints, and evaluation functions are distributed among agents. The problem is solved using distributed cooperative search algorithms that are based on message communication.

RCDCOP is an extended class of DCOPs that contains dedicated representations of resources and constraints related to the resources. We define an RCDCOP that represents the resource allocation problem in subsection II-A. The resource allocation problem is directly translated into an RCDCOP using variables, constraints, and functions.

The RCDCOP for the resource allocation on the network is defined by \(<A, X^r, X^l, D^r, D^l, F, C>\). Here, \(A\) represents a set of agents. \(X^r\) is a set of variables that represent amounts of supplied or consumed resources in the nodes. \(X^l\) is a set of variables that represent amounts of transferred resources in the links. \(D^r\) and \(D^l\) are family sets of finite domains of variables in \(X^r\) and \(X^l\), respectively. \(F\) is a set of cost functions and \(C\) is a set of resource constraints.

Each agent \(i \in A\) in the RCDCOP corresponds to a node in the resource allocation problem. For the sake of simplicity, we use the notation of an agent and its corresponding node interchangeably. Additionally, a partial order on a set of agents is defined based on the feeder-tree rooted at a node. Based on the feeder-tree and the corresponding partial order, notations of parent agent \(p_i\) and set \(Ch_i\) of child agents are defined for each agent \(i\).

Agent \(i\) has a variable \(x^r_i \in X^r\) that represents the amount of supplied or consumed resources. \(i\) also has a set \(X^l_i \subset X^l\) of variables that represent the amount of transferred resources via \(i\). \(x^l_{i,j} \in X^l_i\) represents the amount of resources transferred from agent \(i\) to its child agent \(j\). Similarly, \(x^l_{p_i,i} \in X^l_i\) represents the amount of resources transferred from \(i\)’s parent agent \(p_i\) to \(i\). Agent \(i\) decides
the values of the variables except \( x_{p,i} \), whose value is determined by \( p_i \).

\( D^r_i \in D \) defines the domain of variable \( x^r_i \) for agent \( i \).

\( D^l_i \) contains possible values of \( r_i \). \( D^l_{i,j} \in D^l_i \) defines the domain of variable \( x^l_{i,j} \) for link \((i,j)\). \( D^l_{i,j} \) contains values in \( L_{i,j} \).

\( f_i(x^r_i) \in F \) is a cost function that corresponds to \( f_i(r_i) \) for node \( i \) in the resource allocation problem. Similarly, \( c_i \in C \) is a resource constraint for node \( i \). The resource constraint \( c_i \) and global cost function \( f(\mathcal{X}) \) for assignment \( \mathcal{X} \) for all variables in \( X^r \cup X^l \) are defined as follows:

\[
 c_i : x^r_i + \sum_{j \in N} x^l_{i,j} = 0 \quad (3)
\]

\[
 f(\mathcal{X}) = \sum_{i \in N} f_i(x^r_i) \quad (4)
\]

Here \( x^r_i \) takes a corresponding value in \( \mathcal{X} \). The optimal allocation \( \mathcal{X}^r \) minimizes \( f(\mathcal{X}) \) under the constraints.

As shown in Section V, the problem shown above is a variation of the RCDCOP in [6].

C. Computation of globally optimal solution

A solution method is applied to the RCDCOP defined in Subsection II-B. Basically, the method is a simple version of the solution method shown in [6]. We extracted substantial computation from the previous method to clarify the essentials of the algorithm. Also, the algorithm is slightly modified to meet our problem. A comparison of both studies is discussed in Section V.

The computation of the cost value is recursively defined. The optimal cost \( g^*_i(\{(x^r_{p,i},d_{p,i})\}) \) for assignment \((x^r_{p,i},d_{p,i})\) of a resource from \( i \)'s parent \( p_i \) and the subtree rooted at agent \( i \) is represented as follows:

\[
 g^*_i(\{(x^r_{p,i},d_{p,i})\}) = \min_{\mathcal{X}_i} g_i(\{(x^r_{p,i},d_{p,i})\} \cup \mathcal{X}_i) \quad (5)
\]

\[
 g_i(\{(x^r_{p,i},d_{p,i})\} \cup \mathcal{X}_i) = \delta_i(\{(x^r_{p,i},d_{p,i})\} \cup \mathcal{X}_i) \oplus \bigoplus_{j \in C_h,(x^l_{i,j},d_{i,j}) \in \mathcal{X}_i} g_j(\{(x^l_{i,j},d_{i,j})\}) \quad (6)
\]

\[
 \delta_i(\{(x^r_{p,i},d_{p,i})\} \cup \mathcal{X}_i) =
\begin{cases} 
 f_i(d_i) & \text{resource constraint } c_i \text{ is satisfied.} \\
 \infty & \text{otherwise} 
\end{cases} \quad (7)
\]

Here, \( \mathcal{X}_i \) denotes an assignment such that \( \{(x^r_i,d_i)\} \cup \bigcup_{j \in C_l}(\{(x^l_{i,j},d_{i,j})\},d_i \in D^l_i, d_{i,j} \in D^l_{i,j}, \oplus \) denotes an aggregation operator. While it is basically defined as a summation, we will add another operator in a later section.

In Equations (5), (6), and (7), for the sake of simplicity, it assumed that each agent is able to refer to cost values and assignments of other agents. However, in actual computation, the values are unknown until they are received from other agents via messages. To represent the unknown cost values, a lower limit value \( 0 \) and an upper limit value \( \infty \) are employed. As a result, a cost value is separated into a lower bound and an upper bound of the true value. We use notation \( lb(\cdot) \) and \( ub(\cdot) \) to denote the lower and upper bounds.

The algorithm is based on tree-search and dynamic programming. In the processing, the partial ordering of agents is employed. The processing consists of two phases: the computation of the cost values and the decision of the optimal assignments of the variables.

In the computation of the cost values, the globally optimal cost value is computed in a bottom-up manner. The tree-search repeats the computation for each solution using the boundaries of the cost values. In the root agent, the boundaries eventually converge to the optimal value. Then, the optimal assignments of the root agent’s variables are determined. Similarly, another optimal assignment is recursively determined in a top-down manner.

The pseudo code of the solver is shown in Figure 1. The processing employs the following two messages:

- **VALUE**: VALUE messages are sent from an agent to its child agents. The messages propagate assignments of partial solutions. In the final steps, they also carry flags that represent the termination of the agents.
- **COST**: COST messages are sent from an agent to its parent agent. The message is employed to send cost values for the current partial assignment of a parent agent’s variable.

\[ \text{fTR}_i \text{ and } \text{fPT}_r \] represent termination of agent \( i \) and \( i \)'s parent, respectively. Basically, its processing consists of...
the operations shown above. It is assumed that a tree is generated using preprocessing. The root agent initiates the search sending a VALUE message (line 10). Agents then repeatedly exchange VALUE and COST messages. When \(i\) receives a VALUE from its parent (line 11), assignment from \(i\)'s parent is updated. When \(i\) receives a COST from its child (line 13), \(g_{i,d}^j\) is updated. The conditions in lines 14 and 15 ensure monotonicity in convergence of boundaries when assignments of variables are asynchronously changed. After receiving messages, a condition for termination is checked (line 18). If the condition is not satisfied, the agent checks the boundaries of the current assignment of its own variable. Then, the agent changes the assignments of its own variables based on an appropriate search strategy (line 22). Here, we employ a best first search. When the root agent finds the convergence of the globally optimal cost value, the search terminates in a top-down manner.

### III. METHODS REDUCING UNEQUALNESS

#### A. Basic idea

In the problem and the solution method shown in Section II, only the globally optimal solution that minimizes the total cost value is considered. However, the shares of the cost values for all agents should be as equal as possible in several practical resource allocation problems. We therefore consider the equality of the agents in addition to the total cost value. A problem that considers two criteria is a multiple objective problem. While Pareto solutions are usually computed for general multiple objective problems, their computational cost is relatively high because of the large solution spaces. Thus, we instead treat the equality as an additional objective. Moreover, the shares of the total cost values are determined within the results of the computation of the globally optimal cost value. The solution space of the original optimization problem therefore does not increase. On the other hand, new problems are defined under the original optimal cost value.

For the equality, we consider the following criteria: Both values are minimized.

- variance of cost values
- maximum cost value

While the variance directly represents inequality, it requires the distribution of cost values. On the other hand, minimizing the maximum cost value means improving the worst cost of the agents. While computation of the minimum-maximum cost value is relatively easy, it does not assure equality. However, it does reduce the range of cost values in several cases.

In the following section, we present additional methods to employ these criteria.

#### B. Computation of variance

As addressed in Subsection III-A, to evaluate the variance, the distribution of cost values is necessary. Histograms of the cost values are therefore computed in the optimal cost computation phase. Let \(h\) denote a histogram. Histogram \(h\) is a table that represents a map from a cost value to the number of agents of the cost value.

Note that multiple histograms may exist for an assignment. The computation of set \(H_i^f(\{(x_{p,i}^1, d_{p,i}^1)\})\) of histograms that corresponds to \(g^f(\{(x_{p,i}^1, d_{p,i}^1)\})\) in Equation (5) is represented as follows:

\[
H_i^f(\{(x_{p,i}^1, d_{p,i}^1)\}) = \bigcup_{x_i^p} H_i(\{(x_{p,i}^1, d_{p,i}^1)\} \cup X_i^*)
\]

\[
H_i(\{(x_{p,i}^1, d_{p,i}^1)\} \cup X_i^*) = \{(\delta_i(\{(x_{p,i}^1, d_{p,i}^1)\} \cup X_i^*), 1)\} \otimes \bigotimes_{j \in Ch_i, (x_{i,j}^*, d_{i,j}^*) \in X_i^*} H_j^f(\{(x_{i,j}^*, d_{i,j}^*)\})
\]

In Equation (8), \(X_i^*\) denotes \(i\)'s assignment that corresponds to \(g^f(\{(x_{p,i}^1, d_{p,i}^1)\})\). Note that there may be multiple corresponding assignments. In Equation (9), \((a, b)\) denotes a part of a histogram representing that the count of \(a\) is \(b\). \(\otimes\) denotes an aggregation of sets of histograms.

In the aggregation, a new set of histograms that contains the total histograms for all combinations of histograms of \(i\) and \(i\)'s child agents is computed. Additionally, the same histograms are integrated.

Because the histogram is not used in the distributed tree-search, we only use true values of the histogram. For an upper bound of cost value except \(\infty\), corresponding histograms have been computed. Therefore, an agent is able to compute variances for its assignments when it determines the assignments.

In the phase of the decision of the optimal assignments, the variance of allocated cost values is considered and the assignments of the minimal variance are chosen. Root agent \(i\) chooses histogram \(h_i^{**}\) of minimal variance \(v_i^{**}\).

\[
v_i^{**} = \min_{X_i^*} v_i(X_i^*)
\]

\[
v_i(X_i^*) = \min_{h \in H_i(X_i^*)} \text{variance of } h
\]

The optimal assignment \(X_i^{**}\) that corresponds with \(h_i^{**}\) is also chosen. Moreover, histogram \(h_j^{**}\), which is contained in the aggregation of \(h_i^{**}\), is identified for each child agent \(j\). Agent \(i\) then sends its assignment \((x_{i,j}^{**}, d_{i,j}^{**}) \in X_i^{**}\) and \(h_j^{**}\) to each child node \(j\).

Non-root agent \(i\) receives \((x_{p,i}^{**}, d_{p,i}^{**})\) and \(h_i^{**}\) from its parent agent \(p_i\). Note that non-root agent \(i\) does not compute \(h_i^{**}\) because it is received from \(i\)'s parent agent. Agent \(i\) then chooses its \(X_i^{**}\) and corresponding histograms for each child. Similar computation is performed until it propagates to leaf agents.

#### C. Types of agents

When there are agents whose preferences differ, equality between different types of agents is not simply defined
using the single variance of their cost values. In that case, the equality should be separately evaluated for each type of agent. Here, we assume that the types of agents are globally identified. Partitioning of the agents is therefore not addressed in this study.

Let \( T \) denote a set of types of agents. To evaluate the inequality for each type of agent, the distribution of the cost values is computed for the type. The computation of histograms is naturally extended into tuples of histograms. Each agent \( i \) computes a set of tuples of histograms. A tuple \( h^T \) of histograms is represented as \((h^t, \ldots, h^{t'})\), s.t. \( t, \ldots, t' \in T \). A set of tuples of histograms that is computed in \( i \) for all related types is denoted as \( H^T_i \). When the root agent \( i \) chooses its optimal assignments, there are multiple objectives among types of agents. As a Pareto optimal value, we use the minimum summation of variance values for all types. The root agent \( i \) chooses tuple \( h^{T*} \) of histograms \( h_i^{T*} \) for each type \( t \) based on the following evaluation value \( v_i^{T*} \):

\[
v_i^{T*} = \min_{X^*_i} v_i^{T}(X^*_i) \tag{12}
\]

\[
v_i^{T}(X_i) = \min_{h^T \in H^T_i(X_i)} \sum_{h^T \in h^T} \text{variance of } h^T \tag{13}
\]

The optimal assignment \( X_i^{T*} \) is also chosen. Additionally, for each child agent \( j \), a tuple \( h_j^{T*} \) of histogram \( h_j^{T*} \) that is contained in the aggregation of \( h_i^{T*} \) is identified. Agent \( i \) then sends its assignment \( (x_{i,j}^{*}, d_{i,j}^{*}) \) to \( h_j^{T*} \) to each child node \( j \). The decision of the optimal assignments in non-root agents is similarly generalized for multiple types.

While the histogram is useful to compute detailed statistical criteria, its number of combinations exponentially increases with the number of cost values and the number of types in the worst case. To avoid the combinational explosion, as a heuristic, we can leave \( k \) tuples whose summation of variance values is smaller than others.

### D. Using center values

As information that needs low computational cost, the minimum summation of the cost values is available. Because each agent knows its own cost values and several cost values for each subtree, the agent is able to compute the allocation of cost values among the agent and its child agents for each known optimal assignment. Choosing relatively even allocation of the cost values, the optimal assignments are optimistically determined.

The minimum summation is already computed for the main objective. On the other hand, it is necessary to separate the minimum summation from the main objective in the case of multiple types of agents shown in Subsection III-C. Here, the case of multiple types is shown.

In the computation of cost values, agent \( i \) calculates the tuple \( g_i^{T*} \) of the optimal cost values \( g_i^{T*} \) for assignment \((x_{p_i,i}^t, d_{p_i,i}^t)\) of resource from \( i \)'s parent \( p_i \) and the subtree rooted at agent \( i \). The computation is the same as that of Equations (5), (6), and (7) except that \( \delta_i(\{\{x_{p_i,i}^t, d_{p_i,i}^t\} \cup X_i\}) \) is evaluated for only \( i \)'s type. In addition, the minimization in Equation (5) should be replaced by a union to compute a set of tuples similar to Equation (8). However, we arbitrarily choose one of tuples to reduce complexity.

In the decision of the optimal assignments, agent \( i \) enumerates optimal assignments that meet the main objective. For each assignment \( X_i^{t*} \), a vector \( g_i^{X_i^t} \) of cost values is defined as follows:

\[
g_i^{X_i^t} = [\delta_i(\{\{x_{p_i,i}^t, d_{p_i,i}^t\} \cup X_i^t\}), g_j^t(\{\{x_{i,j}, d_{i,j}^t\}\}), \ldots, g_j^t(\{\{x_{i,j}, d_{i,j}^t\}\})] \tag{14}
\]

Here, \( i \) denotes elements of \( i \)'s child agents in \( Ch_i \), \( t \) denotes elements of set \( T \) of types. \((x_{p_i,i}^t, d_{p_i,i}^t)\) is the optimal assignment of \( i \)'s parent agent. \((x_{i,j}, d_{i,j}^t)\) is a part of the assignment of \( X_i^{t*} \).

For all enumerated vectors, vectors \( g^\perp \) and \( g^\top \) are computed. While elements of \( g^\perp \) are minimum values for all vectors, elements of \( g^\top \) are maximum values. Then, the vector \( g^\top \) whose elements are mean values of elements in \( g^\perp \) and \( g^\top \) is computed. Finally, assignment \( \hat{X}_i^{t*} \) is chosen so that \( ||g_i^{X_i^t} - g^\top || \) is minimized. The aim of the central values is to eliminate the influence of the majority of assignments that have the same cost values.

### E. Computation of maximum cost value

As additional sub-objective of the problem, minimization of the maximum cost value for all agents is able to be applied. In actual resource allocation, there may be cases in which improvement of the worst cost is preferred. While the sub-objective improves the worst cost value of the agents, it may not improve the variance of cost values. On the other hand, it works with other criteria. Here, we employ the minimum-maximum cost value as a sub-objective next to the main objective.

In the computation of cost values, agent \( i \) calculates minimum-maximum value \( m_i(\{\{x_{p_i,i}^t, d_{p_i,i}^t\} \cup X_i\}) \) for assignment \((x_{p_i,i}^t, d_{p_i,i}^t)\) of resource from \( i \)'s parent \( p_i \), its own assignment \( X_i \), and the sub-tree rooted at agent \( i \). The computation resembles Equations (6) and (7). To compute the maximum cost value, operator \( \oplus \) in Equation (7) is defined as a maximum function.

Moreover, the aggregation in Equation (5) is modified to combine the main objective and the sub-objective. Here, a binary relation \( < \) on a set of tuples \((a, b)\) is defined. For
(a, b) and (a’, b’), (a, b) < (a’, b’) if and only if a < a’ ∨
(a = a’ ∧ b < b’). Applying the relation < to the minimum
function, Equation (5) is modified as follows:
\[ \min(g_i(\{(x^i_{p^i,i}, d_{p^i,i})\} \cup X_i), m_i(\{(x^i_{p^i,i}, d_{p^i,i})\} \cup X_i)) \]

Other parts of the computation are generalized using the
tuples of \( g_i \) and \( m_i \).

F. Minimum-maximum cost value as main objective

In Subsection III-E, we introduced a combination of a
main objective and a sub-objective. It is easy to modify the
combination so that the objectives are exchanged. In that
case, Equation (15) is modified as follows:
\[ \min(m_i(\{(x^i_{p^i,i}, d_{p^i,i})\} \cup X_i), g_i(\{(x^i_{p^i,i}, d_{p^i,i})\} \cup X_i)) \]

The main objective is now different. On the other hand,
there is a possibility that the modification may reduce the
search because the number of maximum cost values is
usually less than the number of summation values.

There is an issue of non-monotonicity on the combination
of objectives. Consider tuple \((lb(m), ub(m), lb(g), ub(g))\) of
cost values. \((1, 3, 20, 20)\) and \((2, 2, 10, 10)\) satisfy \(lb(m) \leq
ub(m)\) and \(lb(g) \leq ub(g)\). When we define \(lb(m, g) =
(lb(m), lb(g))\) and \(ub(m, g) = (ub(m), ub(g))\), we choose
\(lb(m, g) = (1, 20)\) and \(ub(m, g) = (2, 10)\) as minimum
values using relation <. Those values do not satisfy \(lb(g) \leq
ub(g)\). That often results \(lb(m, g) = ub(m, g) > ub(g)\)
in later aggregations. We consider that as a closed boundary
and use the upper bound as the true value.

G. Embedding to the basic method

There are a few issues about embedding the proposed
methods to the basic solver shown in Figure 1.

The proposed methods are add-on processing. Therefore,
their computation extends the original processing or per-
forms beside the original processing. All of the additional
information that is passed between agents is carried with
the information of the original messages. The modifications
of the messages are only extensions of their payloads.

In actual computation, a cost value for the main objective
is separated into lower and upper bounds. The values in
the proposed methods shown in Subsections III-B, III-C and
III-D are computed with the upper bound of the cost values
for the main objective. The computation of the proposed
methods does not affect the computation of the original
bounds. Additionally, the proposed methods only need upper
bounds of their values because the (true) upper bound values
are always computed when the lower and upper bound cost
values for the main objective converge. Methods shown in
Subsections III-E and III-F extend the main objective.

H. Correctness and complexity

The basic solver is a simple version of the conven-
tional algorithm shown in [6]. Therefore, its correctness
and complexity basically resemble the conventional algorithm.
Time complexity exponentially grows with the number of
variables in the worst case. Space complexity in each agent
exponentially grows with the number of its variables. There-
fore, it is important to limit the number of branches (i.e. the
number of variables for child nodes) and the domains of
variables.

As addressed in Subsection III-A, the proposed method
is performed under the results of the computation of the
cost values for the main objectives. Therefore, the proposed
methods provide the best effort to choose one of the known
candidates of the optimal solution of the original problems.
On the other hand, the proposed method does not increase
the search of the basic solver.

The number of sets of histograms for each assignment
exponentially grows with the number of cost values in
the worst case. Similarly, the number of types of agents
exponentially increases the number of combinations of cri-
tera. The complexity of computing the center values and
minimum-maximum values are in basically the same order
as the original computation of cost values.

IV. Evaluation

The proposed methods are experimentally evaluated using
example problems. The aim of the experiments is to illustrate
several behaviors of the proposed methods. The problems
contain several types of nodes including source and sink
nodes. We designed nodes with parameters \((rl, ru, rt, nt)\). \(rl\)
and \(ru\) are the minimum and maximum amount of required
resource, respectively. \(rt\) is the most preferred amount of
the resource. \(nt\) is the number of nodes for each type. The cost
function is defined as \(f_i(v) = |rt - r|\). The same capacity \(t_x\)
is set for all links. The structure of the networks is a linear
or binary tree. All problem instances are feasible.

Several methods from the following combinations of
methods are compared. The objectives of the problems are
the minimum summation (sum) or the minimum-maximum
value (max) of cost values. These are used as the main
objective or the sub-objective. The methods to choose the
optimal solution are as follows:

- **grd**: in the decision of the optimal cost, each agent
  chooses its own assignments so that its own cost value
  is minimal.
- **hst**: the optimal assignments are chosen using his-
  tograms shown in Subsection III-B. hstk stores at most
  \(k\) histograms.
- **cnt**: the optimal assignments are chosen using center
  values shown in Subsection III-D.

In addition, types of agents are distinguished (t) or not.
To evaluate the number of iterations, the simulation of the
In the results of sum-t-hst, the variance is relatively small for values, the variance of each type of agents is relatively large. While sum-hst decreases global variance and cost values of nodes are shown for each algorithm. Average/variance cost values are also categorized for each type of nodes.

| \( \ell_t \) | alg. | \#iter. | max. cost | ave. cost | var. cost | ave. cost (type) | var. cost (type) | 3 | \#iter. | max. cost | ave. cost | var. cost | ave. cost (type) | var. cost (type) | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sum-grd | 167 | 0 | 4,20 | 1,55 | 12,6 | 1,23 | 1,72 | 3,92 | 0,71 | 1,75 | 3,64 | 0 | 0,32 | 1,83 | 1,42 | 0,08 | 0,79 | 1,25 | 0,35 | 0,80 | 1,86 |
| sum-hst | 167 | 9966 | 3,08 | 1,54 | 0,84 | 1,85 | 1,40 | 1,53 | 1,55 | 0,41 | 0,50 | 364 | 2,129 | 0,83 | 0,17 | 0,08 | 1,03 | 1,01 | 0 | 0,01 | 0,02 |
| sum-t-hst | 167 | 89756 | 3,12 | 1,54 | 1,14 | 1,85 | 1,13 | 1,79 | 1,53 | 0,25 | 0,42 | 364 | 683324 | 1,80 | 0,83 | 0,77 | 0,08 | 0,76 | 1,28 | 0 | 0,03 | 0,03 |
| sum-t-hst10 | 167 | 31939 | 3,12 | 1,54 | 1,14 | 1,85 | 1,13 | 1,79 | 1,53 | 0,25 | 0,42 | 364 | 844877 | 1,80 | 0,83 | 0,77 | 0,08 | 0,76 | 1,28 | 0 | 0,03 | 0,03 |
| sum-t-cnt | 167 | 0 | 3,34 | 1,54 | 1,07 | 1,85 | 1,14 | 1,79 | 1,73 | 0,32 | 0,83 | 364 | 0 | 2,04 | 0,83 | 0,55 | 0,08 | 0,74 | 1,30 | 0,03 | 0,17 | 0,59 |
| max-sum-t-hst | 169 | 58446 | 3,08 | 1,54 | 1,03 | 1,85 | 1,22 | 1,71 | 1,53 | 0,29 | 0,41 | 364 | 845799 | 1,08 | 0,83 | 0,19 | 0,08 | 1,04 | 1,00 | 0 | 0,03 | 0,03 |
| max-sum-t-hst10 | 169 | 21640 | 3,08 | 1,54 | 1,03 | 1,85 | 1,22 | 1,71 | 1,53 | 0,29 | 0,41 | 364 | 57401 | 1,08 | 0,83 | 0,19 | 0,08 | 1,04 | 1,00 | 0 | 0,03 | 0,03 |
| max-sum-t-cnt | 169 | 0 | 3,08 | 1,54 | 0,94 | 1,85 | 1,27 | 1,65 | 1,53 | 0,39 | 0,66 | 364 | 0 | 1,08 | 0,83 | 0,17 | 0,08 | 1,01 | 1,03 | 0 | 0,01 | 0,02 |

Table II

Binary tree of 50 nodes (3 \( \times 25 \) (leaf) = 100 variables) — \( \ell_t = 0 \): \((-14,0,-14), 1: (0,4,4,20), 2: (0,6,6,20)\)

The number of iterations (cycles), the number of histograms, and cost values of nodes are shown for each algorithm. Average/variance cost values are also categorized for each type of nodes.

The variance of the costs is relatively large in the results of sum-grd, because agents greedily choose their own assignments. While sum-hst decreases global variance values, the variance of each type of agents is relatively large. In the results of sum-t-hst, the variance is relatively small for several types (e.g. \( \ell_t=3 \), type=1). On the other hand, the difference between types is relatively large.

max-sum-* needs fewer numbers of iterations because it handles fewer combinations of cost values. In comparison with sum-t-hst*, max-sum-t-hst* and max-sum-t-hst* reduce the total number of histograms because they reduce the number of (locally) optimal solutions.

In results of \(*-t-cnt\), the variance of costs for each type is relatively larger than that of \(*-t-hst\) because cnt only use the summation of cost values. While \(*-t-hst10\) stores the best 10 histograms per agent, the quality of results resembles that of \(*-t-hst\).

Table II shows the results for the networks of binary trees. Basically, the results resemble the case of the linear networks while the number of variables and the size of variables’ domains are relatively large.

In total, we can see that employing histograms and distinguishing types are effective for reducing the variance while requiring high computational costs. Moreover, the number of histograms is able to be limited without significant lack of the effects. The minimum-maximum cost value as the main objective reduces iterations. Also, the minimum-maximum cost value as objectives reduces the number of histograms.

In the case where the capacity of the link is relatively tight, the pre-defined types of agents mismatch with differences of actual limitations among the agents. To overcome that, more studies to identify actual types will be necessary.

V. RELATED WORKS

The problem shown above is a variation of the RCDCOP in [6]. In the previous work, multiple types of resources are allocated to agents. While a resource constraint is originally defined as a global constraint for each type of resource, the global constraint is decomposed into constraints using variables that represent shares of the resource. The representation of the decomposed constraints is basically the same for our representation. There are minor differences from the previous work: 1) Only one type of resource is defined. 2) We limit the resource constraints from inequality constraints to equality constraints. 3) The initial location of the resource is defined. Also, it can divide the resource into multiple source nodes. 4) As a result of the distributed source nodes in the feeder-trees, the variables take negative values...
if necessary. 5) Capacities of the links are defined. 6) Instead of binary cost functions, unary cost functions are defined to represent preferences of nodes.

The proposed method simultaneously computes different criteria. It appears that the approach also relates multiple objective DCOPs [9], while we avoided exact multiple objective problems that generate high computational costs to compute the Pareto front. How to employ the proposed criteria in the framework of multiple objective DCOPs will be investigated in a future work.

In our study, agents are partitioned based on their type. How to partition the agents automatically is possibly related to other studies, including the Coalition Structure Generation [10].

We employed a basic solver to clarify the essentials of the computation. There are several efficient methods for these types of solvers [2], [11], [12]. Also, there are other solvers based on the pseudo-trees [3], [13]. The proposed methods can be applied with these methods.

VI. CONCLUSION

In this work, we focused on a resource sharing problem that is motivated from a power supply network containing distributed sources. The resource allocation problem was formalized as a resource constrained distributed constraint optimization problem in which the cost functions represent preferences of agents on resource use. We then proposed several methods to reduce inequality of cost values that are allocated to agents. The experimental result shows the effects of the proposed methods. Employing histograms and distinguishing types are effective for reducing the variance while requiring high computational costs. The number of histograms is able to be limited without significant lack of the effects. The minimum-maximum cost value as the main objective reduces the number of iterations and histograms.

Our future work will include investigations of the proposed criteria in the framework of multiple objective DCOPs, partitioning of agents to identify their types automatically, and studies on more practical problems.

ACKNOWLEDGMENT

This work was supported in part by Grant-in-Aid for Young Scientists (B), 22700144 and the Artificial Intelligence Research Promotion Foundation.

REFERENCES


