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. The 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 preprocessing.

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 $\langle N, L, R, F, \mathcal{L}\rangle$, 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:

- $\mathcal{L}_{i,j}$: $\mathcal{L}_{i,j} \in \mathcal{L}$ is a finite set of amounts of resource that is transferred through link (i, j). Amount $l \in \mathcal{L}_{i,j}$ is a finite value such that $-l^c_{i,j} \le l \le l^c_{i,j}$, where $l^c_{i,j}$ is the capacity of link (i, j). The positive and negative values in $\mathcal{L}_{i,j}$ 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.

$$r_i + \sum_{(i,j)\in L^i} l_{i,j} = 0 \qquad (1)$$

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

$$f(\mathcal{R}) = \sum_{i\in N} f_i(r_i) \qquad (2)$$

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 $\langle A, X^r, X^l, D^r, D^l, F, C\rangle$. 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 $x^l_{p_i,i}$ whose value is determined by p_i.

$D^r_i \in D^r$ defines the domain of variable x^r_i for agent i. D^r_i contains possible values of r_i. $D^l_{i,j} \in D^l$ defines the domain of variable $x^l_{i,j}$ for link (i,j). $D^l_{i,j}$ contains values in $\mathcal{L}_{i,j}$.

$f_i(x^r_i) \in F$ 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:

$$c_i:\ x^r_i + \sum_{j \in X^l_i} x^l_{i,j} = 0 \tag{3}$$

$$f(\mathcal{X}) = \sum_{i \in N} f_i(x^r_i) \tag{4}$$

Here x^r_i takes a corresponding value in $\mathcal{X}$. The optimal allocation $\mathcal{X}^*$ 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 $g^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\})$ for assignment $(x^l_{p_i,i}, d_{p_i,i})$ of a resource from i's parent p_i and the subtree rooted at agent i is represented as follows:

$$g^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\}) = \min_{\mathcal{X}_i} g_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X}_i) \tag{5}$$

$$g_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X}_i) = \delta_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X}_i) \oplus \bigoplus_{j \in Ch_i, (x^l_{i,j}, d_{i,j}) \in \mathcal{X}_i} g^*_j(\{(x^l_{i,j}, d_{i,j})\}) \tag{6}$$

$$\delta_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X}_i) = \begin{cases} f_i(d_i) & \text{resource constraint } c_i \text{ is satisfied.} \\ \infty & \text{otherwise} \end{cases} \tag{7}$$

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

```
Main(){
  Initialize().
  until $forever$ do { // message loop
    until receive loop is broken do { receive messages. }
    if $\neg fTr_i \wedge (p_i = \epsilon \vee d_{p_i,i} \neq \epsilon)$ then { Maintenance(). }
  } }
Initialize(){
  $d_{p_i,i} \leftarrow \epsilon$. $\mathcal{X}_i \leftarrow \phi$. $fTr_i \leftarrow$ F. $fPTr_i \leftarrow$ F.
  for all $j \in Ch_i$ and $d \in D^l_{i,j}$ do { $g^*_{j,d} \leftarrow \phi$. }
  if $p_i = \epsilon$ then { send (VALUE$_{\epsilon \to i}$, $\epsilon$, T) to $i$. } } // initiate
Receive(VALUE$_{p_i \to i}$, $d$, $fTr$){ // receive parent's assignment
  $d_{p_i,i} \leftarrow s$. $fPTr_i \leftarrow fTr$. }
Receive(COST$_{i \leftarrow j}$, $d$, $g^*$){ // receive subtree's cost value
 if $lb(g^*) > lb(g^*_{j,d})$ then { $lb(g^*_{j,d}) \leftarrow lb(g^*)$. }
 if $ub(g^*) < ub(g^*_{j,d})$ then { $ub(g^*_{j,d}) \leftarrow ub(g^*)$. } }
Maintenance(){
  update $g_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X})$ and $g^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\})$.
  if $fPTr_i \wedge lb(g^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\})) = ub(g^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\}))$
  then { choose an assignment $\mathcal{X}^*_i$ // termination
            that gives $ub(g^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\}))$.
        $\mathcal{X}_i \leftarrow \mathcal{X}^*_i$. $fTr_i \leftarrow$ T. }
  else{ update $X_i$ based on a search strategy. } // search
  for all $j \in Ch_i$ do { send (VALUE$_{i \to j}$, $d_{i,j}$, $fTr_i$) to $j$
    where $(x^l_{i,j}, d_{i,j}) \in \mathcal{X}_i$. } // send own assignment
  if $p_i \neq \epsilon$ then { send (COST$_{p_i \leftarrow i}$, $d_{p_i,i}$, $g^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\})$)
    to $p_i$. } } // send subtree's cost value
```

Figure 1. basic solver

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.

fTr_i and $fPTr_i$ 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), $g^*_{j,d}$ 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 $H^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\})$ of histograms that corresponds to $g^*(\{(x^l_{p_i,i}, d_{p_i,i})\})$ in Equation (5) is represented as follows:

$$H^*_i(\{(x^l_{p_i,i}, d_{p_i,i})\}) = \bigcup_{\mathcal{X}^*_i} H_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X}^*_i) \quad (8)$$

$$\begin{aligned} &H_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X}_i) = \\ &\{\{\langle \delta_i(\{(x^l_{p_i,i}, d_{p_i,i})\} \cup \mathcal{X}_i), 1\rangle\}\} \otimes \\ &\bigotimes_{j \in Ch_i, (x^l_{i,j}, d_{i,j}) \in \mathcal{X}_i} H^*_j(\{(x^l_{i,j}, d_{i,j})\}) \end{aligned} \quad (9)$$

In Equation (8), $\mathcal{X}^*_i$ denotes i's assignment that corresponds to $g^*(\{(x^l_{p_i,i}, d_{p_i,i})\})$. Note that there may be multiple corresponding assignments. In Equation (9), $\langle a, b\rangle$ 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 ∞, 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.

$$v^{**}_i = \min_{\mathcal{X}^*_i} v_i(\mathcal{X}^*_i) \quad (10)$$

$$v_i(\mathcal{X}_i) = \min_{h \in H_i(\mathcal{X}_i)} \text{variance of } h \quad (11)$$

The optimal assignment $\mathcal{X}^{**}_i$ 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 $(x^l_{i,j}, d^{**}_{i,j}) \in \mathcal{X}^{**}_i$ and h^{**}_j to each child node j.

Non-root agent i receives $(x^l_{p_i,i}, d^{**}_{p_i,i})$ 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 $\mathcal{X}^{**}_i$ 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 $(h^t, \cdots, h^{t'})$ s.t. $t, \cdots, t' \in T$. A set of tuples of histograms that is computed in i for all related types is denoted as H_i^T. 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_i^{T**} of histograms h_i^{t**} for each type t based on the following evaluation value v_i^{T**}:

$$v_i^{T**} = \min_{\mathcal{X}_i^*} v_i^T(\mathcal{X}_i^*) \tag{12}$$

$$v_i^T(\mathcal{X}_i) = \min_{h^T \in H_i^T(\mathcal{X}_i)} \sum_{h^t \in h^T} \text{variance of } h^t \tag{13}$$

The optimal assignment $\mathcal{X}_i^{**}$ 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 $(x_{i,j}^l, d_{i,j}^{**}) \in \mathcal{X}_i^{**}$ and 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^{T*} of the optimal cost values $g_i^{t*}(\{(x_{p_i,i}^l, d_{p_i,i})\})$ s.t. $t \in T$ for assignment $(x_{p_i,i}^l, d_{p_i,i})$ 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 $\delta_i(\{(x_{p_i,i}^l, d_{p_i,i})\} \cup \mathcal{X}_i)$ 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 $\mathcal{X}_i^*$, a vector $\mathbf{g}_{\mathcal{X}_i^*}$ of cost values is defined as follows:

$$\mathbf{g}_{\mathcal{X}_i^*} = \begin{bmatrix} \delta_i(\{(x_{p_i,i}^l, d_{p_i,i}^*)\} \cup \mathcal{X}_i^*) \\ g_j^t(\{(x_{i,j}^l, d_{i,j}^*)\}) \\ \vdots \\ g_{j'}^t(\{(x_{i,j'}^l, d_{i,j'}^*)\}) \\ \vdots \\ g_j^{t'}(\{(x_{i,j}^l, d_{i,j}^*)\}) \\ \vdots \\ g_{j'}^{t'}(\{(x_{i,j'}^l, d_{i,j'}^*)\}) \end{bmatrix} \tag{14}$$

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

For all enumerated vectors, vectors $\mathbf{g}^\perp$ and $\mathbf{g}^\top$ are computed. While elements of $\mathbf{g}^\perp$ are minimum values for all vectors, elements of $\mathbf{g}^\top$ are maximum values. Then, the vector $\mathbf{g}^c$ whose elements are mean values of elements in $\mathbf{g}^\perp$ and $\mathbf{g}^\top$ is computed. Finally, assignment $\widetilde{\mathcal{X}}_i^{**}$ is chosen so that $||\mathbf{g}_{\mathcal{X}_i^*} - \mathbf{g}^c||$ 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 $m_i(\{(x_{p_i,i}^l, d_{p_i,i})\} \cup \mathcal{X}_i)$ for assignment $(x_{p_i,i}^l, d_{p_i,i})$ of resource from i's parent p_i, its own assignment $\mathcal{X}_i$, and the sub-tree rooted at agent i. Basically, 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 $a < a' \vee (a = a' \wedge b < b')$. Applying the relation $<$ to the minimum function, Equation (5) is modified as follows:

$$(g_i^*(\{(x_{p_i,i}^l, d_{p_i,i})\}), m_i^*(\{(x_{p_i,i}^l, d_{p_i,i})\})) = \min_{\mathcal{X}_i}(g_i(\{(x_{p_i,i}^l, d_{p_i,i})\} \cup \mathcal{X}_i), m_i(\{(x_{p_i,i}^l, d_{p_i,i})\} \cup \mathcal{X}_i)) \quad (15)$$

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:

$$(m_i^*(\{(x_{p_i,i}^l, d_{p_i,i})\}), g_i^*(\{(x_{p_i,i}^l, d_{p_i,i})\})) = \min_{\mathcal{X}_i}(m_i(\{(x_{p_i,i}^l, d_{p_i,i})\} \cup \mathcal{X}_i), g_i(\{(x_{p_i,i}^l, d_{p_i,i})\} \cup \mathcal{X}_i)) \quad (16)$$

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 $\langle lb(m), ub(m), lb(g), ub(g)\rangle$ of cost values. $\langle 1, 3, 20, 20\rangle$ and $\langle 2, 2, 10, 10\rangle$ 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 $lb(m) = ub(m) \wedge lb(g) > ub(g)$ 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 performs 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 conventional 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. Therefore, 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 criteria. 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 $\langle rl, ru, rt, nt\rangle$. 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 $f_i(r) = |rt - r|$. The same capacity l_c 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 histograms shown in Subsection III-B. hst$k$ 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

Table I
LINEAR NETWORKS OF 15 NODES $(2 \times 14 + 1$ (LEAF) $= 29$ VARIABLES) — TYPES = 0: $\langle -6, 0, -6, 3 \rangle$, 1: $\langle 0, 2, 2, 6 \rangle$, 2: $\langle 0, 3, 3, 6 \rangle$
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.

l_c	3											12										
alg.	#iter.	#hist.	max.	ave.	var.	ave. cost (type)			var. cost (type)			#iter.	#hist.	max.	ave.	var.	ave. cost (type)			var. cost (type)		
			cost	cost	cost	0	1	2	0	1	2			cost	cost	cost	0	1	2	0	1	2
sum-grd	167	0	4.40	1.54	2.06	1.85	1.21	1.72	3.92	0.73	1.75	364	0	3.12	0.83	1.42	0.08	0.79	1.25	0.32	0.80	1.86
sum-hst	167	9966	3.08	1.54	0.84	1.85	1.40	1.53	1.53	0.41	0.50	364	21297	1.08	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	0.003	0.003
sum-t-hst10	167	31939	3.12	1.54	1.14	1.85	1.13	1.79	1.53	0.25	0.42	364	84487	1.80	0.83	0.77	0.08	0.76	1.28	0	0.003	0.003
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
sum-max-t-hst	169	58446	3.08	1.54	1.03	1.85	1.22	1.71	1.53	0.29	0.41	361	845799	1.08	0.83	0.19	0.08	1.04	1.00	0	0.003	0.003
sum-max-t-hst10	169	21640	3.08	1.54	1.03	1.85	1.22	1.71	1.53	0.29	0.41	361	57401	1.08	0.83	0.19	0.08	1.04	1.00	0	0.003	0.003
sum-max-t-cnt	169	0	3.08	1.54	0.94	1.85	1.27	1.65	1.53	0.39	0.66	361	0	1.08	0.83	0.17	0.08	1.01	1.03	0	0.01	0.02
max-sum-t-hst	124	32851	3.08	1.54	1.03	1.85	1.22	1.71	1.53	0.29	0.41	230	262341	1.08	0.83	0.19	0.08	1.04	1.00	0	0.003	0.003
max-sum-t-hst10	124	15852	3.08	1.54	1.03	1.85	1.22	1.71	1.53	0.29	0.41	230	34443	1.08	0.83	0.19	0.08	1.04	1.00	0	0.003	0.003
max-sum-t-cnt	124	0	3.08	1.54	0.94	1.85	1.27	1.65	1.53	0.39	0.66	230	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 $(3 \times 25 + 25$ (LEAF) $= 100$ VARIABLES) — TYPES = 0: $\langle -14, 0, -14, 10 \rangle$, 1: $\langle 0, 4, 4, 20 \rangle$, 2: $\langle 0, 6, 6, 20 \rangle$
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.

l_c	7											28										
alg.	#iter.	#hist.	max.	ave.	var.	ave. cost (type)			var. cost (type)			#iter.	#hist.	max.	ave.	var.	ave. cost (type)			var. cost (type)		
			cost	cost	cost	0	1	2	0	1	2			cost	cost	cost	0	1	2	0	1	2
sum-grd	202	0	12.54	3.17	9.72	4.91	2.15	3.31	20.17	3.36	7.55	661	0	7.20	1.33	5.39	0.34	1.25	1.92	3.44	3.16	7.29
sum-t-hst10	202	70249	8.48	3.17	5.09	4.91	2.05	3.41	10.44	1.16	2.00	661	235190	3.12	1.33	1.66	0.34	1.39	1.78	0.76	0.21	0.31
sum-t-cnt	202	0	10.02	3.17	5.83	4.91	2.11	3.34	12.69	1.53	3.47	661	0	4.58	1.33	1.79	0.34	1.29	1.88	0.86	0.90	2.14
sum-max-t-hst10	203	52677	8.48	3.17	4.60	4.91	2.34	3.12	10.44	1.40	1.75	660	174293	2.54	1.33	0.83	0.34	1.50	1.67	0.76	0.25	0.30
sum-max-t-cnt	203	0	8.48	3.17	5.15	4.91	2.32	3.14	10.71	1.86	2.84	660	0	2.54	1.33	0.96	0.34	1.55	1.62	0.76	0.57	0.62
max-sum-t-hst10	181	47203	8.48	3.17	4.61	4.91	2.34	3.12	10.44	1.40	1.75	577	151821	2.54	1.33	0.83	0.34	1.50	1.67	0.76	0.25	0.30
max-sum-t-cnt	180	0	8.48	3.17	5.17	4.91	2.31	3.14	10.71	1.89	2.87	577	0	2.54	1.33	0.96	0.34	1.55	1.62	0.76	0.58	0.62

solution method is synchronized for each cycle. In a cycle, each agent processes messages in its receiving queue. The agent also assigns messages to its sending queue. At the end of the cycle, the messages are transferred. For each setting, the results of 50 instances are averaged.

Table I shows the results for the linear networks. 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 cost values 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. l_c=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 sum-max-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

[1] R. Mailler and V. Lesser, “Solving distributed constraint optimization problems using cooperative mediation,” in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004, pp. 438–445.

[2] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo, “Adopt: Asynchronous distributed constraint optimization with quality guarantees,” *Artificial Intelligence*, vol. 161, no. 1-2, pp. 149–180, 2005.

[3] A. Petcu and B. Faltings, “A scalable method for multiagent constraint optimization,” in *19th International Joint Conference on Artificial Intelligence*, 2005, pp. 266–271.

[4] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg, “Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks,” *Artificial Intelligence*, vol. 161, no. 1-2, pp. 55–87, 2005.

[5] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham, “Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling,” in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004, pp. 310–317.

[6] T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, and H. Matsuo, “Resource constrained distributed constraint optimization with virtual variables,” in *23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 120–125.

[7] A. Kumar, B. Faltings, and A. Petcu, “Distributed constraint optimization with structured resource constraints,” in *AAMAS ’09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 923–930.

[8] S. Thiebaux and M. odile Cordier, “Supply restoration in power distribution systems - a benchmark for planning under uncertainty,” in *In Pre-Proceedings of the 6th European Conference on Planning (ECP-01*, 2001, pp. 525–532.

[9] F. M. Delle Fave, R. Stranders, A. Rogers, and N. R. Jennings, “Bounded decentralised coordination over multiple objectives,” in *10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, 2011, pp. 371–378.

[10] S. Ueda, A. Iwasaki, M. Yokoo, M.-C. Silaghi, K. Hirayama, and T. Matsui, “Coalition structure generation based on distributed constraint optimization,” in *AAAI*, 2010.

[11] M. C. Silaghi and M. Yokoo, “ADOPT-ing: unifying asynchronous distributed optimization with asynchronous backtracking,” *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 2, pp. 89–123, 10 2009.

[12] T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, and H. Matsuo, “Directed soft arc consistency in pseudo trees,” in *AAMAS ’09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 1065–1072.

[13] A. Petcu and B. Faltings, “O-DPOP: An algorithm for Open/Distributed Constraint Optimization,” in *National Conference on Artificial Intelligence*, 2006, pp. 703–708.