Reducing the Number of Samples in Distributed Cooperative Solution Method for Resource Supply Networks

Toshihiro Matsui, Masayuki Kaneko, Yuho Takama and Hiroshi Matsuo
Nagoya Institute of Technology
Gokiso-Cho, Showa-ku, Nagoya, Aichi, 466-8555, Japan
{matsui.t@, {kenko, takama}@matlab.,matsuo@}nitech.ac.jp

***Abstract*—Distributed Constraint Optimization Problems (DCOPs) are applied to resource allocation problems in resource supply networks. In previous studies, distributed cooperative solution methods based on feeder trees have been utilized. However, in most cases with resource supply networks, the size of variable's domains in the problems is very large, since the variables originally take continuous values. This is critical even if the networks are trees because it increases the number of combinations. Therefore, sampling of solutions is necessary to restrict the size of the problems. In this study, we propose methods to reduce the number of samples for resource allocation problems of resource supply networks. To maintain the feasibility with the samples, boundaries for the resource amount and cost values were introduced. With the proposed methods, the size of problems is reduced while the methods keep relatively better feasibility and quality of the solutions.**

I. Introduction

Distributed resource allocation systems, including power supply networks, have been studied as application domains of multiagent systems. In autonomous distributed networks, cooperative optimization methods are necessary to solve the problem of resource allocation. The Distributed Constraint Optimization Problem (DCOP) is a fundamental framework of cooperative problem solving in multiagent systems [1], [2], [3], [4]. The states of agents and the relationships between agents are formalized into a constraint optimization problem, which is distributed on the multiagent systems. Several types of distributed cooperative algorithms are employed for the DCOPs. The representation of DCOPs and corresponding solution methods can also be extended to address specific issues.

Resource allocation problems motivated by the power supply networks of smart-grid systems have been studied as distributed optimization problems. In the supply networks, resources that are initially distributed among source nodes have to be shared among all nodes. In related works [5], [6], dedicated classes of DCOPs and solvers were proposed for resource allocation on feeder trees. These studies are considered as a variation of Resource Constrained DCOP (RCDCOP) [7], which is a dedicated class of problems where shared resources are represented as global constraints that can be decomposed into agents. Their solution methods are based on pseudo trees for the problems.

While several studies represent these problems as discrete optimization problems [5], [6], the size of the variable's domains in the problems is very large in most cases because it originally takes continuous values. This is a critical issue, even if the networks are trees, since it increases the number of combinations. Therefore, sampling of solutions is necessary to restrict the size of the problems.

There are studies on DCOPs with continuous variables based on sampling, interpolation and numeric techniques [8], [9]. On the other hand, for resource allocation problems, a specific aggregation of variable values is employed to evaluate constraints of resources. In addition, since resource constraints are hard constraints, the feasibility is important even if the problem is approximated with samples.

In this study, we propose methods to reduce the number of samples for the resource allocation problems in resource supply networks. To maintain the feasibility with the samples, boundaries for the resource amount and cost values are introduced. With the proposed methods, the size of problems is reduced while the methods keep relatively better feasibility and quality of the solutions.

The rest of the paper is organized as follows. In Section II, we give background for the study. Then, in Section III, we propose the method to reduce the size of problems. The proposed methods are experimentally evaluated in Section IV. Discussions and related works are shown in Section V. We conclude our study in Section VI.

II. Background

A. Problem Definition

In this study, we use a definition of a resource allocation problem similar to [5], [6]. Most of the definitions are inherited from these studies.

In this network, the resource amounts of several nodes are allocated to other nodes. The previous studies assume that the network structure is limited to trees [5], [6]. Since feeder trees are common in actual power networks, such an assumption is reasonable.

The components of the network are as follows:

- Nodes: Each node in the network supplies or consumes an amount of resource. There are limitations on the

amounts of supply and consumption. Each node also has a preference on the resource amount.
- Links: Each link connects two nodes. The links and nodes form paths to transfer an amount of resource. The capacity of the link limits the amount of resource that is transferred in a link. As a common assumption, the resource loss during the transfer is ignored.

In addition to the above limitations on the resource amount, there is another type of constraints on the transferred resources. In each node, the total amount of the resource that is supplied and consumed has to be zero. The goal of the problem is to globally optimize an aggregation of preference under these constraints.

The problem is formally 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 set of resource amounts on the nodes, a set of cost functions, and a set of resource amounts on the links, respectively.

For node $i \in N$, the preference of the supply and the consumption of the resource amounts are defined as:

- R_i: $R_i \in R$ is a finite set of resource amounts that are supplied or consumed by node i. When amount $r \in R_i$ has a negative value, r represents an amount of the supplied resource. On the other hand, amount r represents an amount of consumed resource when it has a positive value. 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. Here, we use cost functions to represent preferences because our solution methods are designed for minimizing problems.

Link (i, j) is defined for a pair $\langle i, j \rangle$ of nodes. For each node i, the set of neighborhood nodes that are directly connected with links is denoted as Nbr_i. The transfer of the resource on link $(i, j) \in L$ is represented as:

- $l_{i,j}$: $l_{i,j}$ is the amount of resource transferred through link (i, j). $l_{i,j}$ must take a value such that $-l^c_{i,j} \leq l_{i,j} \leq l^c_{i,j}$, where $l^c_{i,j}$ is the capacity of link (i, j). The sign of value $l_{i,j}$ represents the direction of the transfer. The direction of a flow on the network must be defined. $l_{i,j}$ takes a positive value when the corresponding link transfers an amount of resource in the same direction as the flow.

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

$$\sum_{(i,j) \in L^{in}_i} l_{i,j} = r_i + \sum_{(i,k) \in L^{out}_i} l_{i,k} \tag{1}$$

with set L^{in}_i of input links and set L^{out}_i of output links of node i. L^{in}_i and L^{out}_i represent a flow on the network.

For allocation $\mathcal{R}$ of the resource amounts for all nodes, the global cost $f(\mathcal{R})$ is defined as $f(\mathcal{R}) = \sum_{i \in N} f_i(r_i)$. Here, r_i takes a corresponding value in allocation $\mathcal{R}$. The goal of the problem is to find the optimal allocation $\mathcal{R}^*$ that minimizes $f(\mathcal{R})$ under the constraints.

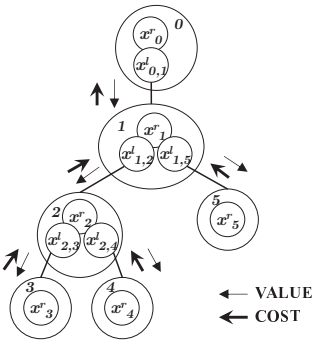


Fig. 1. Pseudo tree and message propagation for RCDCOP (acyclic network)

B. Representation as Resource Constrained DCOP

DCOPs have been studied as frameworks 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. This problem is solved using distributed cooperative algorithms that are based on message communication.

RCDCOP [7] is an extended class of DCOPs that contains dedicated representations of resources and constraints related to the resources. The resource allocation problem shown in Subsection II-A is formalized as RCDCOPs [5], [6]. Here, we show an RCDCOP that represents the resource allocation problem.

The RCDCOP for the resource allocation on the network is defined by $\langle A, L, X^r, D^r, F, C \rangle$. Here, A represents a set of agents. Each agent $i \in A$ 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. L is a set of links. X^r is a set of variables that represent the amounts of supplied or consumed resources in the nodes. D^r is a set of finite domains of variables in X^r. F is a set of cost functions and C is a set of constraints.

Additionally, X^l and D^l are introduced for links. X^l is a set of variables that represent the amounts of transferred resources in the links. D^l is a set of finite domains of variables in X^l. The domain of each variable for a link is dynamically computed as shown in Subsection II-C.

For a set of agents, a partial order is defined based on the pseudo tree [3] rooted at a node. Figure 1 shows a pseudo tree for an RCDCOP. In the figure, six agents/nodes are represented as nodes of the pseudo tree. The pseudo tree corresponds to a spanning tree of the original network. In this work, the pseudo tree is the same as the original tree, since original network is a feeder tree, Based on the pseudo tree, notations of parent agent p_i and a set of child agents Ch_i are defined for each agent i.

Agent i has a variable $x^r_i \in X^r$ that represents the amount of the supplied or consumed resource. i also has a set $X^l_i \subset X^l$ of variables that represent the amount of transferred resources from i. $x^l_{i,j} \in X^l_i$ represents the amount of resources transferred from agent i to its child agents $j \in Ch^l_i$. Similarly, $x^l_{p_i,i} \in X^l_{p_i}$ represents the amount of resources transferred

from i's parent agent p_i to i. Agent i decides the values of the variables except for $x^l_{p_i,i}$, whose value is determined by p_i.

$D^r_i \in D^r$ and $D^l_{i,j} \in D^l$ represent the domain of variable x^r_i for agent i and the domain of variable $x^l_{i,j}$ for link (i,j), respectively. While D^r_i corresponds to R_i, $D^l_{i,j}$ is dynamically computed as mentioned above.

$f_i(x^r_i) \in F$ is a cost function that corresponds to $f_i(r_i)$ for node i in the resource allocation problem. Similarly, $c^r_i \in C$ is a resource constraint for node i. Resource constraint c^r_i is defined as:

$$c^r_i : \ x^l_{p_i,i} = x^r_i + \sum_{j \in Ch_i} x^l_{i,j}. \tag{2}$$

Additionally, in the root node, $x^l_{p_i,i}$ must be zero.

Moreover, capacity constraint $c^l_{i,j} \in C$ is defined for each link (i,j) as:

$$c^l_i : \ -l^c_{i,j} \leq x^l_{i,j} \leq l^c_{i,j}. \tag{3}$$

Global cost function $f(\mathcal{X})$ for assignment $\mathcal{X}$ for all variables in $X^r \cup X^l$ is defined as $f(\mathcal{X}) = \sum_{i \in A} f_i(x^r_i)$. Here, x^r_i takes a corresponding value in $\mathcal{X}$. The optimal allocation $\mathcal{X}^*$ minimizes $f(\mathcal{X})$ under the constraints.

C. Computation based on Pseudo Trees

Several exact solution methods can be applied to the RCD-COP defined in Subsection II-B. In [5], [6], solution methods were shown for when the networks are trees. Here, we show important computations of the pseudo tree. See [2], [3], [5], [6], [7] for more details.

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

$$g^*_i(x^l_{p_i,i}) = \min_{\mathcal{X}_i} g_i(\{x^l_{p_i,i}\} \cup \mathcal{X}_i) \tag{4}$$

$$g_i(\{x^l_{p_i,i}\} \cup \mathcal{X}_i) = \delta_i(\{x^l_{p_i,i}\} \cup \mathcal{X}_i) + \sum_{j \in Ch_i} g^*_j(x^l_{i,j}) \tag{5}$$

$$\delta_i(\{x^l_{p_i,i}\} \cup \mathcal{X}_i) = \begin{cases} f_i(d_i) \text{ s.t. } (x^r_i, d_i) \in \mathcal{X}_i & c^r_i \wedge c^l_{p_i,i} \\ \infty & \text{otherwise.} \end{cases} \tag{6}$$

Here, $\mathcal{X}_i$ denotes an assignment such that $\{(x^r_i, d_i)\} \cup \bigcup_{j \in Ch_i} \{(x^l_{i,j}, d_{i,j})\}, d_i \in D^r_i, d_{i,j} \in D^l_{i,j}$. The value of $x^l_{i,j}$ corresponds to the value in $\mathcal{X}_i$.

In addition, domain $D^l_{p_i,i}$ of the variable for link (p_i, i) is computed based on the scope of $g^*_i(x^l_{p_i,i})$ as follows:

$$D^l_{p_i,i} = \{d^l_{p_i,i} | g^*_i(d^l_{p_i,i}) \neq \infty\}. \tag{7}$$

The above expression means that $g^*_i(x^l_{p_i,i})$ is eliminated if $g^*_i(x^l_{p_i,i}) = \infty$.

In Equations (4), (5), and (6), for the sake of simplicity, we assume that each agent is able to refer to the cost values and assignments of other agents. In actual computation, we employ an algorithm based on DYDOP [5] which is a variation of dynamic programming for DCOPs [3].

The processing consists of two phases: the bottom-up computation of the cost values and the top-down computation of the decision of the optimal assignments of the variables. These computations are performed using COST and VALUE messages as shown in Figrue 1[1]. In the bottom-up phase of the computation, for each possible assignment of $x^l_{p_i,i}$, cost values are aggregated in a bottom-up manner using COST messages. In the computation, assignments with the infinity cost values are eliminated as shown above.

The computed cost values are stored in a table that basically consists of elements $x^l_{p_i,i}$ and $g^*_i(x^l_{p_i,i})$. Additionally, x^r_i and $x^l_{i,j}$ for each $j \in Ch_i$ that are aggregated to $x^l_{p_i,i}$ are also stored in the table. Here, a row of the table is called a sample. Since a sample corresponds to an assignment, we interchangeably use them. For sample s, an element * of s is denoted by $s.*$. We define the size of a table as the number of samples in the table. The COST message from agent i to p_i sends a table that contains $s.x^l_{p_i,i}$ and $s.g^*_i(x^l_{p_i,i})$ for each sample s (i.e. $x^l_{p_i,i}$). The received COST message is stored as a table consisting of $x^l_{i,j}$ and $g^*_j(x^l_{i,j})$ for each child j.

After the bottom-up computation of cost values, the root agent has its table of global cost values that corresponds to $g^*_i(0)$ if there is the optimal assignment. Then, the root agent determines the optimal assignments of its variables. In the top-down phase of the computation, similarly, optimal assignments of other variables are recursively determined in a top-down manner using VALUE messages that propagate the optimal assignment of $x^l_{p_i,i}$. When each agent i determines the optimal assignments for i and i's subtrees from the optimal assignment $x^{l*}_{p_i,i}$, i looks up s^* such that $s^*.x^l_{p_i,i} = x^{l*}_{p_i,i}$. Then $s^*.x^r_i$ and $s^*.x^l_{i,j}$ for each $j \in Ch_i$ are assigned. Those elements have been stored in i's table by the bottom-up computation.

However, where the size of the variables' domains is large, these methods are not applicable due to the large size of the agents' local problems. In addition, the size of the local problem exponentially grows with the degree of the node. This is critical, even if the networks are trees, as it increases the number of combinations. Therefore, sampling of solutions is necessary to restrict the problem size. On the other hand, this also reduces the feasibility of the original problem.

III. Reducing samples

A. Basic Ideas

In the proposed method, we limit the size of tables with given parameters. To reduce the number of samples in the table, most of the samples are integrated to other samples with a heuristic.

However, when each node arbitrarily chooses a set of samples for its table, the total amount of resources may not equal zero. To overcome this problem, we introduce the boundaries of resource amounts for each samples. The

[1]While we use another simple representation, bottom-up and top-down computations correspond to Value Calculation and Value Propagation phases shown in [5]. Similarly, COST and VALUE messages correspond to *PowerCost* and *FlowCO* messages. In addition, we consider each $PowerCost$ message as a table since its elements $FlowCO_j$s represent pairs of a resource amount and a cost value. Each $FlowCO_j$ corresponds to a sample of resource amounts.

boundaries represent the amount of resources that satisfies constraints on resources. When the total amount of resources is not equal to zero, several samples are modified to satisfy the constraint by inserting errors based on the boundaries.

We also introduce the cost values for the boundaries to estimate cost values of modified samples. The cost values are evaluated to determine an allocation of errors on the amount of resources. The boundaries and cost values are computed with the original samples.

B. Partial Integration of Samples

To reduce the number of samples, the maximum limit number M of the samples (i.e. the maximum number of rows in the tables of cost values) is applied to the tables. When a new sample s is computed and added to the table, there are following cases.

(a) There is a sample t such that $t.x^l_{p_i,i} = s.x^l_{p_i,i}$. In this case, both samples are aggregated by minimizing cost values. Therefore, the number of samples does not increase. Otherwise, the following cases are applied.

(b) There is a sample t whose value of $t.x^l_{p_i,i}$ is within a *range* of $s.x^l_{p_i,i}$. If there are multiple samples in the range, the sample with the highest cost value is selected. In this case, both samples are aggregated by minimizing cost values. Therefore, the number of samples does not increase. Otherwise, the following cases are applied.

(c) The number of samples in the table is less than the maximum limit number M. In this case, the new sample s is added to the table. Otherwise, case (d) is applied.

(d) There is a sample t whose value of $t.x^l_{p_i,i}$ is the nearest value of $s.x^l_{p_i,i}$. In this case, both samples are aggregated by minimizing cost values. Therefore, the number of samples does not increase.

In the case of (b), for a new sample s, samples t whose values of $t.x^l_{p_i,i}$ are within a *range* of $s.x^l_{p_i,i}$ are compared. To evaluate the range, the distance between sample t and s is defined as follows.

$$dis(s,t) = |s.x^l_{p_i,i} - t.x^l_{p_i,i}|. \tag{8}$$

For each agent i, the range of samples is defined with threshold parameter dis_i. If $dis(s,t) \leq dis_i$, then t is in the range of s. We define dis_i based on the range of $x^l_{p_i,i}$ as follows. First, the minimum value $x^{lmin}_{p_i,i}$ and the maximum value $x^{lmax}_{p_i,i}$ of $x^l_{p_i,i}$ are computed aggregating the minimum and maximum values in D^r_i and $D^l_{i,j}$ for each child agent j. Then dis_i is calculated as follows.

$$dis_i = (x^{lmax}_{p_i,i} - x^{lmin}_{p_i,i})/M. \tag{9}$$

For new sample s, $s.g^*(x^l_{p_i,i})$ and $t.g^*(x^l_{p_i,i})$ of sample t in the table are compared. Then one of the samples with the highest cost value is eliminated.

In addition, $dis(s,t)$ is also employed to evaluate the distance of two samples in case (d). Figure 2 shows the procedure of the above method. In the procedure, $\prec$ denotes the preference of samples. While $\prec$ is simply defined as $<$ here, the operator is modified to evaluate the feasibility of samples prior to the evaluation of cost values as shown in Subsection III-E3.

```
1  S ← ∅ // set of samples (i.e. table)
2  for all X ∈ D^r_i ⊗ ⨂_{j∈Ch_i} D^l_{i,j} {
3   generate sample s s.t.
4    s.x^l_{p_i,i} = s.x^r_i + Σ_{j∈Ch_i} s.x^l_{i,j},
5    s.g*_i(x^l_{p_i,i}) = g({s.x^l_{p_i,i}} ∪ X),
6    {(x^r_i, s.x^r_i)} ∪ {(x^l_{i,j}, s.x^l_{i,j}) | j ∈ Ch_i} = X.
7   if t s.t. t.x^l_{p_i,i} = s.x^l_{p_i,i} exists in S { // case (a)
8    if s.g*_i(x^l_{p_i,i}) ≺ t.g*_i(x^l_{p_i,i}) { replace t by s. } }
9   else if t s.t. dis(s,t) ≤ dis_i exists in S { // case (b)
10   choose t s.t. ∀t' ∈ S \ {t}, t' ⪯ t.
11   if s.g*_i(x^l_{p_i,i}) ≺ t.g*_i(x^l_{p_i,i}) { replace t by s. } }
12  else{
13   if |S| < M { add s to S. } // case (c)
14   else{ // case (d)
15     select t s.t. argmin_{t∈S} dis(s,t).
16     if s.g*_i(x^l_{p_i,i}) ≺ t.g*_i(x^l_{p_i,i}) { replace t by s. } } } }
```

Fig. 2. Integration of samples

This method is based on heuristics to maintain better cost values while considering the distribution of samples. On the other hand, the feasibility and quality of the solutions decrease when the number of samples is significantly reduced.

C. Boundaries for Feasibility

For resource allocation problems, the feasibility of solutions is a serious issue. When the number of samples is reduced, the feasibility decreases. Therefore, an additional method is required to solve the problem when no feasible solutions are left in the samples.

While there are two types of constraints, infeasibility is mainly caused by resource constraints shown as Equation (2). As a result of eliminating samples, a resource constraint may not be satisfied in the root node agent. In such case, $x^l_{p_i,i}$ of the root agent is not equal to zero. That is, there may be no assignments of resources whose sum equals zero.

When an agent has a continuous amount of resources, the resource amounts around each sample are available to absorb extra amounts in the resource constraint. Filling such extra amounts into continuous values requires boundaries of possible continuous values. The original computation is therefore extended to compute the boundaries of the continuous resources.

For each sample s in the table of agent i, the lower bound $s.x^{\perp}_{p_i,i}$ and upper bound $s.x^{\top}_{p_i,i}$ of $s.x_{p_i,i}$ are introduced. If there are only discrete amounts of resources in the subtree rooted at agent i, $s.x^{\perp}_{p_i,i} = s.x_{p_i,i} = s.x^{\top}_{p_i,i}$. That is, there is no flexibility to allow absorption of extra amounts of resources. On the other hand, if the amount of the resources is continuous, the boundaries are determined based on the possible range of these amounts.

The range of resource amount R_i is defined by $r^{\perp}_i$ and $r^{\top}_i$, which are the minimum and maximum values in R_i. With these ranges, boundaries $r^{\perp}$ and $r^{\top}$ of $r \in R_i$ are defined as $r^{\perp}_i \leq r^{\perp} \leq r$ and $r \leq r^{\top} \leq r^{\top}_i$. In addition, for each link (i,j), the range of resource amount $l_{i,j}$ is defined by $-l^c_{i,j} \leq l_{i,j} \leq l^c_{i,j}$, as shown in Subsection II-A. Therefore,

boundaries $l^{\perp}$ and $l^{\top}$ of amount l are defined based on a range similar to $r^{\perp}$ and $r^{\top}$. The boundaries of $s.x_{p_i,i}$ are computed from these elements.

As shown in Equation (2), $x_{p_i,i}$ is the sum of a set of values of x_i^r and $x_{i,j}^l$ for all child nodes. Additionally the value of $x_{p_i,i}$ is also restricted by Equation (3). Calculations of $x_{p_i,i}^{l\perp}$ and $x_{p_i,i}^{l\top}$ are defined as:

$$x_{p_i,i}^{l\perp} = \max\left(-l_{p_i,i}^c, x_i^{r\perp} + \sum_{j\in Ch_i} x_{i,j}^{l\perp}\right) \tag{10}$$

$$x_{p_i,i}^{l\top} = \min\left(l_{p_i,i}^c, x_i^{r\top} + \sum_{j\in Ch_i} x_{i,j}^{l\top}\right). \tag{11}$$

Here, $x_i^{r\perp}$ and $x_i^{r\top}$ take values $r^{\perp}$ and $r^{\top}$ such that $x_i^{r\perp} = r^{\perp}$ and $x_i^{r\top} = r^{\top}$. Note that there are cases such that $x_{p_i,i}^{l\top} < -l_{p_i,i}^c$ or $l_{p_i,i}^c < x_{p_i,i}^{l\perp}$. In such cases, we eliminate the sample. In other cases, $x_{p_i,i}^{l\perp} \le x_{p_i,i}^{l\top}$ is always true.

$x_{p_i,i}^l$ is also modified to satisfy $-l_{p_i,i}^c \le x_{p_i,i}^l \le l_{p_i,i}^c$ and $x_{p_i,i}^{l\perp} \le x_{p_i,i}^l \le x_{p_i,i}^{l\top}$ if necessary. However, the modification of $x_{p_i,i}^l$ causes an error on the amount of resources. This error is treated as a offset in the computation of error values as shown in Subsection III-E2 (b).

As shown above, boundaries $r^{\perp}$ and $r^{\top}$ of $r \in R_i$ can take values from their range. Therefore, there are several strategies for choosing the boundaries. Here we employ the *widest* strategy: $r_i^{\perp} = r^{\perp}$ and $r^{\top} = r_i^{\top}$. With this boundary, each sample covers as wide a range as possible. As a result, the feasibility of the solutions increases. On the other hand, the ranges of a number of samples will overlap.

The computation of the bottom-up phase is extended with the computations shown above. In addition to the cost values computation, the boundaries are calculated. When the root agent computes its table, there may be no samples in the table due to violation of its resource constraint. In such cases, the root agent chooses an assignment by modifying the sample. When $s.x_{p_i,i}^l$ is not zero, s satisfies the resource constraint by adding error $e_{p_i,i}^l = 0 - s.x_{p_i,i}^l$ to $s.x_{p_i,i}^l$. To this end, $e_{p_i,i}^l$ must be absorbed by i and i's child agents.

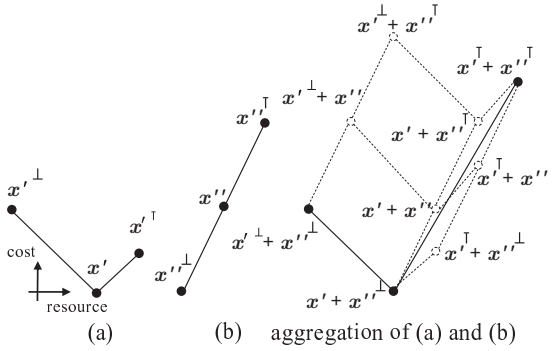
When there are multiple samples that satisfy the resource constraint with errors, we have several strategies for choosing a sample. One method chooses the sample with the minimum error.

In the top-down computation phase, root agent i sends assignment $s.x_{i,j}^l$ to each child agent j based on the selected sample s. In this communication, error $e_{i,j}^l$ for sample $x_{i,j}^l$ is also propagated. That is, the root agent shares its error with its child agents. If there are multiple child agents, the error is distributed among the agents. The share of the error values is shown as follows:

$$e_{p_i,i}^l = e_i^r + \sum_{j\in Ch_i} e_{i,j}^l \tag{12}$$

$$\text{where } s.x_i^{r\perp} \le s.x_i^r + e_i^r \le s.x_i^{r\top} \tag{13}$$

$$s.x_{i,j}^{l\perp} \le s.x_{i,j}^l + e_{i,j}^l \le s.x_{i,j}^{l\top}. \tag{14}$$



(a) (b) aggregation of (a) and (b)

Fig. 3. Aggregation of cost values for boundaries

Here, e_i^r denotes the error for the assignment of i. There are also several strategies for sharing the error. One method allocates the error to all nodes in proportion to the range of the bounds.

Each non-root agent i receives assignment $x_{p_i,i}^l$ and error $e_{p_i,i}^l$ from its parent agent p_i. Agent i simply chooses the corresponding sample s in its table based on $x_{p_i,i}^l$. i then allocates the share of error $e_{p_i,i}^l$ among i and the child nodes in Ch_i. In the computation of the shares of error values, a strategy similar to the case of the root node is employed. i then sends assignment $s.x_{i,j}^l$ and error $e_{i,j}^l$ to each child agent j.

The above method increases the feasibility by modifying samples. While this modification is reasonable, the cost value should be also evaluated.

D. Estimation of Cost Values Considering Errors

The boundaries of errors on amounts of resources are addressed above. Here, we focus on the estimation of cost values in such cases.

Estimating the cost values requires several assumptions. We assume knowledge of resources of continuous amounts. With this knowledge, an interpolation method of cost values is applied to the boundaries of the amounts of resources $x^{\perp}$ and $x^{\top}$ for each x. Let $h(r)$ denote the cost value for $r \in R_i$. Here we employ $h(r^{\perp}) = f_i(r^{\perp})$ and $h(r^{\top}) = f_i(r^{\top})$. Additionally, cost values $h(x_{i,j}^{l\perp})$ and $h(x_{i,j}^{l\top})$ are also applied to $x_{i,j}^{l\perp}$, and $x_{i,j}^{l\top}$.

The aggregation of those cost values is defined below. Besides the addition of resource amounts, the cost values are calculated as:

$$h(x^{\perp}) = h(x'^{\perp}) + h(x''^{\perp}) \text{ where } x^{\perp} = x'^{\perp} + x''^{\perp}. \tag{15}$$

For $h(x^{\top})$, the cost value is similarly calculated.

On the other hand, the aggregation for value x that is not bound values $x^{\perp}$ and $x^{\top}$ is defined as follows.

$$h(x) = h(x'^{*}) + h(x''^{*}) \text{ where } x = x'^{*} + x''^{*}. \tag{16}$$

$$x'^{*} = \text{argmin}_{x\in\{x'^{\perp}, x', x'^{\top}\}} h(x) \tag{17}$$

$$x''^{*} = \text{argmin}_{x\in\{x''^{\perp}, x'', x''^{\top}\}} h(x) \tag{18}$$

The above calculation is considered as shown in Figure 3. In this example, two boundaries (a) and (b) are aggregated. In the

figure, the horizontal axis stands for resource amounts while the vertical axis stands for cost values. Black dots represent cost values for $x^{\perp}$, x and $x^{\top}$. The aggregations for the boundaries $x'^{\perp} + x''^{\perp}$ and $x'^{\top} + x''^{\top}$ based on Equation (15) are reasonable.

On the other hand, as shown in the figure, the true aggregation may generate boundaries of cost values in different directions. Here we employ the value with the minimum cost value ($x' + x''^{\perp}$). As a result of this approximation, the number of vertices does not change.

Note that the computation of $h(x)$ shown in Equation (16) may cause an error on the amount of resources since x may not be equal to $x' + x''$. This error is treated as a offset in the computation of error values as shown in Subsection III-E2 (c).

When the boundaries of resources are limited by the capacity of links as shown in (10) and (11), cost values $h(x^l_{p_i,i})$, $h(x^{l\perp}_{p_i,i})$ and $h(x^{l\top}_{p_i,i})$ should be appropriately managed. We simply modify the cost values in proportion to limited and original bounds. As shown in Figure 3, there are two line segments $(x^{l\perp}_{p_i,i}, x^l_{p_i,i})$ and $(x^l_{p_i,i}, x^{l\top}_{p_i,i})$ for each sample. Therefore, $h(x^l_{p_i,i})$, $h(x^{l\perp}_{p_i,i})$ and $h(x^{l\top}_{p_i,i})$ are moved on these line segments so that $x^l_{p_i,i}$, $x^{l\perp}_{p_i,i}$ and $x^{l\top}_{p_i,i}$ satisfy the capacity of links.

In the aggregation of minimizing cost shown in Subsection III-C, a new sample s is treated based on the cases (a), (b), (c) and (d).

When the assignment is determined with resource amount errors, cost values are also evaluated. With the estimated cost values, agents determine the share of errors on resource amounts. The cost value for an amount of resource x is estimated based on $h(x^{\perp})$, $h(x)$ and $h(x^{\top})$. In addition, to estimate cost values from a limited number of samples, interpolation methods are necessary. As basic strategies, we employ a linear interpolation.

The estimation of cost values is based on the sign of error $e^l_{p_i,i}$ in agent i. If $e^l_{p_i,i} < 0$ then the error is shared decreasing the margin of lower bounds of resource amounts of agents. For x^r_i, the gradient of the cost value is calculated as follows.

$$-(h(x^{r\perp}_i) - h(x^r_i))/(x^{r\perp}_i - x^r_i)) \tag{19}$$

Similarly, the gradient of cost values for $x^l_{i,j}$ is as follows.

$$-(h(x^{l\perp}_{i,j}) - h(x^l_{i,j}))/(x^{l\perp}_{i,j} - x^l_{i,j})) \tag{20}$$

Otherwise, the margins of upper bounds are similarly decreased. The gradients of cost values for x^r_i and $x^l_{i,j}$ are represented as follows.

$$(h(x^{r\top}_i) - h(x^r_i))/(x^{r\top}_i - x^r_i)) \tag{21}$$

$$(h(x^{l\top}_{i,j}) - h(x^l_{i,j}))/(x^{l\top}_{i,j} - x^l_{i,j})) \tag{22}$$

Considering the gradients, the error is allocated to the agents whose gradient values are small. Here, we greedily allocate errors. The margins of resource amounts in agents are filled with errors in ascending order on gradient values.

E. Several Details

1) Tables and messages: The proposed method needs additional properties in the table of samples. In the original table, each sample has $x^l_{p_i,i}$, $g^*_i(x^l_{p_i,i})$, x^r_i and $x^l_{i,j}$ for each $j \in Ch_i$. On the other hand, in the proposed method, each sample also has $x^{l\perp}_{p_i,i}$, $x^{l\top}_{p_i,i}$, $g^*_i(x^{l\perp}_{p_i,i})$, $g^*_i(x^{l\top}_{p_i,i})$, $e^l_{p_i,i}$ and $e^l_{i,j}$ for each $j \in Ch_i$. COST messages are extended to contain $x^{l\perp}_{p_i,i}$, $x^{l\top}_{p_i,i}$, $g^*_i(x^{l\perp}_{p_i,i})$, $g^*_i(x^{l\top}_{p_i,i})$ and $e^l_{p_i,i}$ in addition to $x^l_{p_i,i}$ and $g^*_i(x^l_{p_i,i})$. VALUE messages are extended to contain $e^{l*}_{p_i,i}$ in addition to the information of the optimal assignment $x^{l*}_{p_i,i}$. Here $e^l_{p_i,i}$ and $e^l_{i,j}$ are error values on amounts of resources that are computed in the bottom-up computation. $e^{l*}_{p_i,i}$ is the error value for the optimal assignment $x^{l*}_{p_i,i}$. As shown above, the proposed method employs additional properties. Therefore, there are a trade-off between the number of samples and the additional properties.

2) Errors on ammounts of resoruces: In the proposed method, three types of errors on amounts of resources are considered. (a) The root agent inserts an error to absorb extra or insufficient amounts of resources. Then the error is assigned to the root agent and its subtrees. Other nodes also determine assignments of the error. This type of errors is represented as $e^{l*}_{p_i,i}$ and sent using a VALUE message. (b) When $x^l_{p_i,i}$ of a sample is modified to satisfy the capacity of link (p_i, i), an error is inserted for $x^l_{p_i,i}$. This error is stored as $e^l_{p_i,i}$ and sent using a COST message. (c) When two samples are aggregated as shown in Figure 3, an error may be inserted. This error is stored to as $e^l_{i,j}$ for each $j \in Ch_i$. $e^l_{i,j}$ takes non-zero value when $x^{l\perp}_{i,j}$ or $x^{l\top}_{i,j}$ is aggregated for $x^l_{p_i,i}$ instead of $x^l_{i,j}$. Since error values (b) and (c) are fixed values, they are treated as offset values in the assignments of error (a).

3) Addtitional heuristics: In the proposed method, operator $\prec$ in Figure 2 is modified to evaluate the feasibility of samples prior to the evaluation of cost values. Here the feasibility means whether $x^l_{p_i,i}$ is feasible or not. Note that an assignment in $[x^{l\perp}_{p_i,i}, x^{l\top}_{p_i,i}]$ can be feasible even if $x^l_{p_i,i}$ is infeasible. However, we prefer the samples of feasible $x^l_{p_i,i}$ for the opportunities of globally feasible solutions. Also, when the root node determines the optimal assignment, small error values on amounts of resources are preferred prior to cost values. Since the accuracy of approximated cost values in the current method is relatively low, we prefer assignments of small error values. To address these issues more studies will be necessary.

F. Correctness and Complexity

The proposed method is clearly an incomplete algorithm since it eliminates solutions without considering exact optimality. On the other hand, the algorithm always terminates and outputs a feasible solution or the information of infeasibility.

Since the proposed method limits the size of the tables in each agent to M, the maximum size of messages is also M. The proposed method does not directly address the issue of table size that grows with the number of child agents. Therefore, each agent evaluates at most $|R_i| \cdot M^n$ samples for

TABLE I
FEASIBILITY AND SIZE OF TABLE

cost func.	alg.	capacity of links $[-\infty, \infty]$ feasibility [%]	$[-\infty, \infty]$ max. sz. of tbl.	[-500,500] feasibility [%]	[-500,500] max. sz. of tbl.
random	exact	100	6201	100	1001
	lmt100rnd	88	100	88	100
	lmt10	44	10	44	10
	lmt100	40	100	48	100
	lmt10-res	100	10	100	10
	lmt100-res	100	100	100	100
	lmt10-res-cost	100	10	100	10
	lmt100-res-cost	100	100	100	100
linear	exact	100	6201	100	1001
	lmt100rnd	40	100	40	100
	lmt10	32	10	32	10
	lmt100	52	100	52	100
	lmt10-res	100	10	100	10
	lmt100-res	100	100	100	100
	lmt10-res-cost	100	10	100	10
	lmt100-res-cost	100	100	100	100
quadratic	exact	100	6201	100	1001
	lmt100rnd	40	100	40	100
	lmt10	32	10	32	10
	lmt100	56	100	56	100
	lmt10-res	100	10	100	10
	lmt100-res	100	100	100	100
	lmt10-res-cost	100	10	100	10
	lmt100-res-cost	100	100	100	100

TABLE II
COST VALUE

cost func.	alg.	capacity of links $[-\infty, \infty]$ cost value min	$[-\infty, \infty]$ cost value max	$[-\infty, \infty]$ cost value ave	[-500,500] cost value min	[-500,500] cost value max	[-500,500] cost value ave
random	exact	197	366	286	218	336	280
	lmt10-res	413	27331	13305	314	29026	14198
	lmt100-res	413	27331	14275	252	29026	13421
	lmt10-res-cost	309	2881	1167	323	2820	1082
	lmt100-res-cost	309	2881	1159	273	2971	949
linear	exact	6	3367	1009	6	3367	1009
	lmt10-res	6	6455	2037	6	6401	2054
	lmt100-res	6	6501	1947	6	6449	1957
	lmt10-res-cost	6	4783	1463	6	4783	1391
	lmt100-res-cost	6	4783	1424	6	4865	1364
quadratic	exact	6	117390	20772	6	117390	20772
	lmt10-res	12	160304	33258	12	183161	40071
	lmt100-res	6	161127	40885	6	183689	46372
	lmt10-res-cost	6	625738	159495	6	657767	145978
	lmt100-res-cost	6	625738	137207	6	678851	125186

the number n of child nodes. However, the maximum table size M can be individually reduced for each child node.

IV. EVALUATION

The proposed method was experimentally evaluated. The experiments were performed using simulation programs. We used example problems that were motivated by the feeder trees on power supply networks. The problems were designed for the proposed methods while those resemble the problems shown in [5], [6]. As the first experiment, we chose relatively simple settings.

Since the networks are trees, the problem consists of n nodes and $n-1$ links. Each tree was generated as a binary tree as possible. For the sake of simplicity, amounts of resource R_i are the same for all agents. We designed R_i as a set of integer values in $[r_i^{\perp}, r_i^{\top}]$. Similarly, the capacities of links $l_{i,j}^c$ have the same value for all links. In the following, we will show the results in the case of $n = 50, [r_i^{\perp}, r_i^{\top}] = [-100, 100], l_{i,j}^c = \infty$ or 500. All problem instances have feasible solutions.

As cost function $f(r_i)$, the following three types of functions were used:

- Random: $f(r_i)$ takes random integer values from $[1, 1000]$ based on an uniform distribution.
- Linear: Each node i has its most preferred amount r_i^* of its resource. r_i^* is randomly set based on a uniform distribution. In addition, node i has a weight value w_i in $[1, 10]$. $f(r_i)$ is defined as $w_i \cdot |r_i^* - r_i|$.
- Quadratic: While it resembles 'linear,' $f(r_i)$ is defined as $w_i \cdot (r_i^* - r_i)^2$.

For each type of problem, 25 instances were averaged.

The following solution methods were evaluated:

- exact: The solution method shown in Subsection II-C. We consider this method as a baseline since it resembles the conventional method [5].
- lmt100rnd: This method integrates samples when the number of samples in the tables reaches the maximum limit value 100. However, instead of sample distances, random selection is always employed.
- lmt100/10: The method that partially integrates samples (Subsection III-B). The maximum number of samples in a table is 100 or 10.
- lmt100/10-res: The method that manages the boundaries of the amounts of resources is employed with 'lmt100/10' (Subsection III-C).
- lmt100/10-res-cost: The method that manages the estimation cost values is employed with 'lmt100/10-res' (Subsection III-D).

Except for 'exact', the maximum number of samples in the table is limited to $M = 100$ or 10.

Table I shows the results for different types of problems. Here, we evaluated the feasibility of solutions and the maximum size of tables. In the case that the link capacity is $[-\infty, \infty]$, 'lmt100/10-res' and its variations are effective to satisfy feasibility, since those methods employ errors on resource amounts to satisfy the resource constraints.

In most cases, those methods improved the feasibility of solutions in comparison with 'lmt100/10' even if the limit value M was small. While 'lmt100' was slightly better to solve feasible solutions than 'lmt100rnd' in the case of 'linear' and 'quadratic', both methods are insufficient for these problem instances. On the other hand, the effects of those method were inverted in the case of 'random'.

In the case of link capacity $[-500, 500]$, similar results were obtained. We also evaluated the case of link capacity $[-50, 50]$. In this case, all methods found the optimal solution for each problem. This result reveals the fact that the influence of the link capacity is not monotonic for the effects of these solution methods.

Table II shows the cost values of feasible solutions. 'lmt100/10-res-cost' is more effective than 'lmt100/10-res' in

the case of linear cost functions. In addition, similar effects are shown in the case of random cost functions.

However, in the case of quadratic functions, 'lmt100/10-res-cost' increased the cost values. It is considered that the simple linear approximation of 'lmt100/10-res-cost' do not match with such cost functions. While 'lmt100/10-res-cost' has effects in several cases, its accuracy is relatively low. To compute better estimation cost values, dedicated sampling and approximation methods will be necessary.

In these experiments, we limited the size of tables (i.e. the number of samples) $M = |D^l_{i,j}|$ to relatively small values. Each agent computes $|D^r_i| \cdot \prod_{j \in Ch_i} |D^l_{i,j}|$ samples to aggregate tables form child agents and its own cost function. Even in the case of $R = [-100, 100]$, $M = 100$ and a binary tree. Non-leaf agents compute $201 \cdot 100^2 = 2010000$ samples in the worst case.

The proposed methods need computational overheads. Therefore, there is a trade-off between the original method and the proposed methods. The trade-off cannot be exactly evaluated since our current implementation is not well optimized. In the case of link capacity $[-\infty, \infty]$ and linear cost functions, 'exact' needs almost 103 seconds while 'lmt100-res-cost' needs almost 11 seconds on Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz machine with 16GB memory. Reducing the size of messages will also take an advantage in practical systems.

V. Discussions and related works

There are several studies on DCOPs with continuous values. In [8], continuous piecewise linear functions (CPLFs) are employed to represent objective functions of continuous values. With CPLFs, domains of functions are represented by a set of convex polytopes. To aggregate objective functions, the summation of two CPLFs is defined. The CPLFs were applied to Max-Sum algorithm [4]. This method is considered as an approximation based on sampling. While a CPLF represent a whole objective function, extensions are necessary to handle resource constraints.

In [9], numeric methods are applied to Max-Sum algorithm. Each agent performs a newton method or a gradient method to compute locally optimal samples of its local functions. This method also needs additional extensions to solve the problems with resource constraints.

In this study, we mainly focused on resource constraints on resource supply networks. Boundaries on resource amounts are introduced to maintain feasibility with the limited number of samples. On the other hand, our current method employs relatively naive approximation of cost functions as the first study. To improve solution quality, our approach will be integrated with the related works shown above. In addition, as shown in Figure 3, there are issues in the aggregation of cost functions on resource amounts.

Similar to the conventional studies, we focus on a problem at a discrete time. For temporal sequences of problems with a continuous flow of resources, there are opportunities to apply several techniques for dynamic DCOPs [10], [11].

VI. Conclusion

In this study, we proposed methods to reduce the number of sample data in solution methods for the resource allocation problems of power supply networks. To maintain the samples, boundaries for the amount of resources and cost values were introduced. With the proposed methods, the size of problems is reduced while the methods keep relatively better feasibility and quality of the solutions.

Our future work will include more accurate schemes to represent the bounded samples and sophisticated interpolation methods. Iterative approaches that employ search methods to feedback and to refine the solution quality will also improve the proposed method.

Acknowledgment

This work was supported in part by KAKENHI Grant-in-Aid for Scientific Research (C), 25330257 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] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings, "Decentralised coordination of low-power embedded devices using the max-sum algorithm," in *7th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2008, pp. 639–646.

[5] S. Miller, S. D. Ramchurn, and A. Rogers, "Optimal decentralised dispatch of embedded generation in the smart grid," in *11th International Conference on Autonomous Agents and Multiagent Systems*, vol. 1, 2012, pp. 281–288.

[6] T. Matsui and H. Matsuo, "Considering equality on distributed constraint optimization problem for resource supply network," in *2012 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2012, pp. 25–32.

[7] 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.

[8] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings, "Decentralised coordination of continuously valued control parameters using the max-sum algorithm," in *8th International Conference on Autonomous Agents and Multiagent Systems*, vol. 1, 2009, pp. 601–608.

[9] T. Voice, R. Stranders, A. Rogers, and N. R. Jennings, "A hybrid continuous max-sum algorithm for decentralised coordination," in *19th European Conference on Artificial Intelligence*, 2010, pp. 61–66.

[10] A. Petcu and B. Faltings, "Optimal solution stability in dynamic, distributed constraint optimization," in *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2007, pp. 321–327.

[11] W. Yeoh, P. Varakantham, X. Sun, and S. Koenig, "Incremental dcop search algorithms for solving dynamic dcops," in *10th International Conference on Autonomous Agents and Multiagent Systems*, vol. 3, 2011, pp. 1069–1070.