ANALYSIS FOR DISTRIBUTED COOPERATION BASED ON LINEAR PROGRAMMING METHOD

Toshihiro Matsu and Hiroshi Matsuo
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, 466-8555, Aichi, Japan

Keywords: Multi-agent, Distributed cooperative problem solving, Linear programming, Optimization.

Abstract: Distributed cooperative systems have optimization problems in their tasks. Supporting the collaborations of users, or sharing communications/observations/energy resources, are formalized as optimization problems. Therefore, distributed optimization methods are important as the basis of distributed cooperation. In particular, to handle problems whose variables have continuous domains, solvers based on numerical calculation techniques are important. In a related work, a linear programming method, in which each agent locally performs the simplex method and exchanges the sets of bases, has been proposed. On the other hand, there is another interest in the cooperative algorithm based on a linear programming method whose steps of processing are more distributed among agents. In this work, we study the framework of distributed cooperation based on a distributed linear programming method.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Distributed cooperative systems have optimization problems in their tasks. Supporting the collaborations of users, or sharing communications/observations/energy resources, are formalized as optimization problems. To solve the problems in distributed cooperative processing, understanding the protocols of the distributed optimization algorithms is important. In the research area of Distributed Constraint Optimization Problems (Modi et al., 2005; Petcu and Faltings, 2005; Mailler and Lesser, 2004), cooperative problem solving is mainly studied for (non-linear) discrete optimization problems. On the other hand, to solve the problems whose variables have continuous domains, another type of solvers is also important. Other related works propose optimization algorithms based on numerical calculation techniques for distributed cooperative systems (Wei et al., 2010; Burger et al., 2011). In a related work (Burger et al., 2011), simplex algorithm for linear programming has been applied to multiagent systems. In the method, each agent locally performs the simplex method to solve its problem and exchanges the sets of bases. A good point of the method is the simple protocol. On the other hand, there is another interest in the cooperative algorithm based on a linear programming method whose processing is more distributed among agents. While there are a number of studies about parallel simplex methods (e.g. (Ho and Sundarraj, 1994; Yarmish and Van Slyke, 2009)), their goals are slightly different from the situation in multiagent cooperation. In this work, we study a basic framework of distributed cooperation based on a distributed linear programming method whose parts are distributed among agents. The essential distributed processing and extracting the parallelism are investigated.

2 PREPARATIONS

2.1 Linear Programming Problems

The linear programming problems are fundamental optimization problems that consist of n variables, m linear constraints, a linear objective function (Chvatal, 1983). For the sake of simplicity, we assume the following problems.

$$\max: \quad \mathbf{c}^T\mathbf{x} \tag{1}$$
$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq 0 \tag{2}$$

Here, n-dimensional vector $\mathbf{x}$ consists of decision and slack variables. Each constraint contains a slack variable. $m \times n$ matrix $\mathbf{A}$ and m-dimensional vector $\mathbf{b}$ respectively represent coefficients and constants of the constraints. n-dimensional transposed vector $\mathbf{c}^T$ represents coefficients of the objective function.

2.2 Simplex Method

The simplex method is a fundamental solution method of the linear programming problems(Chvatal, 1983)D In computation of the method, initial bases are selected. Then the bases are repeatedly improved until they reach the optimal solution. In the case of the problems shown in 2.1, slack variables and the objective value are simply selected as the initial bases. The objective value must always be a base.

The set of bases is represented by Boolean vector **h**. Each element h_j of **h** is true if variable x_j is a base. Otherwise, h_j is false. The objective value is omitted in **h** because it is always a base. Matrix **D** is employed as a table that represents bases, constraints and an objective function in each step of the solution method. In the initial state, each element $d_{i,j}$ of **D** takes the following value.

$$d_{i,j} = \begin{cases} -a_{i,j} & 1 \le i \le m \wedge 1 \le j \le n \\ b_i & 1 \le i \le m \wedge j = n+1 \\ c_j & i = m+1 \wedge 1 \le j \le n \\ 0 & i = m+1 \wedge j = n+1 \end{cases} \quad (3)$$

The column for the objective value is omitted similar to **h**. Note that $d_{i,k}$ for base x_k takes -1. In the other row i' such that $i' \in \{1 \cdots m\} \backslash \{i\}$, $d_{i',k}$ is 0. In the problems shown above, the initial **D** means that the slack variables are selected as the initial bases.

2.2.1 Selection of New Base

In the first step of an iteration, the solution method selects one non-base variable x_{j^B} that has a positive coefficient of the objective function. Then x_{j^B} becomes a new base in the following steps. If all coefficients of the objective function are not positive, the solution method stops. In that case, for each i such that $1 \le i \le m$, $d_{i,n+1}$ represents the value of the base variable of row i. Also, $d_{m+1,n+1}$ represents the optimal objective value. Here,j^B is shown as follows.

$$\begin{aligned} j^B &= \operatorname{argmax}_j d_{m+1,j} \\ &\quad \text{s.t. } 1 \le j \le n \wedge \neg h_j \wedge d_{m+1,j} > 0 \end{aligned} \quad (4)$$

2.2.2 Selection of New Non-base

Instead of the new base x_{j^B}, variable x_{j^N} of base variables is selected as a non-base variable. In the representation of **D**, selecting one row i^N decides the corresponding base variable. Here, row i^N that minimizes the maximum feasible value of x_{j^B} is selected.

$$\begin{aligned} i^N &= \operatorname{argmin}_i \frac{d_{i,n+1}}{-d_{i,j^B}} \\ &\quad \text{s.t. } 1 \le i \le m \wedge d_{i,j^B} \ne 0 \wedge \frac{d_{i,n+1}}{-d_{i,j^B}} > 0 \end{aligned} \quad (5)$$

j^N is uniquely determined satisfying the following condition.

$$h_{j^N} \wedge d_{i^N,j^N} = -1 \quad (6)$$

2.2.3 Exchange of Bases

Now the new base and non-base are exchanged. First, in **D**, row i^N that corresponds to the new base is updated. The new value of each element $d'_{i^N,j}$ is shown as follows.

$$d'_{i^N,j} = \frac{d_{i^N,j}}{-d_{i^N,j^B}} \quad (1 \le j \le n+1) \quad (7)$$

Then, for each row i excluding i^N, x_{j^B} is eliminated. Each new value $d'_{i,j}$ is shown as follows.

$$\begin{aligned} d'_{i,j} &= d_{i,j} + d_{i,j^B} d'_{i^N,j} \\ &\quad (i \in \{1, \cdots, m+1\} \backslash \{i^N\},\ 1 \le j \le n+1) \end{aligned} \quad (8)$$

Here, d'_{i,j^B} is 0 because $d'_{i^N,j^B} = -1$. That represents the elimination of x_{j^B}. Additionally, elements of **h** are updated as $h_{j^B} \leftarrow \mathrm{T}$, $h_{j^N} \leftarrow \mathrm{F}$. After the exchange of the bases, the processing is repeated from selecting the new base.

3 A DISTRIBUTED SOLVER

In this work, we study a framework of distributed cooperation based on the linear programming problem and simplex method. Basically, problem and solver are divided into agents. In the initial state, each agent knows partial information that is directly related to the agent. Each agent only updates the initial constraints and its own coefficient of the objective function in the solution method. Information that is exchanged between agents and extraction of parallelism are mainly investigated. For the simple protocol, we employ a mediator that manages information.

3.1 Division of Problem

To represent the state of the agent, variable x_j is related to agent j. For the sake of simplicity, we assign agents for slack variables. In the following context, x_j and j may not be distinguished. In particular, a mediator is represented as z. Based on the variables, initial **D** is divided into agents. Agent j knows constraints that are related to its variable in the initial state. A constraint is known by multiple agents. j also knows coefficients of the objective function for known constraints. On the other hand, mediator z does not know any elements of **D** in the initial state.

Each agent j has table $\mathbf{D}_j$ that contains holes. The notations of $\mathbf{D}_j$ are compatible with $\mathbf{D}$. While $\mathbf{D}_j$ has the same size as $\mathbf{D}$, its unknown elements are zero. $\mathbf{D}_z$ represents the table of mediator z. Agent j also has $\mathbf{h}_j$ that partially contains elements of $\mathbf{h}$. In the initial state, each agent knows whether the variables related to the known constraints are the base or not. For each known base or non-base, $\mathbf{h}_j$ is appropriately initialized. Other elements of $\mathbf{h}_j$ are initialized by default value, false. Additionally, it is assumed that agents share information about the address of agent z and the number of all agents.

3.1.1 Selecting New Base

To select new bases, coefficients of the objective function have to be compared for all non-base variables. In the first step, each agent j sends coefficient $d_{m+1,j}$ of x_j in the objective function to mediator agent z. Exceptionally, in the case where x_j is a base or $d_{m+1,j} < 0$, 0 is sent. Additionally, for each non-base variable $x_{j'}$ that is known by agent j, j computes maximum value $v^{\top}_{j'}$ of $x_{j'}$ in the case where $x_{j'}$ is selected as the new base. This computation is a part of Equation (5). Let $V^{\top}_j$ denote a set of $v^{\top}_{j'}$ that is computed by j. j sends $V^{\top}_j$ with $d_{m+1,j}$ to mediator z.

Mediator z receives coefficients of the objective function from all agents whose variable is a non-base. Also, z receives the maximum values of the variables from all agents. When all values are received, z selects the candidate of new base x_{j^B}. Then z sends a request to change x_{j^B} to a new base. The request and d_{m+1,j^B} are sent to the agent who reported the minimum value of the maximum value $v^{\top}_{j^B}$ of x_{j^B}.

3.1.2 Selecting New Non-base

Selecting the new non-base is performed by agent k that is requested by mediator agent z. As shown in the previous subsubsection, agent k receives the request to change x_{j^B} to a new base and d_{m+1,j^B}. Then, based on $\mathbf{D}_k$, agent k computes row i^N, which corresponds to the new non-base in the case where x_{j^B} is changed to the new base.

3.1.3 Exchanging Bases

The exchange of the bases starts from agent k shown in 3.1.2 and is performed on agents who have part of $\mathbf{D}$ to be updated. In the following, the processing in agent k and the related agents is shown.

First, k updates an element of $\mathbf{h}_k$ based on $\mathbf{D}_k$ and $\mathbf{h}_k$ for row i^N that corresponds to the new non-base. The Boolean value of h_j that satisfies the condition $h_j \wedge d_{i^N,j} = -1$ shown in Equation (6) is set to false. By the update, k identifies that x_j is a new non-base. Next, k updates $\mathbf{D}_k$. Row i^N is previously saved as row vector $\mathbf{d}^{-}_{i^N}$, and row i^N is updated as shown in Equation (7). Then using the updated row i^N and d_{m+1,j^B} that have been received from z, rows of other constraints and its own coefficient $d_{m+1,k}$ of the objective function are updated. In particular, if $k = j^B$, $d_{m+1,k}$ is 0. Other coefficients of the objective function are always 0, which represents the unknown value. Moreover, k updates the value of h_{j^B} in $\mathbf{h}_k$ from false to true. Now k identifies that x_{j^B} is a base.

The update of $\mathbf{D}_k$ has to be sent to agents k' whose $\mathbf{D}_{k'}$ is affected by the update. k sends its coefficient $d_{m+1,k}$ of the objective function to mediator agent z. At the same time, the maximum values $V^{\top}_k$ of the non-base variables that are known by k are sent. k also sends the request that x_{j^B} is changed to a new base. The request, d_{m+1,j^B}, which has been received from z, and row vector $\mathbf{d}^{-}_{i^N}$ are sent to agents whose constraints are affected by the variables contained in the constraint of i^Nth row. When agent k' receives the request of new base x_{j^B} from agent k, k' updates $\mathbf{D}_{k'}$ and $\mathbf{h}_{k'}$ based on d_{m+1,j^B} and $\mathbf{d}^{-}_{i^N}$, which are received in the same message. Then k' sends $d_{m+1,k'}$ to mediator z. The maximum values $V^{\top}_{k'}$ of the non-base variables that are known by k' are also sent.

```
1  initialize $\mathbf{D}_z$. $t_z \leftarrow 0$. add $t_z$ to set $T$.
2  until the processing is terminated do {
3    while $z$'s receive queue is not empty
4      $\wedge$ the loop is not broken do { receive a message.}
5    maintenance. }
6  receive (OV, $d_{m+1,k}$, $V^{\top}_k$, $X^{\uparrow}_k$, set of $X^{\downarrow}_{k,j}$, $x$, $p$)
7    from agent $k$ {
8    store/update $d_{m+1,k}$, $V^{\top}_k$, $X^{\uparrow}_k$ and set of $X^{\downarrow}_{k,j}$.
9    $t_x \leftarrow t_x + p$ for $t_x$ in $T$. }
10 maintenance {
11   if $t = 1$ for all $t$ in $T$ then {
12     empty $T$. select new bases $X^B$.
13     if $X^B$ is empty { terminate the processing. }
14     foreach $x_{j^B}$ in $X^B$ do {
15       select agent $k$ that has minimum $v^{\top}_{j^B}$ in $V^{\top}_k$.
16       $t_{j^B} \leftarrow 0$. add $t_{j^B}$ to set $T$.
17       send (BV, $j^B$, $d_{m+1,j^B}$, $J_{x_{j^B}}$) to $k$. } } }
```

Figure 1: Processing in mediator z.

3.2 Area of Influence in Computation

The solution method needs to specify the agents that are related to the exchange of bases. For that purpose, agent k sends two sets $X^{\uparrow}_k$ and $X^{\downarrow}_{k,j}$ to mediator z. $X^{\uparrow}_k$

```
initialize $\mathbf{D}_k$. initialize $\mathbf{h}_k$.
if $h_k \vee d_{m+1,k} < 0$ then { let $d = 0$.}
  else { let $d = d_{m+1,k}$. }
send (OV, $d$, $V_k^{\top}$, $X_k^{\uparrow}$, set of $X_{k,j}^{\downarrow}$, $z$, $1/n$) to agent $z$.
until the processing is terminated do {
  while $k$'s receive queue is not empty
    $\wedge$ the loop is not broken do { receive a message.} }
receive (BV, $j^B$, $d_{m+1,j^B}$, $J_{x_{j^B}}$) from agent $z$ {
  select row $i^N$
    corresponding to new non$-$base variable.
  foreach $j$ such that $1 \le j \le n$ do {
    if $h_j \wedge d_{i^N,j} = -1$ then { $h_j \leftarrow F$. } }
  save $i^N$th row of $\mathbf{D}_k$ as $\mathbf{d}^-_{i^N}$.
  update $i^N$th row of $\mathbf{D}_k$.
  foreach $i$ such that $1 \le i \le m, i \ne i^N$ do {
    update $i$th row of $\mathbf{D}_k$ using $i^N$th row of $\mathbf{D}_k$. }
  update $d_{m+1,k}$
    using $i^N$th row of $\mathbf{D}_k$ and $d_{m+1,j^B}$ of BV message.
  $h_{j^B} \leftarrow T$.
  if $h_k \vee d_{m+1,k} < 0$ then { let $d = 0$.}
    else { let $d = d_{m+1,k}$. }
  send (OV, $d$, $V_k^{\top}$, $X_k^{\uparrow}$, set of $X_{k,j}^{\downarrow}$, $j^B$, $1/|J_{x_{j^B}}|$)
    to agent $z$.
  foreach agent $k'$ in $J_{x_{j^B}} \setminus \{k\}$ do {
   send (NBV, $j^B$, $d_{m+1,j^B}$ of BV message, $\mathbf{d}^-_{i^N}$, $|J_{x_{j^B}}|$)
      to agent $k'$. } }
receive (NBV, $j^B$, $d_{m+1,j^B}$, $\mathbf{d}^-_{i^N}$, $|J_{x_{j^B}}|$) from agent $k'${
  foreach $j$ such that $1 \le j \le n$ do {
    if $h_j \wedge j$th element of $\mathbf{d}^-_{i^N} = -1$ then { $h_j \leftarrow F$. }
      }
  update $\mathbf{d}^-_{i^N}$ similar to $i^N$th row of $\mathbf{D}$.
  if $k$ has $i^N$th row then { update $i^N$th row of $\mathbf{D}_k$. }
  foreach $i$ such that $1 \le i \le m, i \ne i^N$ do {
    update $i$th row of $\mathbf{D}_k$ using $i^N$th row of $\mathbf{d}^-_{i^N}$. }
  update $d_{m+1,k}$ using $\mathbf{d}^-_{i^N}$ and $d_{m+1,j^B}$ of BV message.
  $h_{j^B} \leftarrow T$.
  if $h_k \vee d_{m+1,k} < 0$ then { let $d = 0$.}
    else { let $d = d_{m+1,k}$. }
  send (OV, $d$, $V_k^{\top}$, $X_k^{\uparrow}$, set of $X_{k,j}^{\downarrow}$, $j^B$, $1/|J_{x_{j^B}}|$)
    to agent $z$. }
```

Figure 2: Processing in non-mediator agent k.

is the set of the variables on which agent k depends. $X_k^{\uparrow}$ is computed by each agent k excluding mediator z. As shown as follows, $X_k^{\uparrow}$ is the set of variables that are related to constraints contained in $\mathbf{D}_k$.

$$X_k^{\uparrow} = \{x_j \mid (d_{i,j} \in \mathbf{D}_k \wedge 1 \le i \le m \wedge 1 \le j \le n \wedge d_{i,j} \ne 0) \wedge (h_j \in \mathbf{h}_k \wedge 1 \le j \le n \wedge h_j)\} \quad (9)$$

Here, the condition of h_j is necessary to inform k that base x_j is changed to a non-base.

$X_{k,j}^{\downarrow}$ is the set of variables, which is affected in the case where agent k changes non-base x_j to a new base. $X_{k,j}^{\downarrow}$ is computed if at least one non-base variable x_j relates to a constraint that is known by k and the maximum number $v_j^{\top}$ of x_j is bounded by a constraint. Otherwise, the set is empty. Here let i^N denote the row of $\mathbf{D}_k$ that corresponds to a new non-base in the case of new base x_j. As shown as follows, $X_{k,j}^{\downarrow}$ is the set of variables that are related to the constraint of i^Nth row.

$$X_{k,j}^{\downarrow} = \begin{cases} \{x_j \mid d_{i^N,j} \in \mathbf{D}_k, & \neg h_k \wedge \\ \quad 1 \le j \le n, d_{i^N,j} \ne 0\} & x_j \text{ is bounded} \\ \phi & \text{otherwise} \end{cases} \quad (10)$$

$X_{k,j}^{\downarrow}$ can be computed before deciding whether variable x_j is selected as the new base or not.

When mediator agent z selects new base x_{j^B}, the set $J_{x_{j^B}}$ of the agents that relates to the change of basis is shown as follows.

$$J_{x_{j^B}} = \{j \mid \exists x \in X_{k,j^B}^{\downarrow}, x \in X_j^{\uparrow}\} \quad (11)$$

Here k represents the agent in which the maximum value $v_{j^B}^{\top}$ of x_{j^B} is minimum.

3.3 Employing Parallelism

In earlier steps of the solution method for sparse problems, it is possible to update multiple bases that do not interfere with each other in parallel. To employ the parallelism, the selection of the new base in mediator z is extended. In the extended processing, non-base variables that have positive coefficients of the objective function are sorted in descending order. Then independent updates of bases are enumerated based on the ordering. The first non-base is always selected as a new base. The following non-bases are similarly selected if they do not interfere with other new bases. Set $J_{x_{j^B}}$ of agents that relates to the update of new base x_{j^B} is shown in Equation (11). Set X^B of the new bases is shown as follows.

$$\forall x_{j^B} \in X^B, \forall x_{j^{B'}} \in X^B, j^B \ne j^{B'}, J_{x_{j^B}} \cap J_{x_{j^{B'}}} = \phi \quad (12)$$

3.4 Pseudo Code

Pseudo codes of the solution method are shown in Figures 1 and 2. In this processing, three types of messages are employed as follows. OV message transports coefficient values of the objective function and related information from each agent to mediator z. BV message transports requests of changing new bases from mediator to related agents. NBV message propagates requests of changing new bases from

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Table 1: Results.

				(a) #cycles until termination						(b) #parallel upd.			(c) #terms in const.			(d) #agents related to upd.		
alg.				ser			par											
a	o	n	m	min.	max.	ave.	min.	max.	ave.	min.	max.	ave.	min.	max.	ave.	min.	max.	ave.
2	1	20	10	20	35	30	11	32	25	1	2	1.27	4	12	5.00	7	20	11.02
		40	20	47	65	57	23	41	30	1	5	2.07	4	22	5.26	7	40	12.08
		80	40	92	122	110	20	41	30	1	9	3.97	4	23	5.20	7	57	11.40
3	2	20	10	20	38	28	20	38	28	1	1	1	5	12	7.09	12	20	16.85
		40	20	41	62	52	32	59	42	1	2	1.26	5	22	7.75	12	40	21.95
		80	40	86	116	103	38	80	53	1	5	2.08	5	37	8.38	12	80	23.47
4	3	20	10	17	38	26	17	38	26	1	1	1	6	12	8.69	17	20	19.41
		40	20	32	74	53	29	71	54	1	2	1.05	6	22	11.15	17	40	31.09
		80	40	86	113	98	53	101	72	1	3	1.44	6	42	12.44	17	80	41.06

the agents that have received the requests to related agents.

Figure 1 shows the processing in mediator agent z. After the initialization, z waits for OV messages from other agents. z detects that all OV messages are received using set T of the counter for the termination detection. Then z selects set X^B of new bases that can be updated in parallel. The changing of new bases is requested by sending BV messages. When set X^B of new bases is empty, mediator z detects the termination of the solution methods. In the case where X^B is not empty and no agent can update the bases, mediator z also detects the situation that the problem is unbounded.

Figure 2 shows the processing in non-mediator agent k. After the initialization, k waits for BV messages from mediator z or NBV messages from other agents k'. When either message is received, k updates $\mathbf{D}_k$ and $\mathbf{h}_k$. Then consequent messages are sent. When the solution method is terminated, there are the cases where the constraint of each base variable x_k does not exist in agent k. In that case, to determine k's assignment, the agent that has the constraint has to notify k of the constraint in post processing.

4 EXPERIMENTS

4.1 Settings of Experiments

We evaluated the example problems whose constraints partially overlap with neighborhood variables on a ring network. The problem is considered as the situation where neighboring agents share a limited amount of resources. Parameters to generate the problem are as follows. n^D: the number of the decision variables. n^D excludes the number of additional slack variables that is the same as the number of the constraints. a: the number of decision variables for each constraint. o: the number of overlapping variables in a constraint that overlaps with the next constraint in the ring network. $[v^{O\perp}, v^{O\top})$: the range of the coefficients of the objective function. $[v^{C\perp}, v^{C\top})$: the range of the coefficients of the constraints. $[v^{R\perp}, v^{R\top})$: the range of the parameter for the constants of the constraints. For each constraint, the parameter represents the ratio of the constant to the summation of all coefficients.

Here, n^DCa and o are set so that the constraints are uniformly placed on the ring network. The values of coefficients and constants are randomly determined with uniform distribution. In the following, the problems are represented using the total number $n = n^D + m$ of variables, the number m of constraints, a and o. $[v^{O\perp}, v^{O\top})$ and $[v^{C\perp}, v^{C\top})$ are respectively $[1, 3)$. $[v^{R\perp}, v^{R\top})$ is $[0.5, 1)$. The results are totaled for 20 instances.

The following two methods are compared. ser: the baseline method that sequentially updates bases. par: the method that employ parallelism if possible. When there is no parallelism, both methods work similarly.

As the criteria of the execution time, we used the number of the cycles of exchanging messages. In a cycle, the following processing is performed. First, each agent processes all messages in its own receiving queue and puts messages in the sending queue if necessary. Then the simulator moves the messages from the sending queues to the destinations' receiving queues for all agents.

4.2 Results

The number of cycles until the termination is shown in Table 1(a). Generally, the method par that simultaneously updates multiple new bases terminates in a lower number of cycles. In very sparse problems like $(a, o, n, m) = (2, 1, 80, 40)$, par effectively reduces the number of cycles. On the other hand, in problems like $(n, m) = (20, 10)$ and $(a, o) = (4, 3)$ whose constraints are relatively overlapped, the effects are small. The number of parallel updates of new bases is shown in Table 1(b). The problems in which the

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

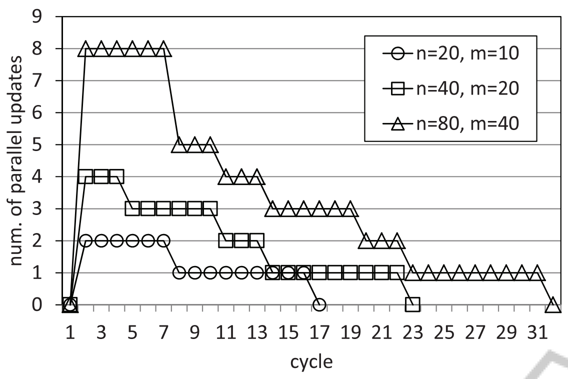

Figure 3: Transition of number of parallel updates of new basis (par, $a = 2, o = 1$).

number of cycles is effectively reduced as shown in Table 1(a) have a relatively large number in the parallelism. The number of terms in a constraint is shown in Table 1(c). The result represents that the size of the constraints increases with the progress of the solution method. Although the maximum number of the terms is less than the number of the variables, the parallelism is lost as shown above. Table 1(d) shows the number of agents related to an update of a new base. The maximum number that equals the number of variables represents that the locality of updates was lost in later cycles.

The transition of the number of parallel updates of the new basis for an example problem $a = 2, o = 1$ is shown in Figure 3. The number of parallelism is relatively large in the first steps and decreases in later cycles. There are two reasons of the decrement. One reason is that the possible new bases are eliminated by the solution method. Another is that the number of variables in the updated constraints increases.

5 CONCLUSIONS

In this work, we studied a framework of distributed cooperative problem solving based on the linear programming method. Essential processing for distributed cooperation and extracting the parallelism are shown. While there is possibility of parallel updates of the new bases in the sparse problems, the global tantalization of the information is necessary for the selection of new bases, and the extraction of the parallelism. Instead of the mediator, there are opportunities to decompose the tantalization using a tree structure of agents. Considering the fact that the locality of the problem is lost with the progress of the solution method, there is the possibility of an approach in which agents store the revealed information and employ it to reduce distributed processing. In (Burger et al., 2011), each step of the simplex method is not decomposed. Instead, each agent solves local problems and exchanges the sets of current bases. Although we focused on the sparse problems and the more distributed solver, the possibility of using the characteristics should be investigated to divide columns and to avoid synchronization. Decomposition of the processing of the mediator using a structured group of agents, applying efficient methods, and comparison/integration with related works will be included in future works.

ACKNOWLEDGEMENTS

This work was supported in part by a Grant-in-Aid for Young Scientists (B), 22700144.

REFERENCES

Burger, M., Notarstefano, G., Allgower, F., and Bullo, F. (2011). A distributed simplex algorithm and the multi-agent assignment problem. In *American Control Conference*, pages 2639–2644.

Chvatal, V. (1983). *Linear programming*. W.H.Freeman Company.

Ho, J. K. and Sundarraj, R. P. (1994). On the efficacy of distributed simplex algorithms for linear programming. *Computational Optimization and Applications*, 3:349–363.

Mailler, R. and Lesser, V. (2004). Solving distributed constraint optimization problems using cooperative mediation. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445.

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

Petcu, A. and Faltings, B. (2005). A scalable method for multiagent constraint optimization. In *9th International Joint Conference on Artificial Intelligence*, pages 266–271.

Wei, E., Ozdaglar, A., and Jadbabaie, A. (2010). A distributed newton method for network utility maximization. In *49th IEEE Conference on Decision and Control, CDC 2010*, pages 1816 –1821.

Yarmish, G. and Van Slyke, R. (2009). A distributed, scaleable simplex method. *The Journal of Supercomputing*, 49:373–381.