Effect of Bundle Method in Distributed Lagrangian Relaxation Protocol

Kenta Hanada
Kobe University and JSPS Research Fellow
5-1-1 Fukaeminami-machi, Higashinada-ku
Kobe 658-0022, Japan
kenta_hanada@stu.kobe-u.ac.jp

Katsutoshi Hirayama and **Tenda Okimoto**
Kobe University
5-1-1 Fukaeminami-machi, Higashinada-ku
Kobe 658-0022, Japan
hirayama@maritime.kobe-u.ac.jp
tenda@maritime.kobe-u.ac.jp

Abstract

The *Generalized Mutual Assignment Problem* (GMAP) is a maximization problem in distributed environments, where multiple agents select goods under resource constraints. *Distributed Lagrangian Relaxation Protocols* (DisLRP) are peer-to-peer communication protocols for solving GMAP instances. In DisLRPs, agents seek a good quality upper bound on the optimal value by solving the Lagrangian dual problem, which is a convex minimization problem. Existing DisLRPs exploit a *subgradient method* to explore a better upper bound by updating the *Lagrange multipliers (prices)* of goods. While the computational complexity of the subgradient method is very low, it cannot detect tha fact that an upper bound converges to the minimum. Moreover, solution oscillation sometimes occurs, which is critical for its performance. In this paper, we present a new DisLRP with a *Bundle Method* and refer to it as *Bundle DisLRP* (BDisLRP). The bundle method, which is also called the stabilized cutting planes method, has recently attracted much attention as a way to solve Lagrangian dual problems in centralized environments. We show that this method can also work in distributed environments. We experimentally compared BDisLRP with *Adaptive* DisLRP (ADisLRP), which is a previous protocol that exploits the subgradient method, to demonstrate that BDisLRP converged faster with better quality upper bounds than ADisLRP.

Introduction

Distributed optimization problems are one critical category for Multi-Agent Systems. In such problems, multiple agents cooperatively try to solve a problem instance and find an optimal assignment or a good quality solution for an entire system. There are many practical and theoretical approaches for such problems, including multi-robot coordination (Zheng and Koenig 2009), distributed facility location (Frank and Römer 2007), and distributed constraint optimization problem (Modi et al. 2003).

Hirayama et al. proposed *Generalized Mutual Assignment Problem* (GMAP) (Hirayama 2006) as a distributed optimization formalism. GMAP is an extended problem of the *Generalized Assignment Problem* (Savelsbergh 1997; Posta, Ferland, and Michelon 2012), which is a classic NP-hard problem in operations research field. The objective of GMAP is to find the globally optimal goods assignment while satisfying resource constraints of individual agents.

The first protocol for solving GMAP instances in purely distributed environment is DisLRP$_L$ (Hirayama 2006). DisLRP$_L$ is a stochastic algorithm that can efficiently compute a feasible solution with a lower bound on the optimal value. However, the bound is generally not so tight and there is no guarantee on its tightness. The second protocol is called DisLRP$_\alpha$ (Hirayama 2007), where we obtain a feasible solution that is guaranteed to have a quality of α (meaning that its objective value is not lower than $\alpha \times OPT$). The third one is Adaptive DisLRP (ADisLRP) (Hirayama, Matsui, and Yokoo 2009), which can yield a relatively tight upper bound on the optimal value to a GMAP instance.

On the other hand, a slightly different formalism of GMAP is proposed in (Hanada and Hirayama 2011). In this formalism, they relax the assignment constraint, which states every good is assigned to exactly one agent, by allowing goods to be assigned to at most one agent. They also provide two basic protocols for solving this version of GMAP. Recently, for task assignment in robotics domains, Luo et. al. proposed an auction-based protocol (Luo, Chakraborty, and Sycara 2013) that is very similar in both formalism and algorithm to the ones in (Hanada and Hirayama 2011).

Here is an outline of the agent behavior in DisLRPs. First, the agents solve their individual 0-1 knapsack problems and announce their assignments of goods to their respective neighbors indicating which goods they tentatively selected. Second, for all goods, they raise the *price* of their *Lagrange multiplier* if it is chosen by two or more agents and reduce their price if it is not chosen by any agent. Third, under the new prices, the agents solve their individual new 0-1 knapsack problems again. They repeat this procedure until all of the goods are chosen by exactly one agent, which means we get a proper set partition for the entire set of goods.

The protocols explore better (smaller) upper bounds on the optimal value by updating the Lagrange multipliers. This is called a *Lagrangian dual problem*. To update the Lagrange multipliers, existing DisLRPs exploit a well known and widely used subgradient method to solve Lagrangian dual problems. The advantage of the subgradient method is that the calculation speed for updating the Lagrange multipliers is very fast. However, it cannot detect the fact that the upper bound converges to the minimum. Moreover, solution oscillation sometimes occurs, which is critical for its performance.

In this paper, we propose a new DisLRP called *Bundle DisLRP* (BDisLRP) to address the above problems. A new feature of BDisLRP is that it exploits a *bundle method* (Bonnans et al. 2006) to simultaneously update the Lagrange multipliers with a synchronization protocol (Hirayama, Matsui, and Yokoo 2009). The bundle method, which is also called the stabilized cutting planes method, has recently attracted much attention as a way to solve Lagrangian dual problems in centralized environments. The bundle method's advantage is that it can provide the optimal value of the Lagrangian dual problem, but the subgradient method cannot. Since this feature brings another termination condition for BDisLRP, it decreases the number of iterations more than existing DisLRPs.

The contribution of this paper is that it shows the bundle method can work even in distributed environments and can also outperform the subgradient method in DisLRP. Even though the bundle method has been studied in centralized contexts, it hasn't been studied in distributed contexts, to the best of our knowledge. Second, we briefly discuss a convergence property of our protocol and the convergence proof for our protocol is the same of the concentrated context. Lastly, we analyze agent privacy for BDisLRP. We show that BDisLRP has some advantage and disadvantage in agent privacy over existing protocols.

This paper first formally defines GMAP followed by its properties. Then it presents the details of existing DisLRPs, which exploit the subgradient method within the protocols. Next, it provides the basic theory of the cutting planes method and the bundle method in a general context and explains our new protocol. Next, it gives experimental results on benchmark instances showing the protocol's actual performance and finally concludes this work.

Generalized Mutual Assignment Problem

The Generalized Mutual Assignment Problem (GMAP) is based on the Generalized Assignment Problem (GAP), which is a classic problem that has been studied for decades in operations research field.

Formulation of GAP

GAP is formulated as following Integer Programming (IP) problem:

$$
\begin{aligned}
\mathcal{GAP} \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J): \\
\max. \quad & \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj} \\
\text{s. t.} \quad & \sum_{k\in A} x_{kj} = 1,\ \ \forall j \in J, && (1) \\
& \sum_{j\in J} w_{kj}x_{kj} \le c_k,\ \ \forall k \in A, && (2) \\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J,
\end{aligned}
$$

where $A = \{1, ..., m\}$ is a set of agents, $J = \{1, ..., n\}$ is a set of goods, $p_{kj} \in \mathbb{R}^+$ is the profit, and $w_{kj} \in \mathbb{R}^+$ is the amount of resources required when agent k selects good j. $c_k \in \mathbb{R}$ is the capacity, i.e., the amount of available resources, of agent k. x_{kj} is a decision variable whose value is set to 1 when agent k selects good j and 0 otherwise. The goal of the problem is to maximize the summation of profits under assignment constraints (1), which means each good is assigned to exactly one agent and the knapsack constraints (2), which means that no agent can use resources that exceed her capacity.

Since it is a NP-hard problem, many exact algorithms (Posta, Ferland, and Michelon 2012; Savelsbergh 1997) and heuristic approaches (Diaz and Fernandez 2001; Yagiura, Ibaraki, and Glover 2006) have been proposed.

Formulation of GMAP

GMAP is the extension of GAP. GMAP is formulated as the following IP problem:

$$
\begin{aligned}
\mathcal{GMAP} \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J_k): \\
\max. \quad & \sum_{k\in A}\sum_{j\in J_k} p_{kj}x_{kj} \\
\text{s. t.} \quad & \sum_{k\in A_j} x_{kj} = 1,\ \ \forall j \in J, \\
& \sum_{j\in J_k} w_{kj}x_{kj} \le c_k,\ \ \forall k \in A, \\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J_k,
\end{aligned}
$$

where A_j is a set of agents who can select goods j and J_k is a set of goods that can be assigned to agent k. Obviously, a union of A_j for all goods j equals A, and a union of J_k for all agents k equals J. Without loss of generality, we can assume $A_j \neq \emptyset$ (i.e., $|A_j|$ does not equal zero).

Distributed Lagrangian Relaxation Protocol

Distributed Lagrangian Relaxation Protocols (DisLRPs) (Hirayama 2006; 2007; Hirayama, Matsui, and Yokoo 2009) are heuristic methods for solving GMAP instances by using only local communication among agents. One feature of DisLRPs is that they can provide an upper bound on the optimal value with no centralized control agent.

To solve GMAP instances in distributed contexts, we have to divide the problem while keeping its structure. *Dantzig-Wolfe decomposition* provides such problem decomposition. Lagrangian relaxation problem is obtained by dualizing the assignment constraints (1) of $\mathcal{GMAP}$ as follows:

$$
\begin{aligned}
\mathcal{LGMP} \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J_k): \\
L(\boldsymbol{\mu}) = \max. \quad & \sum_{k\in A}\sum_{j\in J_k} p_{kj}x_{kj} + \sum_{j\in J}\mu_j\left(1 - \sum_{k\in A_j} x_{kj}\right) \\
\text{s. t.} \quad & \sum_{j\in J_k} w_{kj}x_{kj} \le c_k,\ \ \forall k \in A, \\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J_k,
\end{aligned}
$$

where $\mu_j \in \mathbb{R}$ is called a *Lagrange multiplier* for goods j and vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ is called a *Lagrange multiplier vector*.

Since the objective is additive over the agents and the constraints are separable over the agents, this maximization can be achieved by each agent k solving the following subproblem:

$$\begin{aligned} &\mathcal{LGMP}_k(\pi_k(\boldsymbol{\mu})) && (\text{decide } x_{kj},\ \forall j \in J_k): \\ &L_k(\pi_k(\boldsymbol{\mu})) = \max. && \sum_{j \in J_k} (p_{kj} - \mu_j)\, x_{kj} + \sum_{j \in J_k} \frac{\mu_j}{|A_j|} \\ &\text{s. t.} && \sum_{j \in J_k} w_{kj} x_{kj} \leq c_k, \\ & && x_{kj} \in \{0,1\},\ \forall j \in J_k. \end{aligned}$$

where $\pi_k(\boldsymbol{\mu})$ is a projection of $\boldsymbol{\mu}$ over the jobs in J_k. Each agent k is responsible for $\mathcal{LGMP}_k(\boldsymbol{\mu})$.

To solve GMAP, without gathering all of the information in one place, a distributed solution is possible by exploiting the following properties of the relation between the decomposed subproblems and the global problem (Hirayama 2006):

Proposition 1. *For any value of $\boldsymbol{\mu}$, the total sum of the optimal values of $\mathcal{LGMP}_k(\pi_k(\boldsymbol{\mu}))|k \in A$ provides an upper bound on the optimal value of $\mathcal{GMAP}$.*

Proposition 2. *For some value of $\boldsymbol{\mu}$, if all of the optimal solutions to $\mathcal{LGMP}_k(\pi_k(\boldsymbol{\mu}))|k \in A$ satisfy the assignment constraints (1) of $\mathcal{GMAP}$, then these optimal solutions constitute an optimal solution to $\mathcal{GMAP}$.*

The upper bound should be as close as possible to the optimal value: the smaller the upper bound, the better. Thus we have another problem called the *Lagrangian dual problem*:

$$L(\boldsymbol{\mu}^*) = \min_{\boldsymbol{\mu}}.\ L(\boldsymbol{\mu}) = \min_{\boldsymbol{\mu}}.\ \sum_{k \in A} L_k(\pi_k(\boldsymbol{\mu})),$$

where $\boldsymbol{\mu}^*$ is its optimal solution. By the duality theory, the Lagrangian dual problem has the following property:

Proposition 3. *An objective function of the Lagrangian dual problem is always convex.*

Procedure

The following is the basic DisLRP procedure:

(Step 1) All agents set counter t to 1 and initialize all elements of their Lagrange multiplier vector $\pi_k(\boldsymbol{\mu}^{(t)})$ to 0.

(Step 2) Under a current value of $\boldsymbol{\mu}^{(t)}$, each agent k solves her knapsack problem to compute $L'_k(\boldsymbol{\mu}^{(t)})$. Then the agents send these results to their respective neighbors, who are a group of agents who share interests in the same good. Formally, the neighbors of agent k are a union of agents (except for k) with decision variables that appear in the assignment constraint on good j in R_k. Namely, the neighbors of agent k is denoted by $\bigcup_{j \in J_k} A_j \setminus k$.

(Step 3) If all assignment constraints of the original problem are satisfied, the agents can quit the procedure to provide an optimal solution.

(Step 4) Each agent k updates the Lagrange multiplier vector from $\boldsymbol{\mu}^{(t)}$ to $\boldsymbol{\mu}^{(t+1)}$ and t to $t+1$ and returns to Step 2.

After initializing their counter and price vectors, the agents repeat Steps 2 through 4 until all agents learn that they have reached an optimal solution of $\mathcal{GMAP}$. Counter t represents the number of times the agents perform Steps 2 through 4. We view this one series of executions over Steps 2 through 4 as a unit and call it a round indicated by counter t.

Note that the entire system's global information is sometimes required, for example, computing $L(\boldsymbol{\mu}^{(t)})$. In this work, we use a *spanning tree protocol* to collect this global information. Due to space limitations, see (Hirayama, Matsui, and Yokoo 2009) for details.

Subgradient Method

At Step 4 in the procedure, the existing DisLRPs exploit a well known and widely used subgradient method to solve Lagrangian dual problems.

The relaxed constraints correspond to the subgradient in the Lagrangian relaxation problem. Thus we can compute subgradient g_j for each good j in A_j as follows:

$$g_j = 1 - \sum_{k \in A_j} x_{kj}.$$

Since the subgradient is decided when $\boldsymbol{\mu}$ is input, we define vector $\boldsymbol{g}(\boldsymbol{\mu}) = (g_1, \ldots, g_n)$.

The existing DisLRPs use the following formula to update each element μ_j of $\boldsymbol{\mu}$ in the subgradient method:

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - l^{(t)} \cdot g_j^{(t)},$$

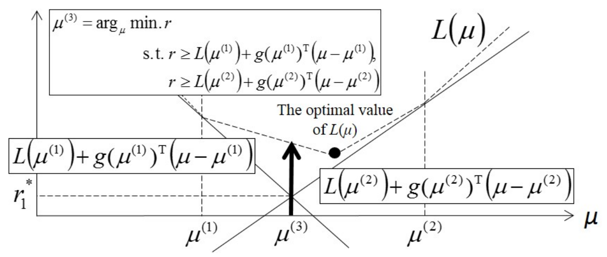
where l is a non-negative number called a *step length*. The first protocol, DisLRP$_L$ (Hirayama 2006) used a static number as the step length. In ADisLRP, Yokoo et al. (Hirayama, Matsui, and Yokoo 2009) introduced an adaptive price updating procedure for step lengths. The procedure adaptively decides the step length using the global information of the upper and (estimated) lower bounds. To update $\boldsymbol{\mu}$, the protocol uses the following step length:

$$l^{(t)} = \frac{\pi(\min\{ub\} - \max\{lb\})}{\|\boldsymbol{g}(\boldsymbol{\mu}^{(t)})\|^2}, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm, $\min\{ub\}$ is the lowest known upper bound, $\max\{lb\}$ is the largest known lower bound, π is a positive scalar parameter whose initial value is 2 that is halved when the least upper bound has not been updated during a specified interval of rounds.

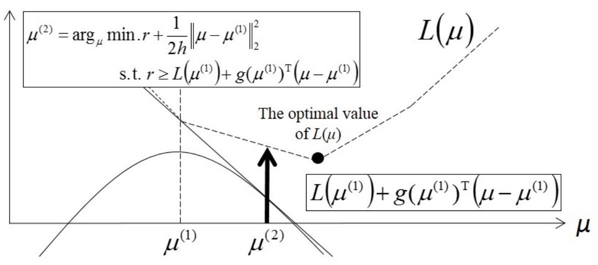
The advantage of the subgradient method is that its updates Lagrange multiplier vectors very quickly. However, the subgradient method that uses the (3) as a step length has no convergence proof[1]; it cannot detect that the upper bound converges to the minimum. Moreover, solution oscillation sometimes occurs, which is one critical issue for the subgradient method.

[1]Even though some convergence proofs depend on algorithms for step lengths, their calculation speed is generally very slow.



$\mu^{(3)} = \arg_\mu \min . r$
s.t. $r \ge L(\mu^{(1)}) + g(\mu^{(1)})^T(\mu - \mu^{(1)})$,
$r \ge L(\mu^{(2)}) + g(\mu^{(2)})^T(\mu - \mu^{(2)})$
$L(\mu)$
The optimal value of $L(\mu)$
$L(\mu^{(1)}) + g(\mu^{(1)})^T(\mu - \mu^{(1)})$
$L(\mu^{(2)}) + g(\mu^{(2)})^T(\mu - \mu^{(2)})$
r_1^*
$\mu^{(1)}$
$\mu^{(3)}$
$\mu^{(2)}$
μ

Figure 1: Example of cutting planes method at round 2



$\mu^{(2)} = \arg_\mu \min . r + \frac{1}{2h}\|\mu - \mu^{(1)}\|_2^2$
s.t. $r \ge L(\mu^{(1)}) + g(\mu^{(1)})^T(\mu - \mu^{(1)})$
$L(\mu)$
The optimal value of $L(\mu)$
$L(\mu^{(1)}) + g(\mu^{(1)})^T(\mu - \mu^{(1)})$
$\mu^{(1)}$
$\mu^{(2)}$
μ

Figure 2: Example of bundle method at round 1

Bundle DisLRP

In this section, we propose Bundle DisLRP (BDisLRP). First, we describe a theory of the cutting planes and bundle methods that will be used in the new protocol. Then we describe the algorithm and some of its features.

Cutting Planes Method

The cutting planes method is also well known as the subgradient method in operations research field. It constructs cutting planes for functions to narrow the possible range of optimal values.

In the subgradient method, the calculation result in round t is destroyed when the procedure goes to the next round: $t+1$. In the cutting planes method, agents record all the information of Lagrange multiplier vector $\boldsymbol{\mu}^{(t)}$, optimal value of Lagrangian relaxation problem $L(\boldsymbol{\mu}^{(t)})$, and subgradient $\boldsymbol{g}(\boldsymbol{\mu}^{(t)})$. We call this $\{\langle \mu^{(t)}, L(\mu^{(t)}), g(\mu^{(t)})\rangle | t \in \{1,\dots,T\}\}$ a *bundle* for all $t \in \{1,\dots,T\}$.

Here is the outline of the cutting planes method. First, based on the definition of subgradient, the following inequality is satisfied for all $\boldsymbol{\mu}$:

$$L(\boldsymbol{\mu}) \ge L(\boldsymbol{\mu}^{(t)}) + g(\boldsymbol{\mu}^{(t)})^{\mathsf{T}}(\boldsymbol{\mu} - \boldsymbol{\mu}^{(t)}),\ \forall \boldsymbol{\mu} \in \mathbb{R}^n. \quad (4)$$

We define the right hand side of inequality (4) as $f^{(t)}(\boldsymbol{\mu})$. Since $f^{(t)}(\boldsymbol{\mu})$ is a tangent hyperplane of the objective function of Lagrangian dual problem $L(\boldsymbol{\mu})$, we can prune the infeasible region of $L(\boldsymbol{\mu})$ by generating $f^{(t)}(\boldsymbol{\mu})$ in each round t. To update $\boldsymbol{\mu}$, we solve the cutting planes method by the following LP problem:

$$\begin{aligned} &\mathcal{CP} && (\text{decide } \boldsymbol{\mu}, r): \\ &\min. && r \\ &\text{s. t.} && f^{(t)}(\boldsymbol{\mu}) \le r,\ \forall t \in \{1,\dots,T\}. \end{aligned}$$

Let r_1^* be the optimal value of $\mathcal{CP}$. If $r_1^* = L(\boldsymbol{\mu}^{(t)})$ in round t, $\boldsymbol{\mu}^{(t)}$ is the optimal solution of the Lagrange dual problem. Figure 1 shows an example on one dimension at round 2.

We can obtain the Lagrangian dual problem's optimal value by iteratively solving the LP problem; however, the cutting planes method has some drawbacks. First, in the first few iterations, the procedure needs an artificial constraint to bound the permitted region. If the permitted region is unbounded, we cannot obtain the optimal value of $\mathcal{CP}$. It is difficult to add such an artificial constraint while keeping the model. Second, the procedure needs all the bundle information until we get the optimal value. This means that the number of constraints grows by the number of iterations. This is a memory consumption problem. Last, $L(\boldsymbol{\mu})$ is basically unstable and has slow convergence. Even though the cutting planes method has some good properties, it is impractical for solving Lagrangian dual problems.

Bundle Method (Stabilized Cutting Planes Method)

The bundle method is a proximal point method based on the cutting planes scheme. We can intuitively understand it by adding a stabilized (quadratic) term to $\mathcal{CP}$'s objective function.

In the bundle method, each agent solves the following quadratic programming (QP) problem to update $\boldsymbol{\mu}$:

$$\begin{aligned} &\mathcal{BM} && (\text{decide } \boldsymbol{\mu}, r): \\ &\min. && r + \frac{1}{2h}\|\boldsymbol{\mu} - \boldsymbol{\mu_{cp}}\|^2 \\ &\text{s. t.} && f^{(t)}(\boldsymbol{\mu}) \le r,\ \forall t \in \{1,\dots,T\}, \end{aligned}$$

where $\boldsymbol{\mu_{cp}} \in \mathbb{R}^n$ is a given vector called a center point and $h \in \mathbb{R}^+$ is a control parameter. Let r_2^* be the optimal value of $\mathcal{BM}$. Figure 2 shows an example on one dimension at round 1.

Note that $\mathcal{BM}$'s solution is unique but not $\mathcal{CP}$'s, since the feasible region is a non-empty closed convex set and $\mathcal{BM}$'s objective function is strictly convex.

The bundle method can avoid the drawbacks of the cutting planes method. First, since the quadratic term always bounds the problem, even in the first round of iterations, we don't have to add an artificial constraint. Second, the bundle method doesn't need to hold all of the bundle information until we get the optimal value. Due to the space limitations, see a previous work for further discussion and details (Bonnans et al. 2006). Most important, it is basically stable and has fast convergence.

BDisLRP procedure

We incorporate the bundle method into BDisLRP, whose basic procedure is the following:

(Step 1) All agents set counter t to 1 and initialize all the elements of their Lagrange multiplier vectors $\boldsymbol{\mu^{(t)}}$ and $\boldsymbol{\mu_{cp}}$ as 0. All agents also initialize parameters h, κ, and δ to certain values, where $\kappa \in (0,1)$ controls the center point and δ is a stopping tolerance.

(Step 2) Under a current value of $\boldsymbol{\mu}^{(t)}$, each agent k solves her knapsack problem to compute $L_k(\pi_k(\boldsymbol{\mu}^{(t)}))$. Then the agents send these results to their respective neighbors.

(Step 3) If all assignment constraints of the original problem are satisfied, the agents can quit the procedure to provide an optimal solution.

(Step 4) If all agents hold bundles from rounds 1 to $t'(\leq t)$, go to **Step 4.1**; otherwise go to **Step 5**.

(Step 4.1) All agents solve $\mathcal{BM}$ to compute r_2^* and $\boldsymbol{\mu}^{(t+1)}$.

(Step 4.2) Let δ' be $L(\boldsymbol{\mu_{cp}}) - r_2^*$. If $\delta' \leq \delta$ holds, the agents can quit the procedure to provide the optimal value of the Lagrangian dual problem.

(Step 4.3) If $L(\boldsymbol{\mu_{cp}}) - L(\boldsymbol{\mu}^{(t')}) \geq \kappa\delta'$ holds, the agents update the center point as follows: $\boldsymbol{\mu_{cp}} \leftarrow \boldsymbol{\mu}^{(t')}$.

(Step 4.4) Each agent updates t to $t+1$ and returns to **Step 2**.

(Step 5) Each agent k updates $\boldsymbol{\mu}^{(t)}$ to $\boldsymbol{\mu}^{(t+1)}$ by the subgradient method. Each agent also updates t to $t+1$ and returns to **Step 2**.

In Step 4.1, all agents solve exactly the same problem of $\mathcal{BM}$. Based on proposition 1, the solution of $\mathcal{BM}$ must be unique to provide the upper bound on the optimal value. As we mentioned before, since $\mathcal{BM}$'s solution is unique, agents can use any exact solver/method to solve it.

Step 4.3 is a key point of the bundle method. Iterations where $\boldsymbol{\mu_{cp}}$ is updated are called *serious steps* (SS), and iterations where $\boldsymbol{\mu_{cp}}$ is not updated are called *null steps* (NS). The SS test enhances the bundle method by sufficiently updating $\boldsymbol{\mu}$. Note that, in the distributed context, the agents simultaneously have to do the SS test by synchronizing the round that updates the center point. This updating process resembles the updating process of the step length in the subgradient method in ADisLRP. Therefore we introduce a synchronization protocol over a previously used spanning tree (Hirayama, Matsui, and Yokoo 2009). With this protocol, the agents simultaneously update the center point.

In the first few rounds, since the agents hold no bundles due to the communication delay, they cannot update $\boldsymbol{\mu}$ by the bundle method. Step 5 is required so that the agents can update $\boldsymbol{\mu}^{(t)}$.

Convergence Property

In this section, we discuss a convergence property for our proposed protocol. There is a convergence proof for a concentrated version of the bundle method (Bonnans et al. 2006), however, there is no convergence proof for a distributed one.

The only difference between the concentrated bundle method and our distributed one is the existence of communication delay. Agents in our protocol collect information to compute the global information such as the upper bound following the spanning tree protocol. The protocol allows agents to communicate with only their neighbors and they transfer information which is needed to compute the global information, therefore the communication delay occur.

We assume that there is no communication failure during execution of the procedure, thus agents can collect all bundles even supposing that there is the communication delay. Thus the convergence proof for our distributed protocol must be the same proof for the concentrated one.

Privacy Analysis

Agent privacy is a key motivation in distributed algorithms (Greenstadt, Pearce, and Tambe 2006). However, it is quite difficult to quantitatively evaluate agent privacy. Therefore we only compare BDisLRP with existing protocols what information are shared by only neighbors or all agents.

Table 1 shows the comparisons between protocols. *All agents* indicates that this information is shared by all agents. *Neighbors* indicates that only neighbors know the actual information. If agent 2 is a neighbor of agent 1 and agent 3 is not a neighbor of agent 1 e.g. a certain good is shared by agent 1 and 2, agent 1 knows information of agent 2 and agent 2 knows information of agent 1 as well. *Hard to estimate* indicates that it is hard to calculate or estimate actual information. In order to get global information, agents must diffuse their local information. For example, when agents receive the number related to the upper bound, agents add their own $L_k(\pi_k(\boldsymbol{\mu}))$ to the received number and forward to another agents. Therefore, agents are hard to know the actual number of neighbors' information except kickoff point of messages. - indicates that this information is not shared among the agents.

The optimal value of agents' knapsack problem $L_k(\pi_k(\boldsymbol{\mu}))$ is necessary to compute the upper bound $L(\boldsymbol{\mu})$. Only in BDisLRP, $L_k(\pi_k(\boldsymbol{\mu}))$ is shared by all agents due to the bundle. It is one of the drawbacks of agent privacy for BDisLRP.

The subgradient $g_j = 1 - \sum_{k \in A} x_{kj}$ can be computed by the solution of agents' knapsack problem. Thus the subgradient is primal information. In DisLRP$_L$, DisLRP$_\alpha$ and ADisLRP, the subgradient is shared by only neighbors. In BDisLRP, however, it is shared by all agents due to the bundle. As it is not good to share primal information in distributed algorithms, this is another drawback for BDisLRP.

$\|\boldsymbol{g}(\boldsymbol{\mu}^{(t)})\|^2$ is needed to calculate the step length in the adaptive subgradient method. DisLRP$_L$ and DisLRP$_\alpha$ don't need it because the step length is a static number. ADisLRP requires $\|\boldsymbol{g}(\boldsymbol{\mu}^{(t)})\|^2$ to compute the step length. In BDisLRP, all agents shares the subgradients and $\|\boldsymbol{g}(\boldsymbol{\mu}^{(t)})\|^2$ can be computed by the subgradients. As a result, agents can compute it.

Lagrange multipliers $\mu_j^{(t)}$ are dual information. It is basically difficult to estimate the primal information by using dual information. $\mu_j^{(t)}$ is shared by all agents in BDisLRP, we consider that it is not a serious drawback for BDisLRP.

ADisLRP must reveal the GMAP profit p_{kj} of selected goods to perform the on-line estimation of lower bounds. p_{kj} is primal information as well as the subgradient. DisLRP$_L$, DisLRP$_\alpha$ and BDisLRP have no on-line estimation of lower bounds, therefore agents need not to know p_{kj}. This is the advantage of BDisLRP over ADisLRP.

We are summarized as follows:

Table 1: Privacy Properties of DisLRPs

	DisLRP$_L$	DisLRP$_\alpha$	ADisLRP	BDisLRP
Optimal value of agents' knapsack problem $L_k(\pi_k(\boldsymbol{\mu}))$	-	-	Hard to estimate	All agents
Subgradient $g_j = 1 - \sum_{k \in A} x_{kj}$	Neighbors	Neighbors	Neighbors	All agents
$\|\boldsymbol{g}(\boldsymbol{\mu}^{(t)})\|^2$	-	-	Hard to estimate	All agents
Lagrange multipliers $\mu_j^{(t)}$	Neighbors	Neighbors	Neighbors	All agents
GMAP profit p_{kj} of selected goods	-	-	Neighbors	-

- DisLRP$_L$ and DisLRP$_\alpha$ are the best protocols in agent privacy.
- BDisLRP reveals more information than the others except for GMAP profit p_{kj} of selected goods.

Experiment

We experimentally compared the BDisLRP and ADisLRP performances.

We used GAP benchmark instances in category e from the OR-Library[2] to evaluate the performances. Since all the instances are a minimization problem, we translated each of them into an equivalent maximization problem by multiplying the costs by -1.

These experiments were conducted on an Intel i7-870@2.93 GHz with 4 Cores, 8 threads, and 8-GB memory. The main codes of BDisLRP and ADisLRP were written in Java and compiled with JDK 1.6.0-33 on Windows 7 Professional (64 bits). We used a commercial solver, ILOG CPLEX 12.5, as a local knapsack solver and a QP solver in each agent.

In all the experiments, we used identical parameter settings for BDisLRP: $h = 8, \kappa = 0.9$, and $\delta = 10^{-6}$.

The termination conditions of BDisLRP are different from ADisLRP. Since BDisLRP can provide the optimal value of the Lagrangian dual problem, we measured it when the procedure quits. ADisLRP cannot provide it, thus we used another termination condition. As with the centralized Lagrangian relaxation approach, $\pi^{(t)}$ takes 2 as its initial value and is halved when the least upper bound has not been updated during some specified interval of rounds. By doing this, $\pi^{(t)}$ is gradually reduced as the rounds proceed and eventually becomes almost close to zero. In our experiments with ADisLRP, we set this interval to 100 rounds. We measured the least upper bound when $\pi^{(t)}$ became smaller than 10^{-6}. Additionally, we set a 30-minute-time limit as the common termination condition of both BDisLRP and ADisLRP to speed up the experiments. We measured the least upper bound when the protocol is terminated by the time limitation.

Comparison with ADisLRP

First, we compared BDisLRP and the previous protocol, ADisLRP. We used the breadth-first search tree (BFS) as a spanning tree for both protocols. Table 2 shows our experiment results. Opt means the optimal value of the instances provided by Posta et al. (Posta, Ferland, and Michelon 2012). Time shows the execution time of our simulator in seconds. > 1800 in the Time column means that the execution time takes more than 30 minutes (1800 seconds). UB is the least upper bound after stopping the procedure. The optimal value of the Lagrangian dual problem doesn't need to necessarily correspond to the optimal value of the original problem. An optimality gap usually exists between them.

In almost all the cases, BDisLRP got better upper bounds than ADisLRP in the shorter rounds. For such small instances as e5100 (assigning 100 goods to five agents) and e5200 (assigning 200 goods to five agents), both BDisLRP and ADisLRP got the same upper bound. However, BDisLRP's round and execution times are much faster than ADisLRP. Since BDisLRP proves the optimality of the Lagrangian dual problem's solution, it can quit the procedure when the optimal value is found while ADisLRP wastes many rounds and time due to extra rounds. For large instances, such as e60900 (assigning 900 goods to 60 agents) and e801600 (assigning 1600 goods to 80 agents), BDisLRP failed to obtain the optimal value of the Lagrangian dual problem. Yet BDisLRP got a better least upper bound than ADisLRP in shorter rounds. As we mentioned before, since ADisLRP's updating cost is very low, the number of its iterations is higher than for BDisLRP. This might be caused by solution oscillation, or perhaps π is ineffective for obtaining a good quality upper bound. This implies that BDisLRP can update $\boldsymbol{\mu}$ more sufficiently with faster convergence than ADisLRP.

Impact of Communication Delay

In DisLRP, we have to consider *communication delay*, which is an additional round to learn each agent's global information. Since the agents only send their local information to their neighbors in the protocol, it takes a few rounds to propagate their information. This means that no agent can obtain the global information in the same round. The communication delay depends on the topology of a spanning tree. For example, in the same complete graph of five agents, the communication delay is three rounds for the breadth-first search tree and five rounds for the depth-first search tree.

Our second experiment evaluated the communication delay's impact on the BDisLRP performance. We used instances that converged to the optimal value of the Lagrangian dual problem in experiment 1. We ran our simulator with no time limit. In this experiment, we compared BDisLRP with both BFS and the depth-first search tree (DFS) on the benchmark instances. Table 3 shows our ex-

[2]http://people.brunel.ac.uk/ mastjjb/jeb/info.html

Table 2: BDisLRP vs. ADisLRP on benchmark instances in category e

Instance	Opt	Round		Time		UB	
		BDisLRP	ADisLRP	BDisLRP	ADisLRP	BDisLRP	ADisLRP
e5100	-12681	1180	4133	216	858	-12673	-12673
e5200	-24930	1021	3884	186	597	-24927	-24927
e10100	-11577	1037	3787	411	1094	-11568	-11568
e10200	-23307	1660	4246	268	1303	-23303	-23303
e10400	-45746	1258	4355	251	1387	$\mathbf{-45746}$	-45745
e15900	-102421	1510	2929	853	>1800	-102419	-102419
e20100	-8436	1417	3043	533	>1800	$\mathbf{-8432}$	-8431
e20200	-22379	1747	3524	>1800	>1800	$\mathbf{-22377}$	-22376
e20400	-44877	2674	2947	>1800	>1800	-44873	$\mathbf{-44875}$
e201600	-180645	2236	1655	>1800	>1800	$\mathbf{-180630}$	-179883
e30900	-100427	1576	1613	>1800	>1800	$\mathbf{-100418}$	-100333
e40400	-44561	799	1699	>1800	>1800	$\mathbf{-44537}$	-41538
e401600	-178293	856	1007	>1800	>1800	$\mathbf{-178248}$	-172704
e60900	-100149	355	1011	>1800	>1800	$\mathbf{-99621}$	-5340
e801600	-176820	148	2641	>1800	>1800	$\mathbf{-138987}$	-9454

Table 3: Impact of communication delay in BDisLRP

Instance	Opt	STree	Round	UB
e5100	-12681	BFS	1180	-12673
		DFS	1770	-12673
e5200	-24930	BFS	1021	-24927
		DFS	1702	-24927
e10100	-11577	BFS	1037	-11568
		DFS	3444	-11568
e10200	-23307	BFS	1660	-23303
		DFS	4400	-23303
e10400	-45746	BFS	1258	-45746
		DFS	4194	-45746
e15900	-102421	BFS	1510	-102419
		DFS	7567	-102419
e20100	-8436	BFS	1417	-8432
		DFS	9449	-8432

periment results. The BFS converges faster than DFS for all the instances. In addition, the more agents there are, the more time that is required in DFS. The number of agents impacts the communication delay in DFS. The SS test, which is an updating process of the center point, is crucial for the performance of the bundle method and requires global information. Thus communication delay affects the BDisLRP performance.

Conclusion

In this paper, we proposed a new Distributed Lagrangian Relaxation Protocol (DisLRP) called Bundle DisLRP (BDisLRP) and incorporated into DisLRP the bundle method, which has been studied to solve Lagrangian dual problems. Since the bundle method has a convergence proof, we can obtain the optimal value of the Lagrangian dual problem even in distributed contexts. We experimentally evaluated the BDisLRP and Adaptive DisLRP (ADisLRP) performances. BDisLRP converged faster with a better upper bound than ADisLRP, especially for large-scale instances. We also measured the impact of communication delay, which is caused by the topology of a spanning tree. Breath First Search (BFS) seems to have no relation with the number of agents for the performance, while Depth First Search (DFS)'s performance is greatly worsened with more agents.

In our experiments, we set static parameters for BDisLRP and proposed dynamic parameter controls. Future work will continue our experimental evaluations in various dynamic parameters. We also presented a basic version of the bundle method. Many enhanced bundle methods have been proposed, such as the *spectral* bundle method. We want to implement more powerful bundle methods to obtain tighter upper bounds, decrease the number of iterations, and speed up the calculation time.

We also need to improve the DFS performance. (Hirayama, Matsui, and Yokoo 2009) reported no impact of communication delay on ADisLRP. The hybrid technique, which combines the bundle and subgradient methods, may be a good solution for DFS's topology.

DisLRPs are heuristic methods for GMAP instances. There is no exact algorithm to solve GMAP instances in distributed environments. The upper bound is very useful information to explore the optimal value, and BDisLRP can obtain a tight upper bound. Another future work is to develop an exact algorithm with BDisLRP.

References

Bonnans, J. F.; Gilbert, J. C.; Lemaréchal, C.; and Sagastizábal, C. A. 2006. *Numerical Optimization: Theoretical and Practical Aspects*. Springer-Verlag New York, Inc.

Diaz, J. A., and Fernandez, E. 2001. A tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research* 132(1):22–38.

Frank, C., and Römer, K. 2007. Distributed facility location algorithms for flexible configuration of wireless sensor networks. In *Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS'07, 124–141.

Greenstadt, R.; Pearce, J. P.; and Tambe, M. 2006. An analysis of privacy loss in distributed constraint optimization. In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI-2006)*.

Hanada, K., and Hirayama, K. 2011. Distributed lagrangian relaxation protocol for the over-constrained generalized mutual assignment problem. In Kinny, D.; Hsu, J.-j.; Governatori, G.; and Ghose, A., eds., *Agents in Principle, Agents in Practice*, volume 7047 of *Lecture Notes in Computer Science*, 174–186. Springer Berlin Heidelberg.

Hirayama, K.; Matsui, T.; and Yokoo, M. 2009. Adaptive price update in distributed Lagrangian relaxation protocol. In *Proceedings of the 8th International Joint Conference on Autonomous Agents Multi-Agent Systems (AAMAS-2009)*, 1033–1040.

Hirayama, K. 2006. A new approach to distributed task assignment using Lagrangian decomposition and distributed constraint satisfaction. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006)*, 660–665.

Hirayama, K. 2007. An α-approximation protocol for the generalized mutual assignment problem. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-2007)*, 744–749.

Luo, L.; Chakraborty, N.; and Sycara, K. 2013. Distributed algorithm design for multi-robot generalized task assignment problem. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 4765–4771.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *Proceedings of the Second International Joint Conference on Autonomous Agents Multi-Agent Systems (AAMAS-2003)*, 161–168.

Posta, M.; Ferland, J. A.; and Michelon, P. 2012. An exact method with variable fixing for solving the generalized assignment problem. *Computational Optimization and Applications* 52(3):629–644.

Savelsbergh, M. 1997. A branch-and-price algorithm for the generalized assignment problem. *Operations Research* 831–841.

Yagiura, M.; Ibaraki, T.; and Glover, F. 2006. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research* 169(2):548–569. Feature Cluster on Scatter Search Methods for Optimization.

Zheng, X., and Koenig, S. 2009. K-swaps: Cooperative negotiation for solving task-allocation problems. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI-2009)*, IJCAI'09, 373–378.