An Asynchronous Heuristic Algorithm for Generalized Mutual Assignment Problem: Gossip-based Approach

Yuki Amemiya¹, Kenta Hanada¹, Kenji Sugimoto¹

¹Graduate School of Science and Technology, Nara Institute of Science and Technology

E-mail: {amemiya.yuki.bz3, hanada, kenji}@is.naist.jp

Abstract

An asynchronous gossip-based Lagrangian heuristic algorithm is proposed for Generalized Mutual Assignment Problem (GMAP) which is a combinatorial maximization problem in distributed environments. Lagrangian decomposition based formulation is introduced for GMAP in order to apply the asynchronous gossip algorithm. This enables us to obtain a feasible solution with quality bounds.

1 Introduction

Multi-agent based distributed optimization algorithms have been studied for several decades. In such algorithms, each agent in the system tries to maximize or minimize the global objective function, while the agents can communicate only with its local neighbors. In system control fields, consensus based algorithms are proposed for continuous convex optimizations [1, 2]. Such algorithms enable us to analyze theoretical results such as convergence properties and error bounds of the optimal value. On the other hand, in artificial intelligence fields, we deal with discrete (combinatorial) optimizations. Since most discrete optimizations are NP-hard and it is difficult to develop fast exact algorithms in order to obtain the optimal solution, many heuristic algorithms are proposed to search the good quality feasible solutions [3, 4, 5].

Generalized Mutual Assignment Problem (GMAP) [3, 4, 5] is a distributed combinatorial maximization derived from Generalized Assignment Problem (GAP) [6], which is one of the classical binary integer programming. Since these are general formulations of mathematical programming, there are a lot of applications for real world issues such as wireless communication networks [7] and multi-robot task assignment [8].

Existing protocols to solve GMAP are called Distributed Lagrangian Relaxation Protocol (DisLRP) [3, 4, 5]. The existing DisLRPs are all synchronous algorithms, that is, all agents update the information at the same time. Although they can obtain good quality upper bounds for the optimal value, the heuristics to obtain lower bounds (feasible solutions) is very simple and the performance is not good. Note that the protocol [4] can calculate the estimated lower bounds, this is however not proved the actual feasible solution.

In this paper, we propose an asynchronous heuristic algorithm for GMAP to obtain good feasible solutions. We reformulate GMAP by using the Lagrangian decomposition technique [9]. This enable us to generate the candidate feasible solutions of GMAP easily. Then, we apply the asynchronous gossip based consensus algorithm to obtain a variety of feasible solutions. We show experimental results to show the effectiveness of our algorithm.

2 Generalized Mutual Assignment Problem

In this paper, we deal with Generalized Mutual Assignment Problem (GMAP) [3, 4, 5] which is a combinatorial maximization in a distributed environment. An ordinary GMAP can be formulated as follows:

$$\max \sum_{i \in A} \sum_{j \in J} p_{ij} x_{ij}$$

s.t. $$\sum_{i \in A_j} x_{ij} = 1, \quad \forall j \in J$$

$$\sum_{j \in J_i} w_{ij} x_{ij} \leq c_i, \quad \forall i \in A,$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in A, \forall j \in J,$$

where $A = \{1, \ldots, m\}$ is a set of agents, $J = \{1, \ldots, n\}$ is a set of jobs, $A_j \subseteq A$ is the subset of $A$, $J_i \subseteq J$ is the subset of $J$, $p_{ij} \in \mathbb{R}$ is the profit, and $w_{ij} \in \mathbb{R}$ is the amount of resources consumption of agent $i$ when agent $i$ selects job $j$, $c_i \in \mathbb{R}$ is the capacity (the amount of available resources) of agent $i$. Note that only agents who belong to $A_j$ can select job $j$. Similarly, the subset $J_i$ means that only jobs which belong to $J_i$ are assigned to agent $i$.

The constraints (1), (2), and (3) are called the assignment, knapsack, and 0-1 constraint, respectively. From now on, we omit the 0-1 constraints due to the space limitation.

We define a undirected graph $G = (A, E)$, where $E \subseteq A \times A$ is the set of edges. The edge $(i, \ell) \in E$ represents availability of the communication between agent $i$ and $\ell$. In this paper, we assume that the graph $G$ is connected.

The aim of this paper is to develop the distributed heuristic protocol in order to obtain the good feasible
solution by applying the asynchronous gossip algorithm studied in the system control fields.

3 Lagrangian Heuristic Protocol

3.1 Lagrangian Decomposition

In this section, we consider a reformulation of GMAP by applying the Lagrangian decomposition technique [9]. Introducing copy variable, we can formulate GMAP as follows:

\[
\max_{\mathbf{p}} \sum_{i \in A} \sum_{j \in J_i} p_{ij} x_{ij} \quad \text{s.t. } x_{ij} = \alpha_{ij}, \quad \forall i \in A, \forall j \in J_i, \quad \sum_{i \in A} \alpha_{ij} = 1, \quad \forall j \in J, \quad \sum_{j \in J} w_{ij} x_{ij} \leq c_i, \quad \forall i \in A,
\]

where \(\alpha_{ij} \in \{0, 1\}\) is the copy variable for the decision variable \(x_{ij}\). The equation (4) is called the copy constraint. Note that this problem is clearly equivalent to the original GMAP formulation since the copy constraints (4) ensure that assignment constraints (1) are satisfied for all \(i \in A\).

We now consider a relaxation problem of the Lagrangian decomposition based formulation in order to consider an asynchronous Lagrangian heuristic algorithm. Relaxing the copy constraint in (4), we obtain the following Lagrangian relaxation problem:

\[
L(\mu) = \max \sum_{i \in A} \sum_{j \in J_i} p_{ij} x_{ij} + \sum_{i \in A} \sum_{j \in J_i} \mu_{ij} (\alpha_{ij} - x_{ij}) \quad \text{s.t. } \sum_{i \in A} \alpha_{ij} = 1, \quad \forall j \in J, \quad \sum_{j \in J} w_{ij} x_{ij} \leq c_i, \quad \forall i \in A,
\]

where \(\mu \in \mathbb{R}^{\sum_{i \in A} |J_i|}\) is the Lagrangian multiplier vector and its element \(\mu_{ij}\) is called the Lagrangian multiplier. Note that the function \(L(\mu)\) provides an upper bound on the optimal value of the original problem for any value of \(\mu\).

We can divide the above relaxation problem into subproblems since there is no cross term among the decision variables and each constraint consists of either \(x_{ij}\) or \(\alpha_{ij}\). The subproblem related to agent \(i\) can be described as

\[
L^n_i(\mu) = \max_{x_{ij}} \left( p_{ij} - \mu_{ij} \right) x_{ij} \quad \text{s.t. } \sum_{j \in J_i} w_{ij} x_{ij} \leq c_i.
\]

On the other hand, the subproblem related to job \(j\) can be described as

\[
L^g_j(\mu) = \max_{\alpha_{ij}} \mu_{ij} \alpha_{ij}
\]

subject to \(\sum_{i \in A_j} \alpha_{ij} = 1\).

The above subproblem can be rewritten as a non-constrained optimization

\[
L^n_j(\mu) = \max_{\mu_{ij}} \{ \mu_{ij} \}.
\]

We should point out that agents can solve all of the subproblems (6) if agents have the Lagrangian multiplier vector \(\mu\). Thus there is no need to treat the subproblem (6) for job \(j\) as an independent agent.

The upper bound should be as close as possible to the optimal value. Thus we have another optimization problem called Lagrangian dual problem as follows:

\[
L(\mu^*) = \min L(\mu) = \min \sum_{i \in A} L^n_i(\mu) + \sum_{j \in J} L^g_j(\mu), \quad \text{ where } \mu^* \text{ is the optimal solution of the Lagrangian dual problem.}
\]

Since this Lagrangian dual problem (7) is non-constrained convex optimization, we can apply existing consensus based algorithms.

3.2 Proposed Algorithm

In this section, we propose a Lagrangian heuristic protocol based on the Lagrangian decomposition formulation. In order to generate good quality feasible solutions by the Lagrangian heuristic algorithm, it is important to obtain the Lagrangian multiplier vector \(\mu\) which gives a good quality solution of the Lagrangian dual problem (7). Thus, the proposed algorithm consists of two parts.

One is an asynchronous gossip based algorithm which updates the Lagrangian multiplier vector \(\mu\) in order to obtain good quality solution of (7). The other is a heuristic algorithm to construct candidate feasible solutions based on the subproblem (6).

3.2.1 Asynchronous Gossip Based Algorithm

We introduce a gossip-based consensus algorithm to solve the Lagrangian dual problem (7). In this paper, we follow the asynchronous algorithm proposed by [2].

Let \(k\) be the \(k\)-th tick of the virtual clock in the algorithm. We define a local vector \(\mu^{(i)}[k]\) of agent \(i\) at time \(k\) as

\[
\mu^{(i)}[k] = [\mu_{11}^{(i)}[k] \mu_{12}^{(i)}[k] \cdots \mu_{mn}^{(i)}[k]]^T.
\]

In the Lagrangian relaxation problem, the relaxed constraint corresponds to a subgradient of (7). We therefore define the subgradient \(s^{(i)}_{ij}\) for each agent \(i\) and job \(j\) at \(\mu^{(i)}[k]\) as

\[
s^{(i)}_{ij}(\mu^{(i)}[k]) = \alpha^{*}_{ij}(\mu^{(i)}[k]) - x^{*}_{ij}(\mu^{(i)}[k]),
\]

where \(\alpha^{*}_{ij}(\mu^{(i)}[k])\) is the optimal solution of the subproblem (6) at \(\mu^{(i)}[k]\) and \(x^{*}_{ij}(\mu^{(i)}[k])\) is the optimal solution of the subproblem (5) of agent \(i\) at \(\mu^{(i)}[k]\).
Let \( r_i[k] \in \mathbb{R} \) be a local step size of agent \( i \) at time \( k \). Then, we introduce the following algorithm.

**Step 1** Each agent sets a random value to each element of \( \mu^{(i)}[1] \).

**Step 2** Wait until next tick \((k \leftarrow k + 1)\). Suppose that agent \( i \) is invoked at time \( k \). Agent \( i \) chooses its neighbor (agent \( \ell \)) randomly. Then, agent \( i \) (agent \( \ell \)) sends \( \mu^{(i)}[k] \{\mu^{(\ell)}[k]\}).

**Step 3** Calculate
\[
\bar{\mu}^{(i)}[k] = \bar{\mu}^{(i)}[k] - \frac{\mu^{(i)}[k] + \mu^{(\ell)}[k]}{2}.
\]
Agent \( i \) (agent \( \ell \)) solves its own subproblem (5) and subproblems (6) for all \( j \in J_i \) at \( \bar{\mu}^{(i)}[k] \) in order to calculate the subgradient \( s^{(i)}(\bar{\mu}^{(i)}[k])\).

**Step 4** Agent \( i \) (agent \( \ell \)) updates the Lagrangian multipliers as follows:
\[
\mu^{(i)}_{ij}[k + 1] = \mu^{(i)}_{ij}[k] - r_i[k]s^{(i)}(\mu^{(i)}[k]), \quad \forall j \in J_i,
\]
\[
\mu^{(\ell)}_{ij}[k + 1] = \bar{\mu}^{(\ell)}_{ij}[k], \quad \forall j \notin J_i.
\]
**Step 5** Return to Step 2.

The following lemma is derived from [2].

**Lemma 1** Let \( r_i[k] \) be the total number of agent \( i \) updates up to time \( k \). Then, the algorithm solves the Lagrangian dual problem (7) if \( k \to \infty \).

### 3.3 Heuristic Algorithm

In this section, we describe the second part of the proposed protocol.

As we mentioned before, agents can solve all of the subproblems (6). Since each agent has a copy of \( \mu \) as \( \mu^{(i)}[k] \), agent \( i \) can solve the optimal solution of the subproblem (6) independently as
\[
\alpha^*_{ij}(\mu^{(i)}_{ij}[k]) = \begin{cases} 1 & \text{if } i = \arg \max_{i \in A_{\ell}} \{\mu^{(i)}_{ij}[k]\}, \\ 0 & \text{otherwise}. \end{cases}
\]

This is just the candidate feasible solution of the original problem since this solution satisfies the assignment constraint (1) but this may violate the knapsack constraints (2). We therefore need additional procedure to check feasibility of (2) for candidates.

We now propose the heuristic protocol as follows. Let \( \alpha^c = [\alpha^c_{11} \alpha^c_{12} \cdots \alpha^c_{mn}]^T \) be a candidate of feasible solutions for the original problem. We introduce a binary variable \( v \in \{0, 1\} \) which indicates whether the candidate \( \alpha^c \) has been finished the investigation or not. We also introduce a value \( f \) to memory the total value of the objective function investigated so far. Let \( f_i \) be a partial objective function related to agent \( i \) at the candidate denoted as
\[
f_i(\alpha^c) = \sum_{j \in J_i} p_{ij} \alpha^c_{ij},
\]
Note that \( f \) will be \( \sum_{i \in A} f_i \) if \( \alpha^c \) is actually feasible solution after complete investigation. We define \( S_d \) and \( S_c \) as the subset of agents. Let \( T_i \) be a taboo list for agent \( i \). If \( \alpha^c \) is in the taboo list \( T_i \), it means that \( \alpha^c \) has already investigated completely.

When agent \( i \) found a new candidate \( \alpha^c \), proved feasible solution with \( f \), or proved infeasible solution, agent \( i \) executes the following procedure.

**Step 1** If agent \( i \) invokes this procedure as a new candidate \( \alpha^c \), initialize \( v = 0 \) and calculate \( f_i \).

**Step 2** If agent \( i \) invokes this procedure as a proved feasible solution, initialize \( v = 1 \) and update \( T_i \leftarrow T_i \cup \{\alpha^c\} \).

**Step 3** If agent \( i \) invokes this procedure as a proved infeasible solution, initialize \( v = 1 \) and update \( T_i \leftarrow T_i \cup \{\alpha^c\} \).

**Step 4** Set \( S_d \) as \( A \setminus \{i\} \), \( S_c \) as \( \{i\} \), and a bundle \( \{\alpha^c, v, S_d, S_c\} \).

**Step 5** Wait until next communication. Suppose that agent \( i \) communicates with agent \( \ell \). If \( \ell \in S_d \), update \( S_d \leftarrow S_d \setminus \{\ell\} \) and send the bundle \( \{\alpha^c, v, f, S_d, S_c\} \) to agent \( \ell \).

**Step 6** If \( S_d = \emptyset \), end procedure. Otherwise, return to Step 5.

From Step 1 to Step 4, agent \( i \) initializes \( v \), \( f \), \( S_d \), and \( S_c \). Agent \( i \) sets \( v = 1 \) and adds \( \alpha^c \) to the taboo list \( T_i \) in Step 2 and 3 since \( \alpha^c \) is completely investigated. After initialization, agent \( i \) iterates Step 5 whenever \( S_d \) is not empty set. Note that agent \( i \) do not send the same bundle to agent \( \ell \).

Let \( N_i \) be the set of neighbors of agent \( i \). If agent \( i \) received the bundle \( \{\alpha^c, v, f, S_d, S_c\} \) from a neighboring agent, agent \( i \) executes the following procedure.

**Step 1** If \( \alpha^c \in T_i \), end this procedure.

**Step 2** If \( v = 1 \), update \( T_i \leftarrow T_i \cup \{\alpha^c\} \) and go to Step 6.

**Step 3** If \( i \in S_c \), go to Step 6.

**Step 4** Checks feasibility of \( \alpha^c \). If \( \alpha^c \) is no more candidate solution, end this procedure and execute the first procedure as a proved infeasible solution. Otherwise, that is, if \( \alpha^c \) is still the candidate solution, update \( f \leftarrow f + f_i \).

**Step 5** Update \( S_c \leftarrow S_c \cup \{i\} \). If \( S_c = A \), end this procedure and execute the first procedure as a proved feasible solution.

**Step 6** If \( |N_i| = 1 \), set \( S_d \leftarrow A \setminus \{i\} \).

**Step 7** Wait until next communication. Suppose that agent \( i \) communicates with agent \( \ell \). If \( \ell \in S_d \), update \( S_d \leftarrow S_d \setminus \{\ell\} \) and send the bundle \( \{\alpha^c, v, f, S_d, S_c\} \) to agent \( \ell \).
(Step 8) If \( S_d = \emptyset \), end procedure. Otherwise, return to Step 7.

In Step 2, agent \( i \) recognizes the fact that we finished the investigation of \( \alpha \) completely. That is why agent \( i \) adds \( \alpha \) to the taboo list \( T_i \). In Step 3, agent \( i \) skips Step 3, 4, and 5 since agent \( i \) has already dealt with \( \alpha \) once.

We should note that it is easy to check feasibility in Step 4 since the order of it is just \( O(n) \).

In Step 5, agent \( i \) checks whether investigation of \( \alpha \) has done or not. If \( S_i = A \) is satisfied, it means that all of the knapsack constraints (2) are satisfied.

In Step 6, this is a specific procedure for leaf nodes. Although leaf nodes have only one neighbor and they do not have to transfer information when they receive it, it is possible to be \( S_d \neq \emptyset \) at leaf nodes, that is, we need more information to finish the investigation. We therefore force leaf nodes to send information back in order to complete investigation of \( \alpha \).

We should point out that this protocol works well if some agents generate the same candidate \( \alpha \) independently. Once at least one bundle information related to \( \alpha \) enters the taboo list, all of the bundle information related to this candidate will be abandoned due to Step 1 in the second procedure and agents finish to investigate.

3.4 Discussions

In existing algorithms [3, 4], since the assignment constraints (1) are directly relaxed to make the Lagrangian relaxation problem, we define the subgradient as \( 1 - \sum_{i \in A} x_{ij} \). That is, in order to calculate the subgradients, agent \( i \) has to gather assignment values of \( x_{ij} \) from all of the neighboring agents related to job \( j \). This brings us a couple of disadvantages. One is that the existence of edge \((i, \ell)\) depends on the possibility of job assignment. For example, if job \( j \) has a possibility to choose both agent \( i \) and \( \ell \), agents \( i \) and \( \ell \) need to communicate with each other to calculate the subgradient \( 1 - x_{ij} - x_{\ell j} \). Namely, a given problem instance strictly determines topology of communication networks. The other is that the calculation of subgradients forces us exchanging information synchronously since the algorithms have to use the same Lagrangian multipliers. That is why the existing algorithms are all synchronous.

On the other hand, we can calculate the subgradient \( s^{(i)}_{ij} \) with no neighbors’ information in the proposed algorithm. That is, agent \( i \) do not have to send any information except the copy of the Lagrangian multiplier vector \( \mu^{(i)}[k] \). This enables us to reduce the amount of information compared to the existing algorithms and using arbitrary topology for communication networks. Moreover, since we need not to use the same Lagrangian multipliers, we do not have to synchronize the communication among the agents.

One drawback of the proposed algorithm is that we cannot compute the actual upper bound of the original problem. Let us define the objective function \( L^{(i)}(\mu^{(i)}[k]) \) of agent \( i \) at \( \mu^{(i)}[k] \) as

\[
L^{(i)}(\mu^{(i)}[k]) = L^i_0(\mu^{(i)}[k]) + \frac{1}{|A|} \sum_{j \in J} L^j_0(\mu^{(i)}[k]).
\]

Then, the entire objective function

\[
L'(\mu^{(1)}[k], \mu^{(2)}[k], \ldots, \mu^{(m)}[k]) = \sum_{i \in A} L^{(i)}(\mu^{(i)}[k])
\]

needs not to be the upper bound of the original problem. Note that \( L'(\mu^{(1)}[k], \mu^{(2)}[k], \ldots, \mu^{(m)}[k]) \) the upper bound if \( \mu^{(1)}[k] = \mu^{(1)}[k] = \cdots = \mu^{(m)}[k] \) holds.

The advantage of the proposed heuristic protocol is that each agent can generate a variety of the candidate solutions independently. In existing protocols, it is difficult to generate the actual feasible solutions [3, 4] due to the assignment constraints (1) and we need to update the Lagrangian multiplier synchronously. Thus, the agents need to corporate to generate them. On the other hand, all we have to do is to check feasibility of the knapsack constraints (2) in our protocol. Moreover, it is easy to generate a lot of candidates since \( \mu^{(i)}[k] \) need not be the same value.

4 Experiments

We made experiments to show the performance of our proposed algorithm. The objective is to show the effectiveness of the algorithm how good the quality of feasible solutions the algorithm generates.

In the proposed algorithm, we used several kinds of local step size \( r_i[k] \). Protocol Asy1.0 shows that we set \( r_i[k] \) as constant number 1.0 for any \( i \) and \( k \). Protocol Asynlastic shows that we set \( r_i[k] \) as a monotonic decreasing function as \( r_i[k] = 1/c_i[k+1] \), where

\[
c_i[k+1] = \begin{cases} 
   c_i[k] + 1 & \text{if the local clock ticks}, \\
   c_i[k] & \text{otherwise},
\end{cases}
\]  

and \( c_i[1] = 1 \) for any \( i \). We selected a random value following the uniform distribution for each initial value of the Lagrangian multiplier \( \mu^{(i)}[1] \in [0, p_{ij}] \). Note that this initial value can be computed by agent \( i \) without any information of its neighbors. We used 20000 rounds as the stopping criteria for the proposed algorithm and we executed five times for each instances with different initial conditions as we mentioned above.

For comparison purpose, we implemented existing DisLRP [3] since the amount of messages are smallest among the existing ones [4, 5]. Note that DisLRP is synchronous while the proposed algorithm is asynchronous. Therefore, we have to pay much attention to fair comparisons. In order to fairly evaluate, we decided the number of cut off rounds so that the number or the amount of communications are the same of the
Table 1: Experimental results

| Instance | #agent | #job | Opt. | Protocol | Round | time(min) | BstVal | BstVal/Opt. |
|----------|--------|------|------|----------|-------|-----------|--------|-------------|
| c1060-1  | 10     | 60   | 1451 | Asy1.0   | 20000 | 571.0076  | 1444.8 | 0.9957      |
|          |        |      |      | Asy0.5   | 20000 | 672.2894  | 1433.8 | 0.9881      |
|          |        |      |      | DisLRP   | 2333  | 203.56    | 1419   | 0.9779      |
|          |        |      |      | CPLEX    | N/A   | 0.078     | 1451   | 1.0000      |
| c1060-2  | 10     | 60   | 1449 | Asy1.0   | 20000 | 556.36    | 1444.2 | 0.9967      |
|          |        |      |      | Asy0.5   | 20000 | 638.1006  | 1432   | 0.9883      |
|          |        |      |      | DisLRP   | 2333  | 196.656   | 1415   | 0.9765      |
|          |        |      |      | CPLEX    | N/A   | 0.094     | 1433   | 1.0000      |
| c1060-3  | 10     | 60   | 1433 | Asy1.0   | 20000 | 551.052   | 1427   | 0.9958      |
|          |        |      |      | Asy0.5   | 20000 | 624.5134  | 1420.4 | 0.9912      |
|          |        |      |      | DisLRP   | 2333  | 203.135   | 1396   | 0.9742      |
|          |        |      |      | CPLEX    | N/A   | 0.094     | 1433   | 1.0000      |
| c1060-4  | 10     | 60   | 1447 | Asy1.0   | 20000 | 579.7742  | 1442.4 | 0.9968      |
|          |        |      |      | Asy0.5   | 20000 | 708.9714  | 1432.8 | 0.9902      |
|          |        |      |      | DisLRP   | 2333  | 190.608   | 1447   | 1.0000      |
|          |        |      |      | CPLEX    | N/A   | 0.078     | 1447   | 1.0000      |
| d05100   | 5      | 100  | −6353| Asy50.0  | 20000 | 629.6046  | −7103.4| 1.1181      |
|          |        |      |      | DisLRP   | 5000  | 465.175   | N/A    | N/A         |
|          |        |      |      | CPLEX    | N/A   | 15.375    | −6353  | 1.0000      |
| d05200   | 5      | 200  | −12742| Asy50.0 | 20000 | 1063.3516 | −14317.8| 1.1237     |
|          |        |      |      | DisLRP   | 5000  | 588.428   | N/A    | N/A         |
|          |        |      |      | CPLEX    | N/A   | 827.875   | −12472 | 1.0000      |
| d10100   | 10     | 100  | −6347| Asy50.0  | 20000 | 549.4762  | −7704  | 1.2138      |
|          |        |      |      | DisLRP   | 2223  | 403.051   | N/A    | N/A         |
|          |        |      |      | CPLEX    | N/A   | 550.031   | −6350  | 1.0005      |
| d10200   | 10     | 200  | −12430| Asy50.0 | 20000 | 980.969   | −14998.6| 1.2066     |
|          |        |      |      | DisLRP   | 2223  | 438.388   | N/A    | N/A         |
|          |        |      |      | CPLEX    | N/A   | 982.203   | −12437 | 1.0006     |
| d 10400  | 10     | 400  | −24961| Asy50.0 | 20000 | 2347.5858 | −30502 | 1.2220     |
|          |        |      |      | DisLRP   | 2223  | 703.946   | N/A    | N/A         |
|          |        |      |      | CPLEX    | N/A   | 2352.250  | −24969 | 1.0003     |

proposed algorithm. That is, the amount of messages at a synchronous round in the existing algorithm are equivalent to \( m - 1 \) times the amount of messages in the proposed algorithm.

In the experiments, we used 10 GAP benchmark instances from the OR-Library [10] (five instances in each of gap 12 and gap d). Since the instances in gap d are the minimization problems, we converted them into maximization problems by multiplying \(-1\) to the objective function. Note that the communication networks in the existing protocols strictly depend on the structure of instance while ones in our proposed algorithm does not. For fair comparisons, we used the complete graph for both existing and proposed algorithm as a network topology.

We used ILOG CPLEX 20.1 as the solver to solve knapsack problems for each agent. We also solved GAP instances as the original formulation by CPLEX within the time which is almost the same of the longest execution one of the distributed algorithms if CPLEX could not obtain the optimal value. We implemented the algorithms by Python3 and ran on a Desktop PC (Core i7-7700 3.6GHz, 4 cores 8 threads, 8GB memory, Windows 10).

Table 1 shows the experimental results. The column Instance means the name of instances, # agent and # job show the number of agents and jobs, respectively. Opt. means the optimal value of the instance. Protocol shows the name of the algorithm we used in the experiment. Round means the cut off round for the algorithm. Time(min) denotes the shortest execution time of algorithms for some trials. BstVal shows the best lower bound that the algorithm obtained. BstVal/Opt. means the quality of the feasible solution.

Although we made experiments with Asy1.0, Asy0.5, Asydic, DisLRP, and CPLEX for the category gap 12
and Asy50.0, Asy$_{dic}$, DisLRP, and CPLEX for the category gap d, we omitted the results of Asy$_{dic}$ for both category gap 12 and gap d from Table 1, since both of them obtained no feasible solution for all trials within the cutoff round 20000.

For all instances, CPLEX obviously shows the best performance among the algorithms due to the centralized algorithm. We compare the quality of the feasible solutions obtained from the algorithms without CPLEX since we are focusing distributed algorithms.

According to Table 1, Asyn1.0 is the best for all instances except c1060-4 among the distributed algorithms in the category gap 12. That is, the proposed algorithm can obtain better quality feasible solutions than DisLRP. These experimental results imply that it is better to use larger constant local step size $r_i[k]$ in order to obtain good quality feasible solutions. On the other hand, the execution time is not so fast compared to existing DisLRP. It is one of the drawbacks of the proposed algorithm.

In the category gap d, our proposed algorithms obtained the feasible solutions while DisLRP failed to obtain ones. In DisLRP, the agents try to construct the candidate feasible solution so as to satisfy the assignment constraint (1), which is the equality constraint. However, this approach is much difficult to do it. On the other hand, candidate feasible solutions that are constructed by the agent in the proposed algorithm is already satisfied the assignment constraint. Moreover, the agents can generate many candidates without any information of its neighbors. That is why we obtain such experimental results.

Nevertheless, the quality of feasible solutions seems not to be good. We have to evaluate by using another settings of the local step size $r_i[k]$ including the monotonic decreasing function.

### 5 Concluding remarks

We proposed an asynchronous gossip-based Lagrangian heuristic algorithm for GMAP which is the combinatorial maximization problem in distributed environments. We introduced Lagrangian decomposition based formulation of GMAP in order to apply the asynchronous gossip algorithm. Our contribution of this study is that We achieved the distributed algorithm that can obtain good quality feasible solutions for GMAP. Moreover, our proposed algorithm can generate feasible solutions of GMAP even when instances are difficult or very large.

In our experiments, the network topology which defines the availability of communications is a complete graph. In the future, we have to consider the other topology such as star, line, and more general cases. We also would like to make experiments with much larger instances which are not be able to be computed by a single computer.

### References

[1] A. Nedić and A. E. Ozdaglar: Distributed subgradient methods for multi-agent optimization, *IEEE Transactions on Automatic Control*, Vol. 54, No. 1, pp. 48–61, 2009.

[2] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Asynchronous gossip algorithms for stochastic optimization,” in *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 3581–3586, 2009.

[3] K. Hirayama, A new approach to distributed task assignment using Lagrangian decomposition and distributed constraint satisfaction, In proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), pp.660–665, 2006.

[4] K. Hirayama, T. Matsui, M. Yokoo: Adaptive Price Update in Distributed Lagrangian Relaxation Protocol Proceedings of the 8th International Joint Conference on Autonomous Agents & Multi-Agent Systems, pp.1033–1040, 2009.

[5] K. Hanada, K. Hirayama, T. Okimoto: Effect of Bundle Method in Distributed Lagrangian Relaxation Protocol, AAAI-15 Workshop on Planning, Search, and Optimization (PlanSOpt-15), 2015.

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

[7] G. Aristomenopoulos, T. Kastrinogiannis, and S. Papavassiliou: Multiaccess multicell distributed resource management framework in heterogeneous wireless networks, *IEEE Transactions on Vehicular Technology*, Vol.61, No.6, pp. 2636–2650, 2012.

[8] L. Luo, N. Chakraborty, and K. Sycara: Distributed algorithm design for multi-robot generalized task assignment problem, Proceedings of 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4765–4771 2013.

[9] M. Guignard and S. Kim: Lagrangian decomposition: A model yielding stronger lagrangean bounds, *Mathematical Programming*, 39, pp. 215–228, 1987.

[10] Beasley, J.E.: Welcome to OR-Library, OR-Library (online), available from (http://people.brunel.ac.uk/~mastjjb/jeb/info.html).