EPiC Series in Computing

Volume 81, 2022, Pages 141–152

Proceedings of 11th International Congress on Advanced Applied Informatics

Adaptive Step Size for a Consensus based Distributed Subgradient Method in Generalized Mutual Assignment Problem

Yuki Amemiya[1], Kenta Hanada[1], and Kenji Sugimoto[1]

Graduate School of Science and Technology, Nara Institute of Science and Technology, Ikoma-shi, Nara, JAPAN

{ amemiya.yuki.bz3, hanada, kenji }@is.naist.jp

Abstract

Generalized Mutual Assignment Problem (GMAP) is a multi-agent based distributed optimization where the agents try to obtain the most profitable job assignment. Since it is NP-hard and even a problem of judging the existence of a feasible solution is NP-complete, it is a challenging issue to solve GMAP. In this paper, a consensus based distributed subgradient method is considered to obtain feasible solutions of GMAP as good as possible. Adaptive step size which is calculated by the lower and estimated upper bounds is proposed for the step size in the subgradient method. In addition, a protocol how to estimate the upper bound is also proposed, where each agent do not have to synchronize it.

1 Introduction

Distributed optimization in multi-agent systems is a problem where multiple agents cooperatively update their states in order to maximize/minimize the objective function of the entire system under several constraints. Since algorithms to solve such a problem have scalability, that is, they are applicable to deal with complex or large scale systems, or good properties to secure the agents' privacy, many problems and its algorithms have been studied.

Generalized Mutual Assignment Problem (GMAP) [3, 4, 5, 6] is a 0-1 integer programming problem which is applicable to many real world issues such as wireless communication networks [1] or multi-robot task assignment [7]. The objective of this problem is to find the most profitable job assignment to the agents under two constraints. One is an assignment constraint where each job is assigned to exactly one agent and the other is a knapsack constraint that restricts the resource consumption when agent selects a certain job. Recently, many distributed algorithms/protocols [3, 4, 5, 6] have been proposed to solve GMAP instances. We can classify existing algorithms into two categories in terms of a research objective. The studies [3, 4, 5] aim to achieve feasible solutions (lower bound) as good as possible. On the other hand, the protocol [6] seeks to obtain good upper bound for the optimal value. Note that this protocol computes the estimated lower bound which is not guaranteed as the actual lower bound.

In this paper, we propose adaptive step size for a consensus based distributed subgradient method in GMAP to obtain feasible solutions as good as possible. The main idea of our proposed

algorithm is to employ adaptive step size which is calculated by the lower and estimated upper bounds. It is inspired by Polyak [10] in a centralized manner and by the study [6] in a distributed manner. Note that the lower and upper bounds have to be synchronized in Polyak type or the protocol [6]. However, in the basic procedure of our algorithm, the states of the agents do not have to be synchronized in principle. We therefore consider an algorithm that works well even though the lower and estimated upper bounds also do not have to be synchronized. In addition, a protocol how to estimate the upper bound is also proposed, where each agent do not have to synchronize it. We show the effectiveness of our proposed algorithm by numerical experiments.

2 Generalized Mutual Assignment Problem

Let $m \in \mathbb{N}$ be the number of agents in a multi-agent system and $n \in \mathbb{N}$ be the number of jobs to be assigned to the agents. We denote $V = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$ as a set of indices for the agents and jobs, respectively. We suppose that job $j \in J$ is assigned to exactly one agent $i \in V_j$ (an assignment constraint), where $V_j \subseteq V$ is a non-empty subset of V. We also suppose that agent $i \in V$ can choose some jobs $j_1, j_2, \ldots, j_{n'} \in J_i$ with a resource constraint (a knapsack constraint), where $J_i \subseteq J$ is a non-empty subset of J. Let $b_i \in \mathbb{N}$ be the cardinality of the set J_i and $b \in \mathbb{N}$ be the sum of b_i over $i \in V$, that is, $b = \sum_{i \in V} b_i = \sum_{i \in V} |J_i|$. In this paper, we consider a problem to find the most profitable job assignment under these two constraints in a distributed manner.

Then, this problem can be formulated as the Generalized Mutual Assignment Problem (GMAP) [3, 4, 5, 6] which is the following 0-1 integer programming problem:

$$x^* = \arg\max_{x \in \{0,1\}^b} \left\{ \sum_{i \in V} \sum_{j \in J_i} p_{ij} x_{ij} \,\middle|\, \sum_{i \in V_j} x_{ij} = 1 \ (j \in J), \ \sum_{j \in J_i} w_{ij} x_{ij} \leq c_i \ (i \in V) \right\}, \tag{1}$$

where $x^* = (x_i^* | i \in V) \in \{0,1\}^b$ ($x_i^* = (x_{ij}^* | j \in J_i) \in \{0,1\}^{b_i}$) is the optimal solution whose element $x_{ij}^* \in \{0,1\}$ is a variable assignment under the optimal solution in an arbitrary order[1], $x = (x_{ij} | i \in V, j \in J_i) \in \{0,1\}^b$ is a decision vector whose element $x_{ij} \in \{0,1\}$ is a binary decision variable, $p_{ij} \in \mathbb{R}$ and $w_{ij} \in \mathbb{R}$ are the profit and weight of agent i when agent i selects job j, and $c_i \in \mathbb{R}$ is the capacity of agent i. Note that this formulation is equivalent to the Generalized Assignment Problem (GAP) [11, 12], which has been well studied for over four decades, if both $V_j = V$ and $J_i = J$ hold for all $i \in V$ and $j \in J$.

GMAP is known to be *NP-hard* and even a problem of judging the existence of a feasible solution is *NP-complete*. In this paper, we assume that a feasible region of GMAP is non-empty, that is, there exists at least one feasible solution. Then, the objective of this paper is to develop a multi-agent based distributed heuristic algorithm in order to obtain feasible solutions as good as possible.

2.1 Lagrangian Decomposition based Reformulation

In order to solve the problem (1) in a distributed manner, we employ the Lagrangian decomposition, which is proposed and analyzed in [2], to divide the problem (1) into subproblems.

[1] We assume that all of the vectors denoted in this paper have the same structure and order of x^*. We omit precise expressions about vector structures due to the space limitation. For example, the vector x has $x_i = (x_{ij} | j \in J_i)$ as an element.

Following the study [3], we first reformulate the original problem (1) as

$$x^*, z^* = \underset{x\in\{0,1\}^b, z\in\{0,1\}^b}{\arg\max} \left\{ \sum_{i\in V}\sum_{j\in J_i} p_{ij}x_{ij} \,\middle|\, z_{ij} - x_{ij} = 0 \ (i \in V,\ j \in J_i),\ \sum_{i\in V_j} z_{ij} = 1 \ (j \in J),\ \sum_{j\in J_i} w_{ij}x_{ij} \le c_i \ (i \in V) \right\}, \tag{2}$$

where $z = (z_{ij} | i \in V, j \in J_i) \in \{0,1\}^b$ is an additional decision vector whose element $z_{ij} \in \{0,1\}$ is a binary decision variable called *copy variable* and z^* is the optimal solution corresponding to the decision vector z in the problem (2). The constraint $z_{ij} - x_{ij} = 0$ $(z_{ij} = x_{ij})$ in the problem (2) is called a *copy constraint*. Note that this reformulation (2) is obviously equivalent to the original problem (1).

Dualizing all of the copy constraints in the problem (2), we obtain the following *Lagrangian relaxation problem*:

$$\xi^*(\mu), \zeta^*(\mu) = \underset{x\in\{0,1\}^b, z\in\{0,1\}^b}{\arg\max} \left\{ \sum_{i\in V}\sum_{j\in J_i} p_{ij}x_{ij} + \sum_{i\in V}\sum_{j\in J_i} \mu_{ij}\left(z_{ij} - x_{ij}\right) \,\middle|\, \sum_{i\in V_j} z_{ij} = 1 \ (j \in J),\ \sum_{j\in J_i} w_{ij}x_{ij} \le c_i \ (i \in V) \right\}, \tag{3}$$

where $\mu = (\mu^*_{ij} | i \in V, j \in J_i) \in \mathbb{R}^b$ is a Lagrangian multiplier vector whose element $\mu_{ij} \in \mathbb{R}$ is called a Lagrangian multiplier, $\xi^* : \mathbb{R}^b \to \{0,1\}^b$ is a map function to the optimal solution corresponding to x at a given μ in the Lagrangian relaxation problem (3), and $\zeta^* : \mathbb{R}^b \to \{0,1\}^b$ is the one corresponding to z.

We are now ready to divide the problem into subproblems. Since there is no cross term among the decision variables and each constraint consists of either x_{ij} or z_{ij}, the subproblem related to agent i can be described as

$$\tilde{L}_i(\mu_i) = \sum_{j\in J_i} \left(p_{ij} - \mu_{ij}\right) \xi^*_{ij}(\mu_i),\ \xi^*_i(\mu_i) = \underset{x_i\in\{0,1\}^{b_i}}{\arg\max} \left\{ \sum_{j\in J_i} \left(p_{ij} - \mu_{ij}\right) x_{ij} \,\middle|\, \sum_{j\in J_i} w_{ij}x_{ij} \le c_i \right\}, \tag{4}$$

where $\tilde{L}_i(\mu_i)$ is the optimal value of the subproblem (4) at μ_i. The subproblem related to the jobs can be also described as

$$\hat{L}(\mu) = \sum_{i\in V}\sum_{j\in J_i} \mu_{ij}\zeta^*_{ij}(\mu), \quad \zeta^*(\mu) = \underset{z\in\{0,1\}^b}{\arg\max} \left\{ \sum_{i\in V}\sum_{j\in J_i} \mu_{ij}z_{ij} \,\middle|\, \sum_{i\in V_j} z_{ij} = 1 \ (j \in J) \right\}, \tag{5}$$

where $\hat{L}(\mu)$ is the optimal value of the subproblem (5) at μ.

In our distributed algorithm design, each agent has a responsibility of only its own subproblem (4). In contrast, since there is no local information in the subproblem (5), all of the agents are able to solve the subproblem (5) for the jobs if the agents have a complete information of the Lagrangian multiplier vector μ.

2.2 Lagrangian Heuristic Algorithm

Here we recall that our objective of this study is to obtain feasible solutions of the problem (1) in a distributed manner. We therefore introduce a *Lagrangian heuristic algorithm* to do it. The Lagrangian heuristic algorithm can be regarded as a map function in order to generate (candidate) feasible solutions of the original problem (1) by using information of the Lagrangian relaxation problem (3) (equivalently the subproblems (4) and (5)) at a given μ.

In this paper, we directly use the map function $\zeta^*(\mu)$ in (5) as the Lagrangian heuristic algorithm. The function $\zeta^*(\mu)$ immediately returns a candidate feasible solution of the original problem (1) since $\zeta^*(\mu)$ satisfies the assignment constraints. Thus, if the candidate satisfies all of the knapsack constraints, it is in fact the feasible solution of the original problem (1). To check whether the candidate does satisfy the knapsack constraints or not in a distributed manner, we follow the study [3] and thus we omit the detail.

2.3 Upper Bound and Lagrangian Dual Problem

In this section, we discuss how to obtain feasible solutions as good as possible by the Lagrangian heuristic algorithm as we mentioned in the previous section. We first define an upper bound of the original problem (1). Let $f^* \in \mathbb{R}$ be the optimal value of the problem (1) and $L : \mathbb{R}^b \to \mathbb{R}$ be a function defined by

$$L(\mu) = \hat{L}(\mu) + \sum_{i \in V} \tilde{L}_i(\mu_i).$$

Then, we have the following statement by the basic duality theory:

Lemma 1. *The function $L(\mu)$ provides an upper bound on the optimal value f^* of the original problem (1) for any $\mu \in \mathbb{R}^b$, that is, the following inequality holds:*

$$f^* \leq L(\mu), \qquad \forall \mu \in \mathbb{R}^b. \tag{6}$$

We then introduce an error $e \in \mathbb{R}$ between the optimal value f^* and the objective value at a certain candidate $\zeta^*(\mu)$ obtained by the Lagrangian heuristic algorithm as

$$e = \left| f^* - F(\zeta^*(\mu)) \right|, \qquad F(\zeta^*(\mu)) = \sum_{i \in V} \sum_{j \in J_i} p_{ij} \zeta^*_{ij}(\mu).$$

Note that $\zeta^*(\mu)$ might be an infeasible solution of the original problem (1), that is, $f^* - F(\zeta^*(\mu))$ can be negative. Applying the inequality (6) to the error, we have

$$e = |f^* - F(\zeta^*(\mu))| \leq |L(\mu) - F(\zeta^*(\mu))| \tag{7}$$

for any $\mu \in \mathbb{R}^b$. Then, we have the following lemma.

Lemma 2. *If there exists some μ such that the optimal solution $(\xi^*(\mu), \zeta^*(\mu))$ satisfies all of the copy constraints appeared in the problem (2), then this optimal solution constitutes the optimal solution of the original problem (1) and thus the error e is equal to 0.*

Proof. As the first claim can be seen in [6], we only prove the second claim. From the inequality

(7), we have

$$e \leq |L(\mu) - F(\zeta^*(\mu))| = \left| \sum_{i \in V} \sum_{j \in J_i} p_{ij} \xi^*_{ij}(\mu) + \sum_{i \in V} \sum_{j \in J_i} \mu_{ij} \left(\zeta^*_{ij}(\mu) - \xi^*_{ij}(\mu) \right) - \sum_{i \in V} \sum_{j \in J_i} p_{ij} \zeta^*_{ij}(\mu) \right|$$
$$= \left| \sum_{i \in V} \sum_{j \in J_i} (p_{ij} - \mu_{ij}) \left(\xi^*_{ij}(\mu) - \zeta^*_{ij}(\mu) \right) \right|.$$

Since $\zeta^*_{ij}(\mu) - \xi^*_{ij}(\mu) = 0$ holds for all $i \in V$ and $j \in J_i$, we have $e = |f^* - F(\zeta^*(\mu))| \leq 0$. □

Clearly, the error e should be minimized. We therefore consider the following optimization

$$\min e = \min_{\mu \in \mathbb{R}^b} |f^* - F(\zeta^*(\mu))| \leq \min_{\mu \in \mathbb{R}^b} |L(\mu) - F(\zeta^*(\mu))| = \left| \min_{\mu \in \mathbb{R}^b} L(\mu) - F(\zeta^*(\mu^*)) \right|,$$

where

$$\mu^* = \arg\min_{\mu \in \mathbb{R}^b} L(\mu) = \arg\min_{\mu \in \mathbb{R}^b} \left(\hat{L}(\mu) + \sum_{i \in V} \tilde{L}_i(\mu_i) \right). \tag{8}$$

This problem (8) is called the *Lagrangian dual problem*.

We should notice here that there may exists the *integrality gap* since GMAP is the 0-1 integer programming problem. That is, there exist some instances such that $f^* < L(\mu)$ holds. In this case, the error e remains positive value even when we obtain μ^*.

3 Proposed Algorithm

In this section, we explain the detail of our algorithm. In order to solve the Lagrangian dual problem (8) in a distributed manner, we first consider the following reformulation of the Lagrangian dual problem (8):

$$L(\mu^*) = \min_{\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(m)}} \left\{ \sum_{i \in V} L_i(\mu^{(i)}) \middle| \mu^{(1)} = \mu^{(2)} = \cdots = \mu^{(m)} \right\}, \tag{9}$$

where $\mu^{(i)} \in \mathbb{R}^b$ is a copy variable of the Lagrangian multiplier vector that agent i has and

$$L_i(\mu^{(i)}) = \tilde{L}_i(\mu^{(i)}_i) + \frac{1}{m} \hat{L}(\mu^{(i)}).$$

The constraint appeared in the problem (9) is called *agreement* or *consensus* term. We should notice here that this kind of the problems can be solved by consensus based distributed algorithms developed and analyzed in studies [8, 9].

In such algorithms, each agent iteratively communicates and updates its own copy variable $\mu^{(i)}[k] \in \mathbb{R}^b$ at round $k \in \mathbb{N}$ so that all of the copy is going to the same of the optimal value of the Lagrangian dual problem. Formally, the algorithm aims to the consensus:

$$\lim_{k \to \infty} \mu^{(1)}[k] = \lim_{k \to \infty} \mu^{(2)}[k] = \cdots = \lim_{k \to \infty} \mu^{(m)}[k] = \mu^*. \tag{10}$$

In this paper, we follow the multi-agent based distributed subgradient method [9] for a basic procedure of our algorithm to update the copy variable $\mu^{(i)}[k]$. We also follow a distributed algorithm to check the feasibility of candidate feasible solutions [3]. Then, we propose a novel approach, *adaptive step size rule*, for the step size in the distributed subgradient method.

3.1 Agent and Communication Model

Let us consider a multi-agent system on a network $G = (V, E, A)$, where G is a weighted undirected graph, $E \subseteq V \times V$ is a set of edges, and $A \in \mathbb{R}^{m \times m}$ is an adjacency matrix whose element a_{ij} is positive if and only if an edge (h, i) exists otherwise 0. Let $N_i \subset V$ be a set of neighbors for agent i, that is, $h \in N_i$ if $(i, h) \in E$.

In this paper, we assume that agent i is able to send/receive messages to/from its neighbor $h \in N_i$. We also assume that the weighted graph G is connected and the adjacency matrix is doubly stochastic, which is the matrix that satisfies the following properties:

$$\sum_{h \in V} a_{hi} = 1 \ (i \in V), \qquad \sum_{i \in V} a_{hi} = 1 \ (h \in V), \qquad 0 \leq a_{hi} \leq 1 \ (h, i \in V).$$

It means that both the sum of rows and columns in A is one and the elements in A are all non-negative. Note that the necessity of doubly stochastic arises from requirements of the distributed subgradient method in [9].

In the multi-agent system, we assume that agent i only knows $V, J, V_j \ (j \in J), J_\ell \ (\ell \in V)$, N_i, i-th column of A, $p_{ij} \ (j \in J_i), w_{ij} \ (j \in J_i)$, c_i, and $\mu^{(i)}[k]$. Under this assumption, the map function ξ_i^* and its variable assignment $\xi_i^*(\mu^{(i)}[k])$ can be used and computed only by agent i and the function L_i is also a local function. On the other hand, any agents can deal with the map function ζ^* since the subproblem (5) includes no local information. Note that the variable assignment $\zeta^*(\mu^{(i)}[k])$ is local information since $\mu^{(i)}[k]$ is local one.

3.2 Consensus based Distributed Subgradient Method

In this section, we denote a basic procedure of the consensus based distributed subgradient method. We first explain the subgradient, which plays an important roll in order to solve the Lagrangian dual problem (9). We say that $s(\mu^{(i)}[k]) = (s_{ij}(\mu^{(i)}[k]) | i \in V, j \in J_i) \in \mathbb{R}^b$ is a subgradient of the function L at $\mu^{(i)}[k]$ when the following relation holds:

$$L(\mu) \geq L(\mu^{(i)}[k]) + \langle s(\mu^{(i)}[k]), \mu - \mu^{(i)}[k] \rangle, \qquad \forall \mu \in \mathbb{R}^b,$$

where $\langle \cdot, \cdot \rangle$ is the inner product of given vectors. Since it is well known that the subgradient in the Lagrangian dual problem (9) corresponds to the relaxed constraint in the Lagrangian relaxation problem (3), we can define each element $s_{ij}(\mu^{(i)}[k])$ of the subgradient $s(\mu^{(i)}[k])$ as

$$s_{ij}(\mu^{(i)}[k]) = \zeta_{ij}^*(\mu^{(i)}[k]) - \xi_{ij}^*(\mu^{(i)}[k]). \tag{11}$$

We should point out that only agent i can calculate the subgradient $s_{ij}(\mu^{(i)}[k]) \ (j \in J_i)$, without any additional information from neighbors since the optimal solution $\xi_{ij}^*(\mu^{(i)}[k])$ is required.

We are now ready to describe the basic procedure of agent i in the consensus based distributed subgradient method. Agent i executes the following procedure:

(Step 1: Initialization) Set $k = 1$ and a random value to $\mu_{\ell j}^{(i)}[k] \ (\ell \in V, \ j \in J_h)$.

(step 2: Communication) Send $\mu^{(i)}[k]$ to all of its neighbors and receive $\mu^{(h)}[k]$ from all of its neighbors $(h \in N_i)$.

(Step 3: Optimization) Calculate a weighted average $\bar{\mu}^{(i)}[k] = (\bar{\mu}_{\ell j}^{(i)}[k] | \ell \in V, \ j \in J_\ell) \in \mathbb{R}^b$ of $\mu^{(i)}[k]$ as

$$\bar{\mu}_{\ell j}^{(i)}[k] = \sum_{h \in N_i \cup \{i\}} a_{\ell i} \mu_{\ell j}^{(h)}[k], \qquad \forall \ell \in V, \ \forall j \in J_\ell.$$

Solve its own subproblem (4) and the subproblem (5) for the jobs at $\bar{\mu}^{(i)}[k]$. Calculate the subgradient $s_{ij}(\mu^{(i)}[k])$ $(j \in J_i)$. Update its own $\mu^{(i)}_{\ell j}[k]$ $(\ell \in V,\ j \in J_\ell)$ as

$$\mu^{(i)}_{\ell j}[k+1] = \begin{cases} \bar{\mu}^{(i)}_{\ell j}[k] - r_i[k] s_{ij}(\bar{\mu}^{(i)}[k]) & (\ell = i), \\ \bar{\mu}^{(i)}_{\ell j}[k] & (\ell \neq i), \end{cases} \tag{12}$$

where $r_i[k] \in \mathbb{R}$ is a positive local step size of agent i to be determined later.

(Step 4: Iteration) Tick $(k \leftarrow k+1)$ and return to Step 2.

In Step 1, agent i initializes its Lagrangian multiplier vector $\mu^{(i)}[1]$. Note that $\mu^{(i)}[1]$ need not to be the same of the other $\mu^{(\ell)}[1]$ $(\ell \in V \setminus \{i\})$. In Step 2, neighboring agents communicate with each other to exchange their Lagrangian multiplier vector.

In Step 3, agent i first calculates the weighted average among $\mu^{(h)}[k]$ $(h \in N_i \cup \{i\})$. This procedure works for achieving the consensus (10). Then, agent i updates $\mu^{(i)}[k+1]$ by the subgradient method in (12) to solve the Lagrangian dual problem (9). Note that agent i achieves a candidate feasible solution $\zeta^*(\bar{\mu}^{(i)}[k])$ at Step 3 since it solves the subproblem (5) for the jobs at $\bar{\mu}^{(i)}[k]$. In this paper, we assume that we can check whether this candidate $\zeta^*(\bar{\mu}^{(i)}[k])$ does actually feasible or not by the distributed algorithm followed by the study [3].

The local step size $r_i[k]$ is the important parameter for the subgradient method in order to achieve good convergence properties. For example, the subgradient method in (12) converges to the consensus (10) if we use the following diminishing step size [9]:

$$r_i[k] = r[k] = \frac{d}{k} \qquad \forall i \in V, \tag{13}$$

where $d \in \mathbb{R}$ is a positive parameter.

The advantage of the above step size is easy to implement in a distributed manner and it is possible to obtain feasible solutions if we give appropriate parameters for each problem instance. However, it is in fact difficult to do it. We therefore consider to develop another type of step size to achieve our research objective.

3.3 Adaptive Step Size and Estimated Upper Bound

In this section, we propose adaptive step size for the consensus based distributed subgradient method. We also propose how to calculate an estimated upper bound which is used by the update in the proposed algorithm.

Our study is inspired by the Polyak type adaptive step size [6, 10] which is described as

$$r[k] = \pi \frac{L(\mu[k]) - f^*}{\|s(\mu[k])\|^2},$$

where $\pi \in \mathbb{R}$ is a positive parameter. Although the subgradient method with the Polyak type step size has a good convergence result both mathematically and empirically in a centralized manner [10], we cannot use this step size directly in the distributed subgradient method. First of all, it is obvious not to be able to use optimal value f^*. Other problem is how to calculate $L(\mu[k])$ which is the global information. Since it is the multi-agent based distributed algorithm and each agent has the different Lagrangian multiplier vector, it takes too much time to decide a certain common μ among the agent, in other words, to synchronize μ among them. Furthermore, it also takes too much time to collect information from the agents in order to calculate $L(\mu[k])$. We therefore would like to develop the step size without any synchronization among the agents.

Let $\beta^{(i)}[k] \in \mathbb{R} \cup \{+\infty\}$ be an estimated upper bound for agent i to be described later and $\alpha^{(i)}[k] \in \mathbb{R} \cup \{-\infty\}$ be the best lower bound for agent i to be also described later. Note that these initial values are $\beta^{(i)}[1] = +\infty$ and $\alpha^{(i)}[1] = -\infty$. Then, we propose the following adaptive step size:

$$r_i[k] = \begin{cases} \pi_i[k] \dfrac{\left|\min_{t=1,\ldots,k}\{\beta^{(i)}[t]\} - \alpha^{(i)}[k]\right|}{\sum_{j \in J_i} (s_{ij}^{(i)}(\bar{\mu}^{(i)}[k]))^2} & \left(\sum_{j \in J_i} (s_{ij}^{(i)}(\bar{\mu}^{(i)}[k]))^2 \neq 0\right) \\ 0 & \left(\sum_{j \in J_i} (s_{ij}^{(i)}(\bar{\mu}^{(i)}[k]))^2 = 0\right) \\ \dfrac{d}{k} & \left(\beta^{(i)}[k] - \alpha^{(i)}[k] = +\infty\right) \end{cases} \tag{14}$$

where $\pi_i[k] \in \mathbb{R}$ is a positive parameter.

We first explain $\alpha^{(i)}[k]$, which is the best lower bound as far as the agent i knows at round k. In our algorithm, the agents are able to obtain feasible solutions since we follow the algorithm proposed by [3]. Thus, agent i can calculate the best lower bound among feasible solutions that agent i has. However, the timing when each agent knows the fact that a certain candidate feasible solution is actually feasible is different due to the communication delay. In this paper, we assume that each agent need not to use the same lower bound at round k, that is, we allow to use the different best lower bound at each iteration among the agents. To do so, we do not have to consider any additional procedure for this matter.

In our algorithm, agent i estimates the upper bound $L(\mu)$. The notation $\beta^{(i)}[k]$ indicates that agent i has $\beta^{(i)}[k]$, which is estimated by its own, at round k. The idea to calculate $\beta^{(i)}[k]$ is that the agents transfer information of $L_i(\mu^{(i)}[k])$ $(i \in V)$ and we don't care about the difference of the Lagrangian multiplier vector among the agents and the communication delay.

Let $t^{(i)}[k]$ and $\gamma^{(i)}[k]$ be vectors

$$t^{(i)}[k] = (t_1^{(i)}[k], t_2^{(i)}[k], \ldots, t_m^{(i)}[k]) \in \mathbb{Z}^m,\ \gamma^{(i)}[k] = (\gamma_1^{(i)}[k], \gamma_2^{(i)}[k], \ldots, \gamma_m^{(i)}[k]) \in (\mathbb{R} \cup \{+\infty\})^m,$$

where the initial state is

$$t^{(i)}[1] = (0, 0, \ldots, 0)\,, \qquad \gamma^{(i)}[1] = (+\infty, +\infty, \ldots, +\infty)\,.$$

Note that each agent has individual $t^{(i)}[k]$ and $\gamma^{(i)}[k]$. The ℓ-th element $t_\ell^{(i)}[k]$ of $t^{(i)}[k]$ indicates a time stamp when $\gamma_\ell^{(i)}[k]$ is calculated at agent ℓ.

Here we describe the procedure of calculating the estimated upper bound. We modify the algorithm stated in Section 3.2. We substitute the following procedure into Step 2 in the algorithm:

(step 2: Communication) Send $\mu^{(i)}[k]$, $t^{(i)}[k]$, and $\gamma^{(i)}[k]$ to all of its neighbors and receive $\mu^{(h)}[k]$, $t^{(h)}[k]$, and $\gamma^{(h)}[k]$ for all of its neighbors $(h \in N_i)$.

Next, we add the following procedure after Step 3 in the algorithm:

(step 3': Upper Bound Estimation) Let $\bar{t}_\ell$ be the maximum value among $t_\ell^{(h)}$ $(h \in N_i)$, formally, $\bar{t}_\ell = \max_{h \in N_i}\{t_\ell^{(h)}\}$. We define $\bar{h} = \arg\max_{h \in N_i}\{t_\ell^{(h)}\}$. Update $t_\ell^{(i)}[k]$ and $\gamma_\ell^{(i)}[k]$ as

$$t_\ell^{(i)}[k+1] = \begin{cases} k & (\ell = i) \\ \bar{t}_\ell & (t_\ell^{(i)} < \bar{t}_\ell) \\ t_\ell^{(i)}[k] & (\text{otherwise}), \end{cases} \qquad \gamma_\ell^{(i)}[k+1] = \begin{cases} L_i(\bar{\mu}^{(i)}[k]) & (\ell = i) \\ \gamma_\ell^{(\bar{h})}[k] & (t_\ell^{(i)} < \bar{t}_\ell) \\ \gamma_\ell^{(i)}[k] & (\text{otherwise}). \end{cases}$$

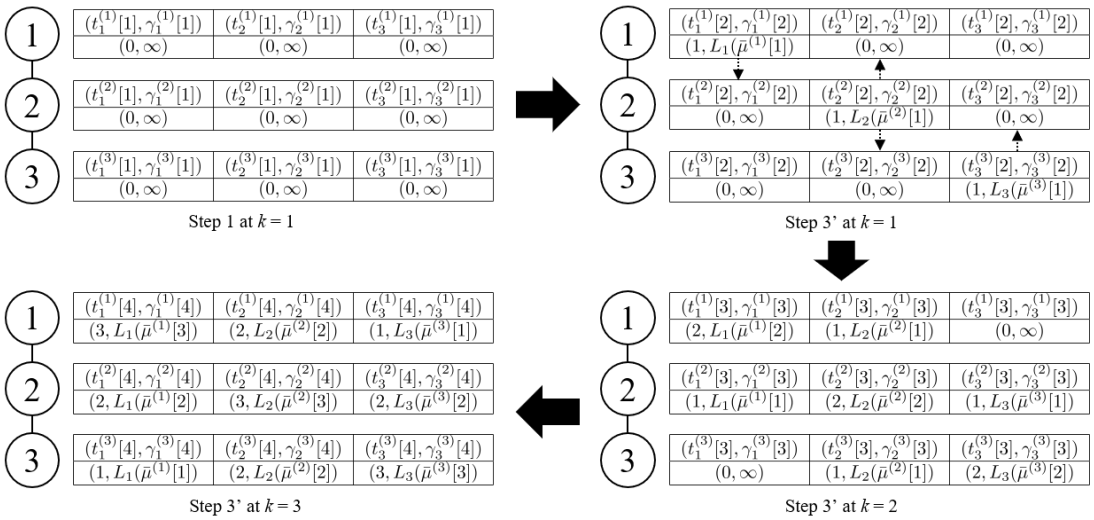

Figure 1: Example of the procedure for the estimated upper bound

Then, the estimated upper bound $\beta^{(i)}[k]$ that agent i has at round k is calculated as

$$\beta^{(i)}[k] = \sum_{\ell \in V} \gamma_\ell^{(i)}[k] = L_1(\bar{\mu}^{(i)}[t_1^{(i)}[k]]) + \cdots + L_i(\bar{\mu}^{(i)}[k]) + \cdots + L_m(\bar{\mu}^{(i)}[t_m^{(i)}[k]]).$$

Fig. 1 depicts the example where the communication network is line graph with 3 agents. A node indicates an agent and an edge indicates a neighboring relation among the agents. The table described in the right side of the agents shows the values of $t^{(i)}[k]$ and $\gamma^{(i)}[k]$ in a certain step at round k. At the beginning of the algorithm (the top left corner in Fig. 1), all of the elements $t^{(i)}[1]$ and $\gamma^{(i)}[1]$ ($i \in V$) are 0 and ∞. In Step 3' at $k = 1$ (the top right corner), agent i updates its values from $t^{(i)}[1]$ and $\gamma^{(i)}[1]$ to $t^{(i)}[2]$ and $\gamma^{(i)}[2]$. Note that agent i only updates $t_i^{(i)}[2]$ as k and $\gamma_i^{(i)}[2]$ as $L_i(\bar{\mu}^{(i)}[1])$ at this update since all of the agents received only $(0, \infty)$ from their neighbors in Step 2 at $k = 1$. Dashes arrows shows that agent i will send $t^{(i)}[2]$ and $\gamma^{(i)}[2]$ from i (tail) to its neighbor (head) in Step 2 at the next round $k = 2$. In fact, we see that these information are propagated to neighbors in the bottom right corner (Step 3' at $k = 2$). The bottom right corner in Fig. 1 shows the states in Step 3' at $k = 2$. Then, the estimated upper bound for each agent in this case can be calculated as

$$\begin{aligned}
\beta^{(1)}[3] &= \gamma_1^{(1)}[3] + \gamma_2^{(1)}[3] + \gamma_3^{(1)}[3] = L_1(\bar{\mu}^{(1)}[2]) + L_2(\bar{\mu}^{(2)}[1]) + \infty = \infty, \\
\beta^{(2)}[3] &= \gamma_1^{(2)}[3] + \gamma_2^{(2)}[3] + \gamma_3^{(2)}[3] = L_1(\bar{\mu}^{(1)}[1]) + L_2(\bar{\mu}^{(2)}[2]) + L_3(\bar{\mu}^{(3)}[1]) < \infty, \\
\beta^{(3)}[3] &= \gamma_1^{(3)}[3] + \gamma_2^{(3)}[3] + \gamma_3^{(3)}[3] = \infty + L_2(\bar{\mu}^{(2)}[1]) + L_1(\bar{\mu}^{(1)}[2]) = \infty.
\end{aligned}$$

We therefore see that agent 2 achieves the finite estimated upper bound and thus it starts to use the adaptive step size in (14) in the next round $k = 3$, while the other agents keep using the diminishing step size d/k. In Step 3' at $k = 3$ (the bottom left corner), all of the agents achieve the finite estimated upper bound at last, that is, all agents use the adaptive step size after round $k = 4$ in this example.

Table 1: Experimental results

Instance	#agent	#job	#Opt.	Alg.	Round	Count	QoLB	QoUB
d05100	5	100	−6353	Adaptive	5000.0 ± 0.0	5	1.0038 ± 0.0018	0.9995 ± 0.0000**
				decreasing	5000.0 ± 0.0	5	**1.0036 ± 0.0017**	-
				DisLRP	374.8 ± 84.5	5	1.0974 ± 0.0082	-
				DisLRP$_\alpha$	2683.2 ± 524.5	5	1.0399 ± 0.0140	-
				Asy	5000.0 ± 0.0	5	1.007 ± 0.002	-
				ADisLRP*	5000	1*	0.9995†**	0.9995
d05200	5	200	−12742	Adaptive	5000.0 ± 0.0	5	1.0021 ± 0.0003	0.9998 ± 0.0000**
				decreasing	5000.0 ± 0.0	5	**1.0018 ± 0.0006**	-
				DisLRP	598.6 ± 194.7	5	1.1108 ± 0.0033	-
				DisLRP$_\alpha$	2832.6 ± 1000.3	5	1.0341 ± 0.0138	-
				Asy	5000.0 ± 0.0	5	1.0068 ± 0.0018	-
				ADisLRP*	5000	1*	0.9999†**	0.9998
d10100	10	100	−6347	Adaptive	5000.0 ± 0.0	5	**1.0152 ± 0.0066**	0.9991 ± 0.0000**
				decreasing	5000.0 ± 0.0	5	1.0380 ± 0.0050	-
				DisLRP	492.6 ± 147.3	5	1.1506 ± 0.0096	-
				DisLRP$_\alpha$	5000.0 ± 0.0	0	-	-
				Asy	5000.0 ± 0.0	5	1.076 ± 0.006	-
				ADisLRP*	5000	1*	1.0013**	0.9991
d10200	10	200	−12430	Adaptive	5000.0 ± 0.0	5	**1.0115 ± 0.0058**	0.9996 ± 0.0000**
				decreasing	5000.0 ± 0.0	5	1.0263 ± 0.0026	-
				DisLRP	421.2 ± 57.3	5	1.1474 ± 0.0084	-
				DisLRP$_\alpha$	4540.8 ± 918.4	1	1.0377 ± 0.0000	-
				Asy	5000.0 ± 0.0	5	1.0118 ± 0.0012	-
				ADisLRP*	5000	1*	1.0002**	0.9996
d10400	10	400	−24961	Adaptive	5000.0 ± 0.0	5	**1.0117 ± 0.0076**	0.9999 ± 0.0000**
				decreasing	5000.0 ± 0.0	5	1.0246 ± 0.0067	-
				DisLRP	527.4 ± 79.8	5	1.1386 ± 0.0058	-
				DisLRP$_\alpha$	5000.0 ± 0.0	0	-	-
				Asy	5000.0 ± 0.0	5	1.0758 ± 0.0062	-
				ADisLRP*	5000	1*	1.0011**	0.9999
e05100	5	100	−12681	Adaptive	5000.0 ± 0.0	5	1.0039 ± 0.0009	0.9994 ± 0.0000**
				decreasing	5000.0 ± 0.0	5	**1.0025 ± 0.0011**	-
				DisLRP	752.4 ± 214.3	5	1.2129 ± 0.0214	-
				DisLRP$_\alpha$	5000.0 ± 0.0	0	-	-
				Asy	5000.0 ± 0.0	5	1.0314 ± 0.0345	-
				ADisLRP*	5000	1*	1.0529**	0.9994
e05200	5	200	−24930	Adaptive	5000.0 ± 0.0	5	1.0010 ± 0.0002	0.9999 ± 0.0000**
				decreasing	5000.0 ± 0.0	5	**1.0009 ± 0.0003**	-
				DisLRP	668.0 ± 180.8	5	1.3422 ± 0.0635	-
				DisLRP$_\alpha$	5000.0 ± 0.0	0	-	-
				Asy	5000.0 ± 0.0	5	1.0028 ± 0.0006	-
				ADisLRP*	5000	1*	1.0037**	0.9999
e10100	10	100	−11577	Adaptive	5000.0 ± 0.0	3	**1.0155 ± 0.0097**	0.9990 ± 0.0001**
				decreasing	5000.0 ± 0.0	3	1.0282 ± 0.0180	-
				DisLRP	1432.6 ± 1791.0	4	1.3712 ± 0.0258	-
				DisLRP$_\alpha$	5000.0 ± 0.0	0	-	-
				Asy	5000.0 ± 0.0	0	-	-
				ADisLRP*	5000	1*	1.1508**	0.9992
e10200	10	200	−23307	Adaptive	5000.0 ± 0.0	4	**1.0053 ± 0.0021**	0.9997 ± 0.0001**
				decreasing	5000.0 ± 0.0	4	1.0076 ± 0.0016	-
				DisLRP	639.6 ± 196.2	5	1.4154 ± 0.0529	-
				DisLRP$_\alpha$	5000.0 ± 0.0	0	-	-
				Asy	5000.0 ± 0.0	0	-	-
				ADisLRP*	5000	1*	1.1036**	0.9998
e10400	10	400	−45746	Adaptive	5000.0 ± 0.0	3	**1.0045 ± 0.0023**	0.9998 ± 0.0003**
				decreasing	5000.0 ± 0.0	3	1.2151 ± 0.1352	-
				DisLRP	1138.2 ± 1155.8	5	1.6316 ± 0.0249	-
				DisLRP$_\alpha$	5000.0 ± 0.0	0	-	-
				Asy	5000.0 ± 0.0	5	1.1252 ± 0.0094	-
				ADisLRP*	5000	1*	1.0523**	1.0000

* We ran the algorithm just once since it is deterministic. ** The value is not actual bound but estimated one.
† The lower bound exceeded the optimal value since it is the estimated value.

4 Experiments

We made experiments to show the performance of our proposed algorithm. The objective of our experiments is to show how the proposed algorithm (Adaptive) can generate good feasible solutions compared to the existing algorithms.

In Adaptive, agent i used the step size (14), where the initial value of $\pi_i[1]$ is 2 and if $\min_{t=1,\ldots,k}\{\beta^{(i)}[t]\}$ is not updated for 50 consecutive rounds, the value of $\pi_i[k]$ is multiplied by 1/2. We also set $d = 10mn$ for the diminishing step size.

The algorithms to be compared are the consensus based distributed subgradient method using the step size (13) (Decreasing), DisLRP [4], DisLRP$_\alpha$ [5], Asy [3], and ADisLRP [6]. In Decreasing, we set $d = 10mn$. In DisLRP, the parameters δ, l, and r were set to 0.3, mn, and 0.999, respectively. In DisLRP$_\alpha$, the parameters δ, l, and r were set to 10, mn, and 0.999, respectively. In Asy, the parameters $r_i[k]$ was set $10mn/c_i[k+1]$. See original papers if you would like to know the details of these parameters.

In Adaptive, Decreasing, and Asy, we selected a random value to the initial Lagrangian multiplier $\mu_{ij}^{(\ell)}[1] \in [p_{\ell j}, 0]$[2] following the uniform distribution. We set the same cutoff round for all algorithms as 5,000. Note that DisLRP and DisLRP$_\alpha$ were immediately terminated if the agents find a feasible solution. Since all of the algorithms (except ADisLRP) have stochastic parameters or processes, each algorithm was executed 5 times for each instance.

In the experiments, we used 10 GAP benchmark instances from the OR-Library[3]. Since the instances in gapd and gape 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 instance structure while our proposed algorithm does not. Therefore, we used the complete graph for both the existing and proposed algorithms as a network topology for fair comparisons. We used ILOG CPLEX 20.1 as a solver for the subproblem (4) for each agent. We implemented the algorithms by Python3 and ran on a Desktop PC (Core i7-10700 2.9GHz, 8 cores 16 threads, 32GB memory, Windows 10).

Table 1 shows the experimental results. The column Instance, #agent, #job, and Opt. mean a name of instances, the number of agents, the number of jobs, and the optimal value of the instance. The column Alg. shows the name of algorithms we used in our experiments. The column Round means the number of iterations when the algorithm was terminated. Count shows the number of trials in which algorithm succeeded to obtain at least one feasible solutions. QoLB means the quality of feasible solutions which is defined by the best lower bound divided by the optimal value f^*. Note that the lower bounds calculated by ADisLRP is estimated one. QoUB means the quality of upper bounds which is defined by the best upper bound divided by the optimal value f^*. Upper bounds can only calculated by the Adaptive and ADisLRP. Note that the upper bounds calculated by Adaptive is estimated one. The results include the mean and standard deviation for Round, QoLB, and QoUB, respectively.

According to Table 1, Apdaptive, decreasing, and DisLRP succeeded to generate feasible solutions at least once for all of the instances. On the other hand, DisLRP$_\alpha$ failed to generate feasible solutions in 7 instances (d10100, d10400, e05100, e05200, e10100, e10200, e10400) and Asy failed to generate feasible solutions in 2 instances (e10100, e10200). Adaptive and Decreasing generate better qualities of feasible solutions than DisLRP, DisLRP$_\alpha$ and Asy for every single instance. Furthermore, in 6 out of 10 instances, the quality of feasible solutions in Adaptive is better than that of Decreasing. In particular, Adaptive is better than Decreasing in every case with 10 agents. Since the only difference between Adaptive and Decreasing is the

[2] p_{ij} is negative for all $i \in V$, $j \in J_i$ in our experiments.
[3] http://people.brunel.ac.uk/~mastjjb/jeb/info.html

step size, this fact implies that our proposed adaptive step size can effective to generate good quality feasible solutions. In addition, the estimated upper bound obtained by Adaptive is at least 0.998 of the optimal value. We therefore see that the estimated upper bound is fairly good accuracy even though this is not mathematically proved upper bound.

5 Conclusion

We proposed the adaptive step size for the distributed subgradient method in GMAP to obtain lower bounds as good as possible. The adaptive step size can be computed by the estimated upper bound, the lower bound, and local subgradients. We also proposed the algorithm to calculate the estimated upper bound in a distributed manner. Our experiments showed that the proposed algorithm is effective to obtain good lower bounds compared to existing algorithms.

In our experiments, we only used the complete graph as the topology due to the comparison purpose. The communication delay which depends on the topology may be crucial for the performance of our algorithm. In the future, we would like to evaluate it.

References

[1] G. Aristomenopoulos, T. Kastrinogiannis, and S.Papavassiliou. Multiaccess multicell distributed resource management framework in heterogeneous wireless networks. *IEEE Transactions on Vehicular Technology*, 61(6):2636–2650, 2012.

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

[3] K. Hanada, Y. Amemiya, and K. Sugimoto. A distributed asynchronous heuristic algorithm in generalized mutual assignment problem. *Transactions of the Japanese Society for Artificial Intelligence*, 2021 (accepted).

[4] K. Hirayama. A new approach to distributed task assignment using Lagrangian decomposition and distributed constraint satisfaction. In *the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-06)*, pages 660–665, 2006.

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

[6] K. Hirayama, T. Matsui, and M. Yokoo. Adaptive price update in distributed lagrangian relaxation protocol. In *the 8th International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-2009)*, pages 1033–1040, 2009.

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

[8] I. Masubuchi, T. Wada, T. Asai, T.H.L. Nguyen, Ohta Y, and Y. Fujisaki. Distributed multi-agent optimization based on an exact penalty method with equality and inequality constraints. *SICE Journal of Control, Measurement, and System Integration*, 9(4):179–186, 2016.

[9] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[10] B.T. Polyak. *Introduction to Optimization*. Optimization Software, 1987.

[11] G. Ross and Richard Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, 8:91–103, 12 1975.

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