Neurocomputing 270 (2017) 91–100

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Fast centralized integer resource allocation algorithm and its distributed extension over digraphs[☆]

Yun Xu[a], Gangfeng Yan[a], Kai Cai[b], Zhiyun Lin[c,d,*]

[a] College of Electrical Engineering, Zhejiang University, China
[b] Urban Research Plaza, Osaka City University, Japan
[c] The School of Automation, Hangzhou Dianzi University, China
[d] The State Key Laboratory of Industrial Control Technology, College of Electrical Engineering, Zhejiang University, China

ARTICLE INFO

Article history:
Received 25 June 2016
Revised 4 February 2017
Accepted 7 March 2017
Available online 16 June 2017

Keywords:
Min-heap
Resource allocation
Distributed algorithm
Consensus
Strongly connected digraph

ABSTRACT

This paper studies the resource allocation problem with convex objective functions, subject to individual resource constraints, equality constraints, and integer constraints. The goal is to minimize the total cost when allocating the total resource D to n agents. We propose a novel min-heap and optimization relaxation based centralized algorithm and prove that it has a computational complexity of $\mathcal{O}(n\log n + n\log D)$ when the resource constraints of individual agents are $[0, D]$, which outperforms the best known multi-phase algorithm with $\mathcal{O}(n\log n\log D)$. By extending the centralized algorithm, we present a consensus based distributed optimization algorithm to solve the same problem. It is shown that the proposed distributed algorithm converges to a global minimizer provided that the digraph (representing the interaction topology of the agents) is strongly connected. All the updates used in the distributed algorithm rely only on local knowledge.

1. Introduction

The resource allocation problem (RAP) deals with how to allocate available resources to a number of users, called agents. Many optimization problems in the financial markets, smart grids, wireless sensor networks, military ad hoc networks, and cloud systems, can be modeled as a RAP. Examples include economic dispatch problems [1], power regulation [2] and take or pay fuel supply problems [3]. Moveover, in many practical examples (e.g., joint replenishment problem [4], software-testing resources allocation [5], and many others in [6–8]), the resources allocated to agents can only be integer numbers.

To formulate the integer resource allocation problem(iRAP), consider a network of n agents. For each agent $i \in \{1, \dots, n\}$, we associate a variable $x_i \in \mathbb{N}$ and a corresponding cost function $F_i(x_i)$. The iRAP is the optimization problem aiming to find an optimal integer x_i to $\underset{x_1,\dots,x_n}{\text{minimize}} \quad \sum_{i=1}^n F_i(x_i)$ under a collective equality constraint $\sum_{i=1}^n x_i = D$ and individual inequality (state) constraints $\underline{x}_i \le x_i \le \overline{x}_i$, where $\underline{x}_i, \overline{x}_i$ and D are given integers. The physical description of D is that there exists a fixed homogeneous pool with D resource units. The lower and upper bounds $\underline{x}_i$ and $\overline{x}_i$ indicate the capability of each agent.

Although many centralized optimization algorithms exist for this problem, it is to reduce the computational complexity that motivates us to revisit the problem. Besides, due to significant communications overhead required for collecting information from all the agents in large-scale networks and lacking of robustness and privacy (by requiring individual agents to provide information) [1], it is desired to design distributed algorithms for solving the iRAP. Thus in this paper, we first design a fast centralized algorithm for solving the iRAP and analyze its computational complexity, and next design a distributed algorithm for the iRAP based on only locally available information.

Since the iRAP with integer constraints can be exactly solved by a simple greedy algorithm with a computational complexity of $\mathcal{O}(D\log n)$ [9], the study of computational complexity becomes one of the main issues for the iRAP. In real applications, D is always far greater than n. Thus, the problem to reduce the computational complexity $\mathcal{O}(D\log n)$ is to reduce the complexity factor caused by D. [10] proposes a Lagrange multiplier based algorithm with a

☆ Lin's work was supported by National Natural Science Foundation of China 61673344. K. Cai's work was supported by Program to Disseminate Tenure Tracking System and Grants-in-Aid for Scientific Research No. 26870169, MEXT, Japan, and by the Open Research Project (No. ICT1600183) of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China.

* Corresponding author at: The School of Automation, Hangzhou Dianzi University, China.

E-mail addresses: kai.cai@eng.osakacu.ac.jp (K. Cai), linz@zju.edu.cn, zhiyun.lin@ieee.org (Z. Lin).

http://dx.doi.org/10.1016/j.neucom.2017.03.089
0925-2312/© 2017 Elsevier B.V. All rights reserved.

computational complexity of $\mathcal{O}(n^2(\log D)^2)$, which is better than $\mathcal{O}(D\log n)$ when $D \gg n^2$. In addition, a polynomial-time algorithm [11] is proposed and runs in $\mathcal{O}(n(\log D)^2)$, which is far better than the algorithm in [10] when $D \gg n \gg 1$. More recently, [12] proposes a multi-phase algorithm (a modified polynomial-time algorithm) that reduces the computational complexity by requiring $\mathcal{O}(n\log n\log D)$.

The first key theoretical contribution of our work is the proposition of a novel min-heap and optimization relaxation (iRAP relaxation) based centralized algorithm that runs in $\mathcal{O}(n\log n + n\log D)$, which substantially reduces the computational complexity in comparison with the existing algorithms in the literature. We first notice the fact that the relaxation for the iRAP (iRAP without integer constraints) converges exponentially, which means an approximate relaxation solution can be obtained when it runs in $\mathcal{O}(nlognD)$ with the state constraints $0 \le x_i \le D$. Besides, we explore the relation between the relaxation solution and the optimal solution, based on which we design a method to adjust the relaxation solution to an optimal solution. We show that the adjusting procedure runs in n iterations. In addition, during the adjusting procedure, we need to utilize the minimum value of $F_i(x_i) - F_i(x_i+1)$ and $F_i(x_i-1) - F_i(x_i)$. Thus, we use a min-heap to obtain the minimum value, which has a computational complexity of $\mathcal{O}(\log n)$ when executing deletion or addition. In total, we show that the computational complexity of the proposed algorithm is $\mathcal{O}(n\log n + n\log D)$. In comparison, our work achieves better computational complexity than the existing algorithm in [12].

By extending the centralized algorithm, we design a distributed algorithm for the iRAP. The difference between the distributed algorithm and the centralized one is that all the updates used in the distributed algorithm must be obtained in distributed ways. Since the initial variables of the centralized algorithm are the approximate relaxation solution, we need to design a distributed algorithm for the iRAP relaxation as well. In order to characterize the network constraints for information exchanging in distributed RAPs, graphs are adopted to describe inter-agent communication topologies. The undirected graph model is considered in [13–16], which means that each agent communicates with its neighbors in a bidirectional manner. As an application, a distributed resource allocation scheme for sensor networks with undirected graphs is proposed in [17]. However, communications are sometimes unidirectional, e.g., directional antennas and different types of omnidirectional antenna may be used in sensor networks. Besides, directed networks reduce the energy consumption and avoid potential risks because of the radiation region decreasing [18]. Thus, we adopt digraphs as a network model in this paper. More recently, a distributed bisection algorithm [19], a ratio consensus based decentralized algorithm [20], and a surplus based consensus algorithm [1] are developed to overcome the difficulties due to directional information flow in multi-agent networks to obtain the relaxation solution. However, global knowledge, which usually cannot be known in a distributed setting, is required for the algorithms or for the design of some critical parameters in these algorithms. To overcome this, a fully distributed algorithm is designed in our previous work [21].

The second key theoretical contribution of our work is to design a distributed algorithm based on the non-negative surplus based algorithm [21]. We employ the relaxation solution that is obtained from the non-negative surplus based algorithm [21], and adjust the relaxation solution to an optimal solution. During the adjusting procedure, we notice that some parameters need to be updated in distributed ways. A variety of consensus algorithms are recalled to update these parameters at each iteration.

The remainder of this paper is organized as follows. In Section 2, preliminaries and problem formulation are introduced. In Section 3.1, a min-heap and iRAP relaxation based centralized algorithm is designed and its computational complexity is analyzed. A distributed algorithm to solve the iRAP and its practical implementation are described in Section 4. At last, simulation results are presented in Section 5. We summarize our paper and state further research problems in Section 6.

Notation: $\mathbb{R}$ denotes the set of real numbers. $\mathbb{N}$ denotes the set of integer numbers. $\mathbf{1}_n$ represents the n-dimensional vector of ones and I_n represents the identity matrix of order n. The symbol $|\cdot|$ denotes the cardinality of a set. For a real number $x \in \mathbb{R}$, $\lceil x \rceil$ denotes the smallest integer greater than or equal to x, and $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Moreover, $[x]_-$ is defined as

$$[x]_- = \begin{cases} x & x \le 0, \\ 0 & x > 0. \end{cases}$$

If x is a vector, $[x]_-$ means that every entry of x takes the above function. For any $v_1, v_2 \in \mathbb{R}^{m\times n}$, we say $v_1 \succeq v_2$ if all the entries of $v_1 - v_2$ are nonnegative and $v_1 \preceq v_2$ if all the entries of $v_1 - v_2$ are nonpositive.

Denote inf as the positive infinity value, and $-inf$ as the negative infinity value.

2. Preliminaries and problem formulation

2.1. Preliminaries for graphs

A *directed graph (digraph)* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a non-empty finite set $\mathcal{V}$ of elements called *nodes* and a finite set $\mathcal{E}$ of ordered pairs of nodes called *edges*. In $\mathcal{G}$, a node i is said to be *reachable* from a node j if there exists a path from j to i. Moreover, $\mathcal{G}$ is said to be *strongly connected* if every node is reachable from every other node.

For each node $i \in \mathcal{V}$, let $\mathcal{N}_i^+ := \{j \in \mathcal{V} : (j,i) \in \mathcal{E}\}$ denote the set of its *in-neighbors*, and let $\mathcal{N}_i^- := \{l \in \mathcal{V} : (i,l) \in \mathcal{E}\}$ denote the set of its *out-neighbors*. Note that at any time k, $\mathcal{N}_i^+ \neq \mathcal{N}_i^-$ generally. In addition, $i \notin \mathcal{N}_i^+$ and $i \notin \mathcal{N}_i^-$. Let $d_i^+ := |\mathcal{N}_i^+|$ be its *in-degree* and let $d_i^- := |\mathcal{N}_i^-|$ be its *out-degree*.

2.2. Resource allocation problem with integer resources

We consider a resource allocation problem over a network of autonomous agents with integer resources. The network is modeled as a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V} = \{1, 2, \ldots, n\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. By this model, each agent is only capable of receiving messages from its in-neighbors and transmitting messages to its out-neighbors.

Each agent i of the network is associated with a local variable (resource) $x_i \in \mathbb{N}$ and a convex cost function $F_i : \mathbb{N} \to \mathbb{R}$. The local variable (resource) $x_i \in \mathbb{N}$ is subject to an inequality constraint $\underline{x}_i \le x_i \le \bar{x}_i$. Besides, the sum of multiple variables (resources) across the network is fixed. Unlike the resource allocation problem presented in [21], the variable in each agent is restricted to be integer in this paper. To sum up, the optimal resource allocation problem with integer resources (iRAP) is stated as follows:

$$\underset{x_1,\ldots,x_n}{\text{minimize}} \quad \sum_{i=1}^{n} F_i(x_i) \tag{1a}$$

subject to:

$$\underline{x}_i \le x_i \le \bar{x}_i, \forall i, \quad \text{(individual state constraints)} \tag{1b}$$

$$\sum_{i=1}^{n} x_i = D, \quad \text{(equality constraint)} \tag{1c}$$

$$x_i \in \mathbb{N}, \forall i. \quad \text{(integer constraints)} \tag{1d}$$

In order to make the problem solvable, D must satisfy $\sum_{i=1}^{n} \underline{x}_i < D < \sum_{i=1}^{n} \overline{x}_i$, and D must be an integer constant. Since the individual state constraints are equivalent to $\lceil \underline{x}_i \rceil \le x_i \le \lfloor \overline{x}_i \rfloor$. Thus, without loss of generality, we suppose that $\underline{x}_i, \overline{x}_i \in \mathbb{N}$.

We can think of D as the total amount of resource.

In this paper, we are interested in reducing the computational complexity of centralized algorithms for solving iRAP (1), where all information is aggregated into one node to do the computation. Moreover, we are also interested in distributed algorithms for solving iRAP (1), where each agent is only allowed to conduct local computation via received information from its in-neighbors. Thus, the local information structure imposed by the digraph should be considered as part of the distributed problem formulation.

If we consider $F_i(x_i): \mathbb{R} \to \mathbb{R}$, then it is assumed that the functions F_i are strictly convex and twice continuously differentiable with the second derivatives that are bounded below as in [1,13].

Assumption 1. The function $F_i(x_i)$ is twice continuously differentiable in $\mathbb{R}$ and the second derivative is lower-bounded in the interval $[\underline{x}_i, \overline{x}_i]$, i.e.,

$$\frac{dF_i^2(x_i)}{dx_i^2} \ge l_i > 0, \forall \underline{x}_i \le x_i \le \overline{x}_i,$$

where l_i is a constant.

Moreover, the following assumption is made for the digraph $\mathcal{G}$ to design distributed algorithm.

Assumption 2. The digraph $\mathcal{G}$ is strongly connected.

3. Centralized algorithm

In this section, we are going to develop a centralized algorithm with fast convergence rate to solve the integer resource allocation problem (iRAP) (1). A centralized method means that there exists a central agent to collect all information from other agents, and the computation is only done by the central agent.

The naive way to solve an optimization problem with integer variables is to simply remove the integer constraints, solve the corresponding optimization problem, and then round the entries of the solution to the original optimization problem with integer constraints. However, for most optimization problems, not only may this solution not be optimal, it may not even be feasible. Thus, more processes are needed to obtain an optimal solution for RAP (1).

To simplify the statements, we call the problem after removing the integer constraints from iRAP (1) as the *iRAP relaxation* problem in this paper. The optimal solution of the iRAP relaxation problem is called as the *relaxation solution*.

3.1. Min-Heap and iRAP relaxation based algorithm

We first notice that if the initial state is very close to the optimal solutions of iRAP (1), an optimal solution can be quickly obtained by using a modification of the classical min-heap based greedy method. Besides, we find that the relaxation solution is very close to the optimal solutions. In addition, we notice that the relaxation solution can be exponentially obtained in a centralized way.

Based on these ideas, we present a min-heap and iRAP relaxation based algorithm to solve iRAP (1), which is divided into two processes. The first process is to obtain the relaxation solution. The second process is to obtain an optimal solution with an approximate relaxation solution as its initial state.

To formulate these two processes, we provide some useful definitions.

First, for each agent i, we define the incremental cost function.

$$J_i(x_i) := \frac{dF_i(x_i)}{dx_i}. \tag{2}$$

The optimal solution $\hat{x}_i^*$ satisfies to the iRAP relaxation:

$$\begin{cases} J_i(\hat{x}_i^*) = \lambda^* & \text{for } \underline{x}_i < \hat{x}_i^* < \overline{x}_i, \\ J_i(\hat{x}_i^*) \le \lambda^* & \text{for } x_i^* = \overline{x}_i, \\ J_i(\hat{x}_i^*) \ge \lambda^* & \text{for } x_i^* = \underline{x}_i, \end{cases} \tag{3}$$

where $\lambda^* \in \mathbb{R}$ is the optimal Lagrange multiplier, and $\hat{x}_i^* \in \mathbb{R}$ is the optimal solution [19]. From (3), we define a projection function for λ to show the relation between the Lagrange multiplier and decision variables

$$\phi_i(\lambda) = \begin{cases} \overline{x}_i & \text{if } \lambda > J_i(\overline{x}_i), \\ J_i^{-1}(\lambda) & \text{if } J_i(\underline{x}_i) \le \lambda \le J_i(\overline{x}_i), \\ \underline{x}_i & \text{if } \lambda < J_i(\underline{x}_i), \end{cases} \tag{4}$$

Next, we introduce the definition of *min-heap* [22] that is an important concept in the centralized method. A *min-heap* contains two heaps (or each element of min-heap contains two parameters): a heap δ that all elements in it are sorted from the smallest to the largest, and a corresponding heap δ^{ID} recording the *ID* of nodes. For example, for a heap $\delta = [\delta_1, \cdots, \delta_n, \delta_1 \le \cdots \le \delta_n]$, $\delta_i^{ID} = j$ means that δ_i is the parameter for node j. We can create min-heap by the typical heap sort method, for which the computational complexity is $\mathcal{O}(n \log n)$.

After introducing these definitions, we present the first algorithm (Algorithm 1) to solve the iRAP relaxation problem, i.e., to obtain the relaxation solution.

Algorithm 1 Modified bisection algorithm to the iRAP relaxation problem.

```
Initialization:
 1: Set a small constant $\varepsilon$.
 2: Create a min-heap $\delta^1$ to store $J_i(\underline{x}_i)$.
 3: Use the min-heap to find $p (2 \le p \le n)$ such that $\sum_{i=1}^{n} \phi_i(\delta_p^1) \ge D$ and $\sum_{i=1}^{n} \phi_i(\delta_{p-1}^1) < D$.
 4: if we can not find $p$
 5: then $\underline{w}_1 = \delta_n^1$, $\overline{w}_1 = inf$.
 6: else if $\sum_{i=1}^{n} \phi_i(\delta_p^1) = D$
 7: then $\underline{w}_1 = \overline{w}_1 = \delta_p^1$.
 8: else $\underline{w}_1 = \delta_{p-1}^1$, $\overline{w}_1 = \delta_p^1$.
 9: end if
10: Similarly, create a min-heap $\delta^2$ to store $J_i(\overline{x}_i)$ and obtain $\underline{w}_2$ and $\overline{w}_2$.
11: $\underline{\lambda} = \max\{\underline{w}_1, \underline{w}_2\}$, $\overline{\lambda} = \min\{\overline{w}_1, \overline{w}_2\}$.
Update:
 1: do
 2:   $\lambda^* = \frac{\underline{\lambda} + \overline{\lambda}}{2}$
 3:   $\tilde{x}_i^* = \phi_i(\lambda^*)$
 4:   $x_{sum} = \sum_{i=1}^{n} \tilde{x}_i^*$
 5:   if $x_{sum} < D$.
 6:   then $\underline{\lambda} = \lambda^*$.
 7:   else $\overline{\lambda} = \lambda^*$
 8:   end if
 9: while $|x_{sum} - D| > \varepsilon$
```

In Algorithm 1, we notice that $\tilde{x}_i^*$ always satisfies the optimal constraint (3). Besides, if $\varepsilon = 0$, $\tilde{x}_i^*$ satisfies the constraint (1c) as the iteration step tends to infinity. In our paper, in order to terminate the algorithm in finite steps and have a smaller computational complexity, we take ε equaling to a proper small constant, for example, $\varepsilon = 1$. This termination condition implies that $\tilde{x}_i^*$ is an approximate relaxation solution.

Next, we present the second algorithm (Algorithm 2) to solve iRAP (1) by using the round-down of the approximate relaxation solution as its initial state.

Algorithm 2 Min-heap Based Algorithm.

```
Initialization:
1: $x_i = \lfloor \tilde{x}_i^* \rfloor$ where $\lfloor \tilde{x}_i^* \rfloor$ is obtained by the modified bisection method (Algorithm 1). Update $x_{sum} = \sum_{i=1}^{n} x_i$.
2: Create a min-heap $\delta^1$ with $n$ elements to store $inf$ if $x_i \geq \bar{x}_i$, and to store $F_i(x_i+1) - F_i(x_i)$ otherwise.$\delta^{ID1}$ records $ID$.
3: Create a min-heap $\delta^2$ with $n$ elements to store $inf$ if $x_i \leq \underline{x}_i$, and to store $F_i(x_i-1) - F_i(x_i)$ otherwise.$\delta^{ID2}$ records $ID$.
Update:
step 1: adjust $x_{sum}$ to $D$.
1: while $x_{sum} \neq D$
2:   if $x_{sum} < D$
3:   then $i = \delta_1^{ID1}$.
4:         Update $x_i = x_i + 1$.
5:         Update (deletion and addition) min-heaps.
6:         $x_{sum} = x_{sum} + 1$.
7:   else if $x_{sum} > D$
8:   then Similarly, update $i = \delta_1^{ID2}$, $x_i = x_i - 1$ ,$x_{sum} = x_{sum} - 1$, and min-heaps.
9:   end if
10: end while
step 2: find one of the optimal solutions.
1: Initialization: $\underline{\delta} = \delta_1^1 + \delta_1^2$
2: while $\underline{\delta} < 0$
3:   $i = \delta_1^{ID1}$, $j = \delta_1^{ID2}$.
4:   Update $x_i = x_i + 1$ and $x_j = x_j - 1$.
5:   Update min-heaps.
6:   $\underline{\delta} = \delta_1^1 + \delta_1^2$.
7: end while
```

In Algorithm 2, we first create two min-heaps and use $\lfloor \tilde{x}_i^* \rfloor$ to initialize these min-heaps. Second, we adjust x_{sum} to D by using these min-heaps. At last, an optimal solution is obtained through the iteration of step 2. The following theorem shows the convergence of Algorithm 2.

Theorem 1. *Suppose Assumption 1 holds. Algorithm 2 leads to one of the globally optimal solutions to iRAP (1).*

The proof of Theorem 1 is given in Appendix.

Remark 1. The update process of min-heap uses two auxiliary heaps to record the ID information and includes deletion and addition operating, for which the computational complexity is $\mathcal{O}(\log n)$ [22].

Remark 2. Considering Algorithm 2, we can obtain that the comparison $x_{sum} > D$ is not needed in the ideal situation (i.e., $\tilde{x}_i^* = \hat{x}_i^*$). However, the bisection method in Algorithm 1 is ended in finite steps. Thus, the initial state x may have a small offset from the ideal one. We need to add the comparison $x_{sum} > D$ in Algorithm 2.

Remark 3. The relaxation solution is the unique optimal solution of the iRAP relaxation problem. However, there may exist multiple solutions for iRAP (1). Algorithm 2 leads to one of the globally optimal solutions depending on the initial state.

3.2. Computational complexity analysis

In this section, we analyze the computational complexity of Algorithm 2. To clearly describe this, we need some notations. Denote $x = [x_1, \ldots, x_n]$. Without special declaration, $x^* = [x_1^*, \ldots, x_n^*]$ is a globally optimal solution obtained by Algorithm 2 in this section.

To analyze the computational complexity, we need to show the effect of the initial state on the convergence rate, i.e., to know the numbers of iteration of the two steps in Algorithm 2. In the following two lemmas, we provide an analytical result for the changing amount of each iteration, and the relation between the relaxation solution and the optimal solutions.

Lemma 1. *The 1-norm $\|x^* - x\|_1$ decreases by 1 for each iteration of the first step and decreases by 2 for each iteration of the second step of Algorithm 2.*

Lemma 2. *Denote S as the set containing all possible globally optimal solutions to iRAP (1). For any $x^* = [x_1^*, \ldots, x_n^*] \in S$, it satisfies the property that $(\forall i \in \mathcal{V}) x_i^* \geq \lfloor \hat{x}_i^* \rfloor$ or $(\forall i \in \mathcal{V}) x_i^* \leq \lceil \hat{x}_i^* \rceil$.*

The proof of Lemmas 1 and 2 are given in Appendix.

Based on Lemmas 1 and 2, we present the numbers of iteration needed by Algorithm 2 in the following theorem.

Theorem 2. *If $\tilde{x}_i^* = \hat{x}_i^*$, Algorithm 2 leads to a globally optimal solution by iterating the first step $D - \sum_{i=1}^{n} \lfloor \hat{x}_i^* \rfloor$ times and the second step less than or equal to $\sum_{i=1}^{n} \lceil \hat{x}_i^* \rceil - D$ times.*

Proof: From the termination condition $x_{sum} \neq D$ of the first step, we obtain that the first step iterates $D - \sum_{i=1}^{n} \lfloor \hat{x}_i^* \rfloor$ times in Algorithm 2.

Next, we prove that the second step iterates less than or equal to $\sum_{i=1}^{n} \lceil \hat{x}_i^* \rceil - D$ times. Lemma 2 shows that there exist two cases. We consider the two cases respectively in the following.

Case 1: $(\forall i \in \mathcal{V}) x_i^* \geq \lfloor \hat{x}_i^* \rfloor$.

From Lemma 1, $\|x^* - x\|_1$ decreases by 1 for each iteration of the first step. Together with the facts that $\|x^* - \lfloor \hat{x}_i^* \rfloor\|_1 = D - \sum_{i=1}^{n} \lfloor \hat{x}_i^* \rfloor$ and the first step iterates $D - \sum_{i=1}^{n} \lfloor \hat{x}_i^* \rfloor$ times in Algorithm 2, we find out that $x = [x_1, \ldots, x_n]$ obtained by the first step is x^*. Thus, the second step will not become active.

Case 2: $(\forall i \in \mathcal{V}) x_i^* \leq \lceil \hat{x}_i^* \rceil$.

Since $\|x^* - x\|_1$ decreases by 1 for each iteration of the first step, x_i obtained by the first step is $\lfloor \hat{x}_i^* \rfloor$ or $\lceil \hat{x}_i^* \rceil$. Without loss of generality, we assume x obtained by the first step is $[\lfloor \hat{x}_1^* \rfloor, \ldots, \lfloor \hat{x}_p^* \rfloor, \lceil \hat{x}_{p+1}^* \rceil, \ldots, \lceil \hat{x}_n^* \rceil]$, $p = \sum_{i=1}^{n} \lceil \hat{x}_i^* \rceil - D$, denoted as $x(k_0)$.

Together with $(\forall i \in \mathcal{V}) x_i^* \leq \lceil \hat{x}_i^* \rceil$, we obtain that $\|x^* - x(k_0)\|_1 \leq 2(\sum_{i=1}^{n} \lceil \hat{x}_i^* \rceil - D)$. Besides, we have shown in Lemma 1 that $\|x^* - x\|_1$ decreases by 2 for each iteration of the second step. Then, we obtain that the second step iterates less than or equal to $\frac{2(\sum_{i=1}^{n} \lceil \hat{x}_i^* \rceil - D)}{2} = \sum_{i=1}^{n} \lceil \hat{x}_i^* \rceil - D$ times. □

Computational complexity analysis: In this part, we analyze the computational complexity of the min-heap and iRAP relaxation based algorithm and compare it to a recent work. To analyze this, we consider the case that $\underline{x}_i = 0$ and $\bar{x}_i = D$ in [12]. Since Algorithm 2 uses $\tilde{x}_i^*$ as its initial state, we need to analyze the computational complexity of Algorithm 1 at first. Considering the initialization part of Algorithm 1, since searching for a min-heap needs a computational complexity of $\mathcal{O}(\log n)$ and each step of the search runs in $\mathcal{O}(n)$, the computational complexity of the initialization part of Algorithm 1 is $\mathcal{O}(n \log n)$. The update part of Algorithm 1 runs a computational complexity of $\mathcal{O}(n \log nD)$ if $\varepsilon = 1$, and the coefficient of nD is less than or equal to $\max_{i \in \mathcal{V}} \frac{\bar{l}_i}{l_i}$ from the property of bisection, where $\bar{l}_i$ is the upper bound and l_i is the lower bound of the second derivative of $F_i(x_i)$ in the interval $[\underline{x}_i, \bar{x}_i]$. In practice, this coefficient is always less than nD. Thus, we first obtain the initial state of Algorithm 2 by the modified bisection method with a computational complexity of $\mathcal{O}(n \log D + n \log n)$.

Next, we analyze the computational complexity of Algorithm 2 with the initial state $\tilde{x}_i^*$. Since we choose $\varepsilon = 1$, the initial state $x_i = \lfloor \tilde{x}_i^* \rfloor$ is either $\lfloor \hat{x}_i^* \rfloor - 1$, $\lfloor \hat{x}_i^* \rfloor$ or $\lfloor \hat{x}_i^* \rfloor + 1$. While $|\lfloor \tilde{x}_i^* \rfloor - \lfloor \hat{x}_i^* \rfloor| \le 1$, we find the two steps iterate less than $2n$ times with the initial state $\lfloor \tilde{x}_i^* \rfloor$ according to Lemma 1. Both the first step and the second step need two min-heaps. Since the computational complexity of deletion or addition for a min-heap is $\mathcal{O}(\log n)$, the computational complexity of these two steps are $\mathcal{O}(n \log n)$.

Thus, the total computational complexity of the min-heap and iRAP relaxation based algorithm is $\mathcal{O}(n \log D + n \log n)$. If $n \ll D$, the computational complexity is $\mathcal{O}(n \log D)$, which is better than the computational complexity $\mathcal{O}(n \log D \log n)$ that is shown in [12].

Remark 4. In fact, if the cost functions are quadratic functions and $x_i \in [0, D]$, by noticing the fact that $\sum_{i=1}^n \phi_i(\lambda) = \alpha\lambda + \beta$ during the update of Algorithm 1, where α and β can be calculated by simple summations after initializing $\bar{\lambda}$ and $\underline{\lambda}$. Thus, the computational complexity of Algorithm 1 can be reduced to $\mathcal{O}(\log D + n \log n)$. The total computational complexity of the min-heap and iRAP relaxation based algorithm is $\mathcal{O}(\log D + n \log n)$.

4. Distributed algorithm

In this section, we provide a distributed algorithm to find an optimal solution of iRAP (1). By extending the centralized algorithm, we first design a consensus and iRAP relaxation based distributed algorithm. Second, the convergence rate of the proposed distributed algorithm is analyzed. At last, termination criteria are presented for practical implementation of the distributed algorithm.

4.1. Consensus and iRAP relaxation based distributed algorithm

In the centralized algorithm, a modified bisection algorithm is presented to provide a solution to the iRAP relaxation problem, which is then used as the initial state to solve the iRAP based on iterations of min-heaps. Notice that if both the solution to the iRAP relaxation and the min-heap based iteration can be found in a distributed manner, a distributed extension of the centralized algorithm to solve iRAP (1) can then be designed.

Thus, similar to the centralized algorithm, we present a consensus and iRAP relaxation based algorithm to solve iRAP (1) in a distributed way, which is divided into two processes. The first process is to obtain the relaxation solution in a distributed way. The second process is to design a distributed successive approximation algorithm to obtain an optimal solution of iRAP (1) with an approximate relaxation solution as its initial state.

To design a distributed algorithm for the iRAP relaxation problem, we assume that each node holds an estimate of λ, denoting as λ_i. The estimate of λ_i needs to achieve consensus by a distributed algorithm. Before we present the distributed algorithm, we define a projection function for λ_i to show the relation between the estimate of Lagrange multiplier and decision variables in the following.

$$\phi_i(\lambda_i) = \begin{cases} \bar{x}_i & \text{if } \lambda_i > J_i(\bar{x}_i), \\ J_i^{-1}(\lambda_i) & \text{if } J_i(\underline{x}_i) \le \lambda_i \le J_i(\bar{x}_i), \\ \underline{x}_i & \text{if } \lambda_i < J_i(\underline{x}_i). \end{cases} \quad (5)$$

We are now ready to present a nonnegative-surplus based algorithm (Algorithm 3) to solve the iRAP relaxation problem, which was developed in [21].

Algorithm 3 Nonnegative-surplus based distributed optimization algorithm.

Initialization:

(1) Choose $x_i(0) \in [\underline{x}_i, \bar{x}_i]$ and $s_i(0) \ge 0$ for all i such that $\sum_{i=1}^n (x_i(0) + s_i(0)) = D$;
(2) Choose $\lambda_i(0)$ such that $\lambda_i(0) = J_i(x_i(0))$.

Update:

$$\lambda_i(k+1) = \lambda_i(k) + \left[\sum_{j \in \mathcal{N}_i^+} a_i(\lambda_j(k) - \lambda_i(k)) \right]_- + \epsilon_i s_i(k), \quad (6a)$$

$$x_i(k+1) = \phi_i(\lambda_i(k+1)), \quad (6b)$$

$$s_i(k+1) = b_i s_i(k) + \sum_{j \in \mathcal{N}_i^+} b_j s_j(k) - (x_i(k+1) - x_i(k)), \quad (6c)$$

where the parameters in the algorithm are chosen as follows:$a_i = \frac{1}{d_i^+ + 1}$, $b_i = \frac{1}{d_i^- + 1}$ and$\epsilon_i = c_i b_i$ with $c_i \in (0, l_i)$.

The operator $[\cdot]_-$ for the item $\sum_{j \in \mathcal{N}_i^+(k)} a_i(k)(\lambda_j(k) - \lambda_i(k))$ in Algorithm 3 ensures non-negative s_i, so we call Algorithm 3 a *nonnegative-surplus based algorithm.*

Lemma 3 (Theorem 1 in [21])**.** *Suppose Assumptions 1 and 2 hold. Algorithm 3 leads to the unique globally optimal solution to the iRAP relaxation problem.*

Lemma 3 implies that we can obtain the relaxation solution by Algorithm 3 with infinite number of iterations. Similar to the centralized algorithm, we denote $\hat{x}^* = [\hat{x}_1^*, \ldots, \hat{x}_n^*]^\mathrm{T} \in \mathbb{R}^n$ as the relaxation solution and $\tilde{x}^* = [\tilde{x}_1^*, \ldots, \tilde{x}_n^*]^\mathrm{T} \in \mathbb{R}^n$ as an approximate relaxation solution obtained by Algorithm 3 with a proper termination criterion.

Next, we present a distributed successive approximation algorithm (Algorithm 4) to solve iRAP (1). In the initialization of Algorithm 4, we choose the initial state as the round down of the approximate relaxation solution obtained by Algorithm 3. Besides, in order to get $\frac{D}{n}$, we assume that D is known by one of the nodes. In addition, we assume that random parameters $r_1, \ldots, r_n$ satisfy $(\forall i \neq j) r_i \neq r_j$ such that there exists only a node updating its state for each iteration of the first step and there exist only two nodes updating their states for each iteration of the second step. In the first step, we adjust the sum of x_i to D, and $\sum_{i=1}^n x_i$ increase or decrease one for each iteration. In the second step, we check whether there exist $x_i > \underline{x}_i$ and $x_j < \bar{x}_j, i \neq j$ satisfying $F_i(x_i + 1) + F_j(x_j - 1) < F_i(x_i) + F_j(x_j)$ so that the update approaches to an optimal solution iteratively. The following theorem shows the convergence of Algorithm 4.

Theorem 3. *Suppose Assumptions 1 and 2 hold. Algorithm 4 leads to one of the globally optimal solutions to iRAP (1).*

Proof: The only difference between this theorem and Theorem 1 is that the updates in Algorithm 2 are based on min-heaps, and the updates in Algorithm 2 are based on distributed consensus algorithms. The main stream remains the same as the proof of Theorem 1. □

As Algorithm 4 is expected to be a fully distributed algorithm, the computation of a number of variables in Algorithm 4 such as x_{avg}, $\frac{D}{n}$, $\underline{\delta}$, $\underline{\omega}$, etc. should also be done in a distributed manner. We firstly present a nonnegative-surplus based average consensus algorithm over digraphs (Algorithm 5) to compute x_{avg} and $\frac{D}{n}$, and show its convergence in the lemma below.

Algorithm 4 Successive approximation algorithm.

```
Initialization:
1: Choose the initial state $x_i = \lfloor \tilde{x}_i^* \rfloor$.
2: Each node selects a random parameter $r_i$.
3: Each node gets $\frac{D}{n}$.
Update:
step 1: adjust $x_{sum} = \sum_{i=1}^n x_i$ to $D$
1: do
2:    Compute $x_{avg} = \frac{\sum_{i=1}^n x_i}{n}$.
3:    if $x_{avg} > \frac{D}{n}$
4:    then $\delta_i = inf$ for $x_i = \underline{x}_i$
          and $\delta_i = F_i(x_i - 1) - F_i(x_i)$ for $x_i > \underline{x}_i$;
5:       Compute $\underline{\delta} = \min_{i\in\mathcal{V}} \delta_i$;
6:       $w_i = inf$ for $\delta_i \neq \underline{\delta}$ and $w_i = r_i$ otherwise;
7:       Compute $\underline{w} = \min_{i\in\mathcal{V}} w_i$;
8:       Update $x_i = x_i - 1$ for $w_i = \underline{w}$.
9:    else if $x_{avg} < \frac{D}{n}$
10:   then $\delta_i = inf$ for $x_i = \underline{x}_i$ and $\delta_i = F_i(x_i + 1) - F_i(x_i)$ for $x_i < \bar{x}_i$; Similarly, compute $\underline{\delta}$, set $w_i$, compute $\underline{w}$ and update $x_i = x_i + 1$ for $w_i = \underline{w}$.
11:   end if
12: while $x_{avg} \neq \frac{D}{n}$
step 2: find an optimal solution.
1: do
2:    $\delta_i^1 = inf$ for $x_i = \underline{x}_i$, $\delta_i^1 = F_i(x_i - 1) - F_i(x_i)$ for $x_i > \underline{x}_i$; Compute $\underline{\delta}^1 = \min_{i\in\mathcal{V}} \delta_i^1$;
3:    Similarly, compute $\underline{\delta}^2 = \min_{i\in\mathcal{V}} \delta_i^2 = \min_{i\in\{i|x_i<\bar{x}_i\}} F_i(x_i + 1) - F_i(x_i)$;
4:    if $\underline{\delta}^1 + \underline{\delta}^2 < 0$
5:       $w_i^1 = inf$ for $\delta_i^1 \neq \underline{\delta}^1$ and $w_i^1 = r_i$ otherwise;
6:       Compute $\underline{w}^1 = \min_{i\in\mathcal{V}} w_i^1$;
7:       $w_i^2 = inf$ for $\delta_i^2 \neq \underline{\delta}^2$ and $w_i^2 = r_i$ otherwise;
8:       Compute $\underline{w}^2 = \min_{i\in\mathcal{V}} w_i^2$;
9:       Update $x_i = x_i - 1$ for $w_i^1 = \underline{w}^1$;
10:      Update $x_j = x_j + 1$ for $w_j^2 = \underline{w}^2$.
11:   end if
12: while $\underline{\delta}^1 + \underline{\delta}^2 < 0$
```

Algorithm 5 Nonnegative surplus based average consensus algorithm.

Initialization: For any initial value $y_i(0)$, choose $s_i(0) = 0$.
Update:

$$y_i(k+1) = y_i(k) + \left[\sum_{j\in\mathcal{N}_i^+} a_i(y_j(k) - y_i(k))\right]_- + \epsilon_i s_i(k), \tag{7a}$$

$$s_i(k+1) = b_i s_i(k) + \sum_{j\in\mathcal{N}_i^+} b_j s_j(k) - (y_i(k+1) - y_i(k)), \tag{7b}$$

where the parameters in the algorithm are chosen as follows:$a_i = \frac{1}{d_i^+ + 1}$, $b_i = \frac{1}{d_i^- + 1}$ and$\epsilon_i = \hat{c}_i b_i$ with $\hat{c}_i \in (0,1)$.

Lemma 4 (Theorem 1 in [23]). *Suppose Assumption 2 holds. Algorithm 5 leads to average consensus of the parameters y_i.*

Secondly, we present a minimum/maximum consensus algorithm over digraphs (Algorithm 6) to compute $\underline{\delta}, \underline{w}, \underline{\delta}^1, \underline{\delta}^2, \underline{w}^1$, and $\underline{w}^2$. The minimum/maximum state is broadcast to all nodes by iterating equal to or less than n times when the graph is strongly connected [24].

Algorithm 6 Minimum and maximum consensus algorithm.

Initialization: $z_i(0)$.
Minimum Update: $z_i(k+1) = \min_{j\in\{i\}\bigcup N_i^+} z_j(k)$
Maximum Update: $z_i(k+1) = \max_{j\in\{i\}\bigcup N_i^+} z_j(k)$

Remark 5. It is worth to point out that Algorithm 5 is a special case of Algorithm 3. The average consensus is the optimal solution to the problem of optimizing the sum of the cost functions $\frac{x_i^2}{2}$ without state constraints. For such special case, $\lambda_i = x_i$.

Remark 6. There are also other methods to achieve average consensus (e.g., the push-sum algorithm in [25]) and minimum/maximum consensus (e.g., max-min consensus algorithms in [26].)

4.2. Convergence rate

In this section, we show how the initial state in Algorithm 4 affects the convergence rate. In other words, we would like to figure out the numbers of iteration of the two steps in Algorithm 4 with the initial state $x_i = \lfloor \hat{x}_i^* \rfloor$. After that, we discuss the computational complexity of the consensus and iRAP relaxation based distributed algorithm.

We first present a lemma to show the changing amount after each iteration of Algorithm 4, which is similar to Lemma 1.

Lemma 5. *Denote $x^* = [x_1^*, \ldots, x_n^*]$ as the globally optimal solution obtained by Algorithm 4. The 1-norm $\|x^* - x\|_1$ decreases by 1 for each iteration of the first step and decreases by 2 for each iteration of the second step of Algorithm 4.*

Proof: The proof of goes the same as the one of Lemma 1. □

Lemma 5 shows that Algorithm 4 is a successive approximation algorithm because it provides positive reinforcement for state changes that are successive steps towards an optimal solution. Together with the relation between the relaxation solution and the optimal solutions in Lemma 2, we show the numbers of iteration needed by Algorithm 4 in the following theorem.

Theorem 4. *If the initial state is $x_i = \lfloor \hat{x}_i^* \rfloor$, Algorithm 4 leads to a globally optimal solution by iterating the first step $D - \sum_{i=1}^n \lfloor \hat{x}_i^* \rfloor$ times and iterating the second step less than or equal to $\sum_{i=1}^n \lceil \hat{x}_i^* \rceil - D$ times.*

Proof: Since Lemmas 2 and 5 hold for the distributed algorithm, it can be proved in the same way as the proof of Theorem 2. □

At last, we discuss the computational complexity of the distributed algorithm. Suppose that the network traffics of Algorithms 3 and 5 with certain termination criteria are $\mathcal{O}(g(n, D))$ and $\mathcal{O}(g(n))$, respectively. Since Algorithm 6 is a finite step consensus and iterates less than or equal to n times, the network traffic of this algorithm is $\mathcal{O}(n\sum_{i=1}^n d_i^+)$. The average computation complexity for each node by the consensus and iRAP relaxation based algorithm is $\mathcal{O}(\frac{g(n,D)}{n} + g(n) + n\sum_{i=1}^n d_i^+)$, which depends on the convergence speed of Algorithm 3 and 5 and the topology structure.

Remark 7. From Lemma 5, we obtain that the convergence rate is closely related to the initial state. Lemma 2 shows that the globally optimal solutions are very close to $\hat{x}^* = [\hat{x}_1^*, \ldots, \hat{x}_n^*]$. Thus, it is better to use an approximate relaxation solution obtained by Algorithm 3 as the initial state than a random one.

Remark 8. Lemma 2 also shows that the optimal solutions will not diverge to infinity if we remove one side of the state constraints or both sides of the state constraints. Together with the changing amount of each iteration shown in Lemma 5, we can still obtain that Algorithm 4 terminates in finite steps. Our proof of Algorithm 4 is still applicable. Thus, our algorithm is suitable for the case without state constraints.

4.3. Termination criteria for distributed algorithms

Since Algorithms 5 and 6 are used in Algorithm 4 and the initial state $x_i = \lfloor \tilde{x}_i^* \rfloor$ is obtained by Algorithm 3, to obtain an optimal solution for iRAP (1) in finite times, we need to design termination criteria for Algorithms 3, 5, and 6. In the following, we provide proper termination criteria for practical implementation of these algorithms, and show that an optimal solution for iRAP (1) can still be obtained with these termination criteria. Before showing these termination criteria, we assume each node has an estimate of (or knows) n.

First, we give a termination criterion for Algorithm 6. Since Algorithm 6 is a finite step consensus algorithm and iterates less than or equal to n times, the termination criterion is to terminate Algorithm 6 in n times of iteration.

Next, we design a termination criterion for Algorithm 3. Add auxiliary variables $\varepsilon > 0$ and κ_i for each nodes. We define

$$\kappa_i(k) = \begin{cases} 1 & \text{if } \dfrac{\sum_{j\in\{i\}\bigcup N_i^+} |\lambda_j(k) - \lambda_i(k)| + |s_j(k)|}{1+d_i^+} < \dfrac{\varepsilon}{n}, \\ 0 & \text{otherwise} . \end{cases} \quad (8)$$

Compute $\min_{i\in\mathcal{V}} \kappa_i(k)$ by the minimum consensus algorithm. If the minimum consensus result is 1 (i.e., $(\forall i \in \mathcal{V})\kappa_i(k) = 1$), we terminate Algorithm 3. Unlike the termination criterion for Algorithm 6, the termination criterion for Algorithm 3 may cause small errors from the ideal consensus value. Since Algorithm 5 is a special case of Algorithm 3, we use the same termination criterion for Algorithm 5 as Algorithm 3.

At last, we show that a globally optimal solution can still be achieved under these termination criteria, i.e., the errors caused by termination criteria do not affect the convergence of Algorithm 4, but only affect the initial state of Algorithm 4, which is supported by the following argument. . In Algorithm 4, x_{avg} and $\frac{D}{n}$ are calculated by Algorithm 5, which are used for comparison. Notice that D and $\sum_{i=1}^{n} x_i$ are integers, for which the tolerate error is 1, which implies that the tolerate error between x_{avg} and $\frac{D}{n}$ is $\frac{1}{n}$. Thus, we take $\varepsilon \leq 0.5$, which is enough for us to do a comparison between x_{avg} and $\frac{D}{n}$. In total, by choosing suitable auxiliary variables for termination criteria, a globally optimal solution can still be achieved by Algorithm 4.

5. Simulation examples

In this section, we provide simulation examples to illustrate our proposed algorithms.

5.1. Computational complexity comparison

In this subsection, we test the efficiency of our proposed centralized algorithm by reporting its running times. The performance metrics include the running time (T), the time reduction (T_{reduce}) by comparing with the best known multi-phase algorithm (Algorithm 2) in [12]). Denote the running time for our algorithm as T, the running time for the algorithm in [12]) as T_{old}. All algorithms are implemented in Matlab R2010b on Intel Core i7-4710MQ dual-core 2.50GHz PC. The timing functions are *tic* and *toc*.

The cost function associated with each agent i takes the quadratic form $F_i(x_i) = a_i x_i^2 + b_i x_i + c_i$, where a_i, b_i, and c_i are randomly selected from (0, 1) for each agent i. We simulate for $D \in [10^3 10^4 10^5 10^6]^\mathsf{T}$ and $n \in [10203050]^\mathsf{T}$. For each fixed pair (n, D), we carry out 50 experiments with random quadratic functions, and calculate the average running times for comparison. The simulation results are recorded in Table 1.

Since the cost functions are quadratic functions, the computational complexity of Algorithm 2 is $\mathcal{O}(\log D + n \log n)$. The influence of D is very small, as shown in Table 1. Table 1 also shows the larger D is, the more reduction will be increased by our algorithm.

Table 1
Performance of Algorithm 2 and multi-phase algorithm in [12].

		n			
D		10	20	30	50
10^3	T_{old}	0.0025	0.0050	0.0078	0.0116
	T	0.0004	0.00085	0.0012	0.0022
	T_{reduce}	84%	83%	84.62%	82%
10^4	T_{old}	0.0040	0.0095	0.0144	0.0241
	T	0.00041	0.00088	0.0013	0.0023
	T_{reduce}	89.75%	90.74%	90.97%	90.46%
10^5	T_{old}	0.0053	0.0123	0.0192	0.0348
	T	0.00042	0.00089	0.0014	0.0023
	T_{reduce}	92.08%	92.76%	92.71%	93.4%
10^6	T_{old}	0.0065	0.0159	0.0246	0.0489
	T	0.00044	0.0009	0.0015	0.0023
	T_{reduce}	93.23%	94.34%	93.9%	95.3%

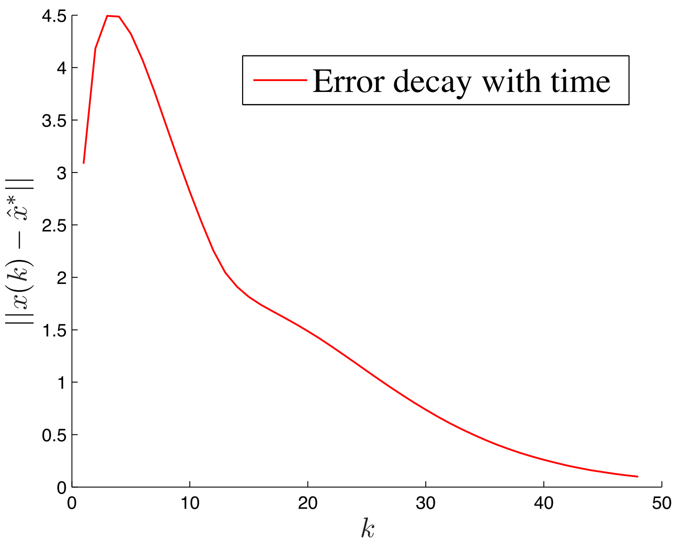

Fig. 1. The numbers of iteration (y-axis) until $\|x(k) - x^*\| < 0.1$.

5.2. A three-agents ring network example

In this subsection, we consider a three-nodes example with the information flow $1 \to 2 \to 3 \to 1$ to demonstrate Algorithm 4. The cost functions are taken from the simple example in [12], which are $F_1(x_1) = \frac{x_1(x_1+1)}{2}$, $F_2(x_2) = x_2(x_2+1) + 0.1$, and $F_3(x_3) = \frac{3x_3(x_3+1)}{2} + 0.2$. The total resource is $D = 12$. The state constraints are $0 \leq x_i \leq D$. In Fig. 1, we set the termination criterion as $\|x(k) - x^*\| < 0.1$, which terminates in 50 steps. The estimation of x^* obtained from Algorithm 3 is $[6.7643.2421.94]^\mathsf{T}$. Then we set the initial state of Algorithm 4 as $[631]^\mathsf{T}$. The updates of x_i's are recorded in Fig. 2. In this example, the first step of Algorithm 4 iterates two times for the first step, and the second step iterates zero time. The obtained optimal solution satisfies that $(\forall i \in \mathcal{V}) x_i^* \geq \lfloor \hat{x}_i^* \rfloor$. By comparing with the result of exhaustive testing, we find out that the result obtained by our Algorithm is an optimal solution.

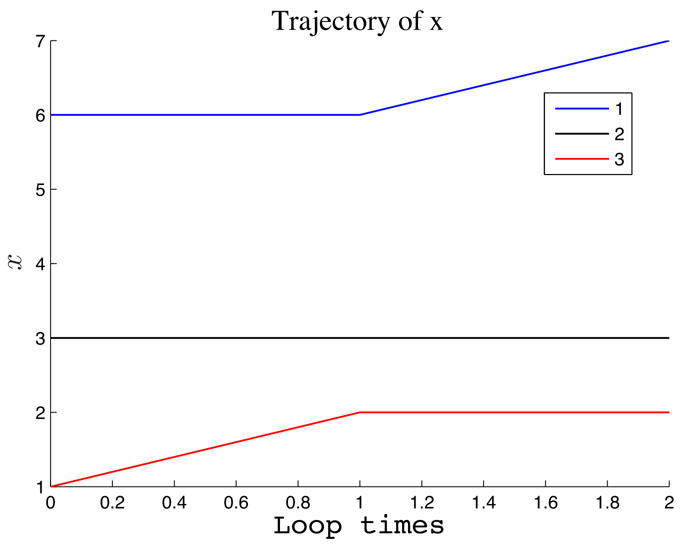

Fig. 2. Trajectory of x by Algorithm 4.

6. Conclusions

The resource allocation problem is a classical fundamental problem in optimization theory and widely applied in practice. In this paper, we first propose a novel min-heap and iRAP relaxation based centralized algorithm and prove that it has a computational complexity of $\mathcal{O}(n\log n + n\log D)$ when the resource constraints are $0 \le x_i \le D$, which outperforms the best known multi-phase algorithm running with a $\mathcal{O}(n\log n \log D)$ computational complexity. By extending the main idea of the centralized algorithm, a consensus based distributed algorithm is developed for a very general network setup over strongly connected digraphs. Moreover, termination criteria are also provided for practical implementation of the distributed algorithm.

Future works may include exploring inherent mechanisms for global convergence of distributed algorithms with a faster convergence rate. Besides, Assumption 1 in this paper indicates that the cost functions are strictly convex. Relaxation of this assumption is a further step towards general non-convex optimization problems. Moreover, the iRAP relaxation based algorithm may not be suitable for more general nonlinear/linear programming problems. Thus, further research may focus on optimization problems with integer constraints by applying the idea of the iRAP relaxation based algorithms developed in this paper.

Appendix

Proof of Theorem 1: In Algorithm 2, each iteration of the second step updates $x = [x_1, \dots, x_n]^\top$, and reduces $\sum_{i=1}^n F_i(x_i)$. Thus, the most recent x does not equal to any historical x. Since the variables are bounded by the state constraints and integer constraints, the second step will terminate in finite times.

Thus, the remaining issue is to argue whether the solution obtained by Algorithm 2 is one of the globally optimal solutions. Denote the solution obtained by Algorithm 2 as $x^* = [x_1^*, \dots, x_n^*]$. Suppose on the contrary there exists $x' = [x_1', \cdots, x_n']$ satisfying (1b)(1c)(1d) and $\sum_{i=1}^n F_i(x_i') < \sum_{i=1}^n F_i(x_i^*)$.

Without loss of generality, we assume that $x_i' > x_i^*$ for $i \in \{1, \cdots, p\}, p < n$, and $x_i' < x_i^*$ for $i \in \{p+1, \dots, q\}, q \le n$, and assume that the remaining nodes hold $x_i' = x_i^*$. From the equality constraint (1c), we obtain $\sum_{i=1}^q (x_i' - x_i^*) = 0$. Thus, $\sum_{i=1}^n F_i(x_i') - \sum_{i=1}^n F_i(x_i^*)$ can be rewritten as

$$\begin{aligned}
&\sum_{i=1}^n F_i(x_i') - \sum_{i=1}^n F_i(x_i^*)\\
&= \sum_{i=1}^p F_i(x_i') - F_i(x_i'-1) + \cdots + F_i(x_i^*+1) - F_i(x_i^*)\\
&\quad + \sum_{i=p+1}^q F_i(x_i') - F_i(x_i'+1) + \cdots + F_i(x_i^*-1) - F_i(x_i^*)\\
&= F_1(x_1') - F_1(x_1'-1) + F_{p+1}(x_{p+1}') - F_{p+1}(x_{p+1}'+1)\\
&\quad + \cdots\\
&\quad + F_i(x_i+1) - F_i(x_i) + F_j(x_j-1) - F_j(x_j)\\
&\quad + \cdots\\
&\quad + F_p(x_p^*+1) - F_1(x_p^*) + F_q(x_q^*-1) - F_q(x_q^*)
\end{aligned}$$

where $i \in \{1, \dots, p\}$, $j \in \{p+1, \dots, q\}$, $x_i^* \le x_i < x_i'$ and $x_j' < x_j \le x_j^*$.

Since $\sum_{i=1}^n F_i(x_i') < \sum_{i=1}^n F_i(x_i^*)$, there exist $i \in \{1, \dots, p\}$ and $j \in \{p+1, \dots, q\}$ such that

$$F_i(x_i+1) - F_i(x_i) + F_j(x_j-1) - F_j(x_j) < 0.$$

From the update of Algorithm 2, we know

$$F_i(x_i^*) + F_j(x_j^*) \le F_i(x_i^*+1) + F_j(x_j^*-1).$$

Then we obtain

$$\begin{aligned}
&F_i(x_i^*+1) - F_i(x_i^*) > F_i(x_i+1) - F_i(x_i)\\
\text{or }&F_j(x_j^*-1) - F_j(x_j^*) > F_j(x_j-1) - F_j(x_j).
\end{aligned}$$

A contradiction to that all $F_i(x_i)$ are strictly convex functions is reached. Thus, $x^* = [x_1^*, \dots, x_n^*]$ is one of the globally optimal solutions. □

Proof of Lemma 1: Since there are two steps in Algorithm 2, the proof of this lemma can be divided into two parts. Denote $x(k) = [x_1(k), \dots, x_n(k)]$ as the k-th iteration of each step.

Considering the first step: Our goal is to prove $\|x^* - x(k+1)\|_1 - \|x^* - x(k)\|_1 = -1$. Since $|\|x^* - x(k+1)\|_1 - \|x^* - x(k)\|_1| = 1$, we suppose on the contrary $\|x^* - x(k+1)\|_1 - \|x^* - x(k)\|_1 = 1$ at some k. This means that there exists a i such that $|x_i^* - x_i(k+1)| - |x_i^* - x_i(k)| = 1$ and $|x_i(k+1) - x_i(k)| = 1$.

Without loss of generality, we assume $D - \sum_{i=1}^n x_i(k) > 0$.

Case 1: $x_i(k+1) > x_i^*$. Then $x_i(k+1) = x_i(k) + 1$. For this case, there exists a $j (j \neq i)$ such that $x_j(k+1) = x_j(k) < x_j^*$. Since x^* is a globally optimal solution, we obtain

$$F_i(x_i^*) + F_j(x_j^*) \le F_i(x_i^*+1) + F_j(x_j^*-1).$$

From the iteration of the first step in Algorithm 2, we know

$$F_i(x_i(k)+1) - F_i(x_i(k)) \le F_j(x_j(k)+1) - F_j(x_j(k)).$$

Then we obtain

$$\begin{aligned}
&F_i(x_i^*+1) - F_i(x_i^*) \ge F_i(x_i(k)+1) - F_i(x_i(k))\\
\text{or }&F_j(x_j^*-1) - F_j(x_j^*) \ge F_j(x_j(k)) - F_j(x_j(k)+1).
\end{aligned}$$

A contradiction to that all $F_i(x_i)$ are strictly convex functions is reached.

Case 2: $x_i(k+1) < x_i^*$.Then $x_i(k+1) = x_i(k) - 1$. A contradiction to the property $|D - \sum_{i=1}^n x_i(k)| - |D - \sum_{i=1}^n x_i(k+1)| = 1$ of the update rule is reached.

In conclusion, $\|x^* - x\|_1$ decreases by 1 for each iteration of the first step.

Considering the second step: Suppose on the contrary that $\|x^* - x\|_1$ dose not decrease by 2 for each iteration of the second step. This means that the following three cases may happen.

Case 1: There exist i and j $(i \neq j)$, such that $x_j(k) < x_j^*$, $x_i(k) > x_i^*$, $x_j(k+1) = x_j(k) - 1$, and $x_i(k+1) = x_i(k) + 1$.

From the iteration of the second step in Algorithm 2, we first obtain

$$F_i(x_i(k)+1)+F_j(x_j(k)-1)<F_j(x_j(k))+F_i(x_i(k)).$$

Together with

$$F_i(x_i^*)+F_j(x_j^*)\le F_i(x_i^*+1)+F_j(x_j^*-1),$$

we obtain

$$F_i(x_i^*+1)-F_i(x_i^*)>F_i(x_i(k)+1)-F_i(x_i(k))$$
$$\text{or } F_j(x_j^*-1)-F_j(x_j^*)>F_j(x_j(k)-1)-F_j(x_j(k)).$$

A contradiction to that all $F_i(x_i)$ are strictly convex functions is then reached.

Case 2: There exist i and i_0 ($i_0\neq i$), such that $x_{i_0}(k)>x_{i_0}^*$, $x_i(k)\ge x_i^*$, $x_{i_0}(k+1)=x_{i_0}(k)-1$, and $x_i(k+1)=x_i(k)+1$.

By the fact that $\sum_{i=1}^n x_i(k)=D$, there exists a j ($j\neq i$ and $j\neq i_0$) such that $x_j(k)<x_j^*$ and

$$F_j(x_i(k))-F_j(x_i(k)+1)<F_j(x_j(k)+1)-F_j(x_j(k)).$$

Together with

$$F_i(x_i^*)+F_j(x_j^*)\le F_i(x_i^*+1)+F_j(x_j^*-1),$$

we obtain

$$F_i(x_i^*+1)-F_i(x_i^*)>F_i(x_i(k)+1)-F_i(x_i(k))$$
$$\text{or } F_j(x_j^*-1)-F_j(x_j^*)>F_j(x_j(k))-F_j(x_j(k)+1).$$

A contradiction to that all $F_i(x_i)$ are strictly convex functions is then reached.

Case 3: There exist j and j_0 ($j_0\neq j$), such that $x_{j_0}(k)<x_{j_0}^*$, $x_j(k)\le x_j^*$, $x_{j_0}(k+1)=x_{j_0}(k)+1$, and $x_j(k+1)=x_j(k)-1$. With the similar proof as for case 2, the same conclusion follows.

In conclusion, $\|x^*-x\|_1$ decreases by 2 for each iteration of the second step. □

Proof of Lemma 2: Suppose on the contrary that there exists a globally optimal solution x^* not satisfying $(\forall i\in\mathcal{V})x_i^*\ge\lfloor\hat{x}_i^*\rfloor$ and $(\forall i\in\mathcal{V})x_i^*\le\lceil\hat{x}_i^*\rceil$. Without loss of generality, we assume $x_i^*<\lfloor\hat{x}_i^*\rfloor$ and $x_j^*>\lceil\hat{x}_j^*\rceil, i\neq j$.

Since x^* is a globally optimal solution, we obtain

$$F_i(x_i^*)+F_j(x_j^*)\le F_i(x_i^*+1)+F_j(x_j^*-1).$$

By the strictly convex property, we have

$$F_i(x_i^*)-F_i(x_i^*+1)\ge F_i(\hat{x}_i^*-1)-F_i(\hat{x}_i^*)$$
$$\text{and } F_i(x_j^*-1)-F_i(x_j^*)\le F_i(\hat{x}_j^*)-F_i(\hat{x}_j^*+1).$$

Then we obtain

$$F_i(\hat{x}_i^*-1)+F_i(\hat{x}_j^*+1)\le F_i(\hat{x}_i^*)+F_i(\hat{x}_j^*).$$

It contradicts to $\hat{x}^*=[\hat{x}_1^*,\dots,\hat{x}_n^*]^\top$ is the unique globally optimal solution to the RAP without integer constraints. Thus, $(\forall i\in\mathcal{V})x_i^*\ge\lfloor\hat{x}_i^*\rfloor$ or $(\forall i\in\mathcal{V})x_i^*\le\lceil\hat{x}_i^*\rceil$ is proved. □

References

[1] S. Yang, S. Tan, J. Xu, Consensus based approach for economic dispatch problem in a smart grid, IEEE Trans. Power Syst. 28 (4) (2013) 4416–4426.
[2] K.J. Morrisse, G.F. Solimini, U.A. Khan, Distributed control schemes for wind-farm power regulation, in: Proceedings of the North American Power Symposium (NAPS), Champaign, IL, USA, 2012, pp. 1–6.
[3] A.J. Wood, B.F. Wollenberg, Power Generation, Operation,and Control, Wiley, New York, USA, 2012.
[4] M. Khoujaa, S. Goyal, A review of the joint replenishment problem literature:1989–2005, Eur. J. Oper. Res. 186 (1) (2008) 1–16.
[5] H. Ohtera, S. Yamada, Optimal allocation and control problems for software-testing resources, IEEE Trans. Reliab. 32 (2) (1990) 171–176.
[6] D.S. Hochbaum, Complexity and algorithms for nonlinear optimization problems, Ann. Oper. Res. 153 (1) (2007) 257–296.
[7] M. Patriksson, A survey on the continuous nonlinear resource allocation problem, Eur. J. Oper. Res. 185 (1) (2008) 1–46.
[8] M. Calinescu, S. Bhulai, B. Schouten, Optimal resource allocation in survey designs, Eur. J. Oper. Res. 226 (1) (2013) 115–121.
[9] O. Gross, A class of discrete type minimization problems, Res. Memo. (1956) 1644.
[10] N. Katoh, T. Ibaraki, H. Mine, A polynomial time algorithm for the resource allocation problem with a convex objective function, J. Oper. Res. Soc. 30 (5) (1979) 449–455.
[11] Z. Galil, N. Megiddo, A fast selection algorithm and the problem of optimum distribution of effort, J. ACM 26 (1) (1979) 58–64.
[12] C. Shi, H. Zhang, C. Qin, A faster algorithm for the resource allocation problem with convex cost functions, J. Discret. Algorithms 34 (1) (2015) 137–146.
[13] L. Xiao, S. Boyd, Optimal scaling of a gradient method for distributed resource allocation, J. Optim. Theory Appl. 129 (3) (2006) 469–488.
[14] Z. Zhang, M. Chow, Convergence analysis of the incremental cost consensus algorithm under different communication network topologies in a smart grid, IEEE Trans. Power Syst. 27 (4) (2012) 1761–1768.
[15] S. Kar, G. Hug, Distributed robust economic dispatch in power systems: A consensus+ innovations approach, in: Proceedings of the Power and Energy Society General Meeting, San Diego, California, USA, 2012, pp. 1–8.
[16] V. Loia, A. Vaccaro, Decentralized economic dispatch in smart grids by self-organizing dynamic agents, IEEE Trans. Syst. Man Cybern. Syst. 44 (4) (2014) 397–408.
[17] H. Lim, V. Lam, M. Foo, Y. Zeng, An adaptive distributed resource allocation scheme for sensor networks, in: Proceedings of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks, Hong Kong, China, 2006, pp. 770–781.
[18] E. Kranakis, D. Krizanc, O. Morales, Maintaining Connectivity in Sensor Networks Using Directional Antennae, Springer Berlin, Heidelberg, 2010.
[19] H. Xing, Y. Mou, M. Fu, Z. Lin, Distributed bisection method for economic power dispatch in smart grid, IEEE Trans. Power Syst. 30 (6) (2015) 3024–3035.
[20] A.D. Dominguez-Garcia, S.T. Cady, C.N. Hadjicostis, Decentralized optimal dispatch of distributed energy resources, in: Proceedings of the 51st IEEE Conference on Decision and Control, Maui, Hawaii, USA, 2012, pp. 3688–3693.
[21] Y. Xu, T. Han, K. Cai, Z. Lin, A fully distributed approach to resource allocation problem under directed and switching topologies, in: Proceedings of the Asian Control Conference, Kota Kinabalu, Malaysia, 2015, pp. 1–6.
[22] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, Cambridge, MA, 2009.
[23] K. Cai, H. Ishii, Average consensus on arbitrary strongly connected digraphs with time-varying topologies, IEEE Trans. Autom. Control 59 (4) (2014) 1066–1071.
[24] J. Corts, Distributed algorithms for reaching consensus on general functions, Automatica 44 (3) (2008) 726–737.
[25] A. Nedic, A. Olshevsky, Distributed optimization over time-varying directed graphs, IEEE Trans. Autom. Control 60 (3) (2015) 601–615.
[26] G. Shi, W. Xia, K.H. Johansson, Convergence of max-min consensus algorithms, Automatic 62 (C) (2015) 11–17.

Yun Xu received the Bachelor degree in College of Biomedical Engineering & Instrument Science from Zhejiang University, China, in 2010, and the Master degree in College of Automation Engineering from University of Electronic Science and Technology of China. He is currently pursuing the Ph.D. degree in Control Theory and Control Engineering at the College of Electrical Engineering, Zhejiang University. His research interests include consensus and distributed optimization.

Gangfeng Yan received his Bachelor and Master degree in Control Theory and Control Engineering from Zhejiang University, China, in 1981 and 1984, respectively. He is currently a professor in the College of Electrical Engineering, Zhejiang University, China. His research interests include hybrid systems, neural networks, and cooperative control.

Kai Cai received the B. Eng. degree in Electrical Engineering from Zhejiang University in 2006, the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto in 2008, and the Ph.D. degree in Systems Science from Tokyo Institute of Technology in 2011. He is currently an Associate Professor in Osaka City University; preceding this position he was an Assistant Professor in the University of Tokyo 2013–2014, and a postdoctoral fellow in the University of Toronto 2011–2013. His research interests include distributed control of multi-agent systems, distributed control of discrete-event systems, and control architecture of complex networked systems.

Zhiyun Lin received his Bachelor degree in Electrical Engineering from Yanshan University, China, in 1998, Master degree in Electrical Engineering from Zhejiang University, China, in 2001, and Ph.D. degree in Electrical and Computer Engineering from the University of Toronto, Canada, 2005. He is currently a professor in the School of Automation, Hangzhou Dianzi University, China. Preceding to this position, he held a professor position at Zhejiang University, China, from 2007 to 2016, and a postdoctoral fellow position at the University of Toronto, Canada, from 2005 to 2007. His research interests focus on distributed control, estimation and optimization, cooperative control of multi-agent systems, hybrid control system theory, and robotics.