RAIRO-Oper. Res. 54 (2020) 1835–1861
https://doi.org/10.1051/ro/2019097

RAIRO Operations Research
www.rairo-ro.org

DECENTRALIZED DECOMPOSITION ALGORITHMS FOR PEER-TO-PEER LINEAR OPTIMIZATION

M. Asli Aydin* and Z. Caner Taşkin

Abstract. We propose Decentralized Benders Decomposition and Decentralized Dantzig–Wolfe Decomposition algorithms for large-scale block angular linear programming problems. Our methods allow multiple peer decision makers to cooperate with the aim of solving the problem without the need of a central coordination mechanism. Instead we achieve cooperation by partial information sharing across a strongly connected communication network. Our main goal is to design decentralized solution approaches for decision makers who are unwilling to disclose their local data, but want to solve the global problem collaboratively for mutual benefit. We prove that our proposed methods reach global optimality in a finite number of iterations. We confirm our theoretical results with computational experiments.

Mathematics Subject Classification. 90C05, 90C06.

Received August 29, 2017. Accepted October 15, 2019.

1. Introduction

The main motivation of this study is driven by some optimization problems that we have encountered in practical applications of supply chain planning. Consider a manufacturer and a supplier within a supply chain. The manufacturer purchases certain materials from the supplier and uses these materials in its manufacturing processes. The supplier has its own resources, possibly produces other products and has other customer demands. In a planning approach commonly used in the industry, the manufacturer first performs its planning to seek an optimal allocation of its capacity to satisfy its customer demands, and identifies materials that it needs to purchase from the supplier to execute this plan. These material requirements are then translated into purchase orders and are passed to the supplier. Next, the supplier runs its own planning process to ensure that its own capacity is used efficiently and its customer demands are satisfied on time with minimum cost. Linear programming-based optimization models are commonly used in planning processes such as sales and operations planning (S&OP) and aggregate production planning [17,22]. Thus, in many practically relevant cases planning problems of the manufacturer and supplier can be formulated as linear programming problems, which are solved by the individual entity in sequence. However, such a sequential approach has two main drawbacks: (i) the two plans may be inconsistent in case the supplier has limited capacity so that it cannot satisfy the manufacturer's demand on time, and (ii) the overall plan generated sequentially may not yield the same quality as a plan that would be generated if the manufacturer and the supplier cooperated to generate a plan simultaneously.

Keywords. Linear programming, decentralized coordination, peer-to-peer optimization, block angular structure.

Department of Industrial Engineering, Boğaziçi University, 34342 Bebek, Istanbul, Turkey.
*Corresponding author: aslim.aydin@boun.edu.tr

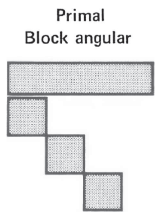

FIGURE 1. Primal block angular structure [4].

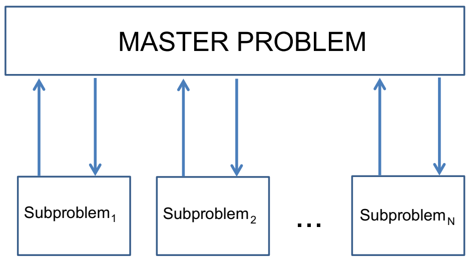

FIGURE 2. Structure and information exchange scheme for existing decomposition methods.

One possible approach that the manufacturer and the supplier can use to cooperate is to build a single optimization problem that combines their variables, constraints and objective functions into a single problem, and add constraints that ensure that the number of items shipped by the supplier is equal to the number of items received by the manufacturer. Then, the resulting optimization problem represents a primal block angular structure as shown in Figure 1.

Each block along the diagonal consists of local constraints of the manufacturer and the supplier associated with their own operations. The set of uppermost complicating constraints is associated with inventory transfer constraints plus any common capacity, material or budget constraints that link the interactions among the entities. This approach is only applicable if the manufacturer and the supplier are completely transparent about their data and optimization models, which might contain sensitive information such as profitability of products, capacities and planning objectives. Therefore, this approach can only be used in practice if the manufacturer and supplier belong to the same company.

A number of decomposition methods have been proposed in the literature to provide an efficient way for solving linear problems in block angular structure [3, 8, 18]. These methods are based on the observation that, without the complicating constraints, the overall problem can be partitioned into independent *subproblems* associated with each block. Thus, instead of solving one large-scale linear optimization problem, the problem is partitioned into several easier to solve subproblems. A solution set can be found by finding an independent solution for each subproblem. However, this rarely gives a solution for the overall problem because of the violation of complicating constraints. Hence a *master problem* is required to coordinate individual solutions of subproblems for obtaining a system-wide optimal solution. Thus, each entity sends a proposal to the center as a solution of its subproblem, then the center examines the proposals to obtain a global optimal solution by solving the master problem. Figure 2 represents the structure and information flow in existing decomposition methods.

Since some constraints and variables are maintained by the subproblems, decomposition methods reduce the amount of information sharing between entities. However, such methods still require a master problem to coordinate all subproblems. Thus, they are applicable in settings such as a multi-national firm consisting of separate companies in different countries, which are aligned *via* a central coordination unit belonging to the

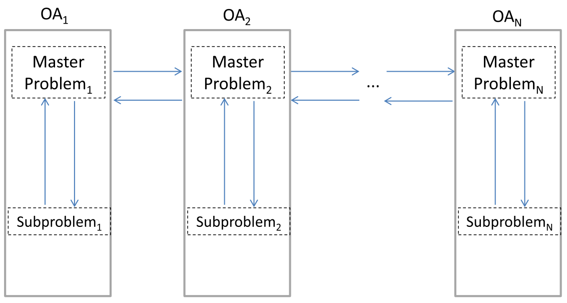

FIGURE 3. Structure and information exchange scheme for proposed methods.

headquarters. However, such approaches are not directly applicable in settings where the collaborating entities belong to independent companies since in this case it is not clear which entity would assume the role of the central coordination unit. Thus, there is a need for a decentralized decision making mechanism that allows multiple independent entities that are unwilling to share their private information but want to collaborate to solve the global problem by partial information exchange. Our main goal in this paper is to design optimization-based coordination mechanisms for linear programming problems in a peer-to-peer setting.

Decentralization approaches that exploit decomposition methods have attained significant attention recently. The authors in [11] address a capacity planning problem where finite capacity of a single facility is allocated among organizations to satisfy demand constraints. They propose Cooperative Interaction *via* Coupling Agents algorithm based on Lagrangian Relaxation where the facility and organizations act as coupling agents. A hybrid method especially for solving a cross-facility capacity allocation problem that combines Lagrangian relaxation and immunity-inspired coordination scheme is proposed in [14]. A distributed simplex method allowing more than two decision makers to solve linear programs is proposed in [9] that specifically addresses the security and access control issues arising in distributed data mining environments. Two-Stage Distributed Simplex Algorithm is proposed for block angular problems in a multi-agent setup in [5]. The algorithm solves the problem with information exchange only among the agents while utilizing the column generation method. In [1], the authors propose a decentralized coordination algorithm especially for sales and operations planning problems in supply chain environments. Decision makers may play one of the two specific roles: an Informed Party or one of the several Reporting Parties. The algorithm allows exchange of primal information among the parties.

In this paper, we propose two methods, *Decentralized Benders Decomposition* and *Decentralized Dantzig–Wolfe Decomposition* for solving block angular linear programs (BALP). We exploit the special structure of the problem to decompose it into a subproblem-local master problem pair for each decision maker, which we call *optimization agent* (OA). We solve the overall problem collaboratively by allowing minimum required information sharing among the OAs through peer-to peer communication. Figure 3 shows the structure and information exchange scheme for our methods.

We prove the convergence of the proposed methods to a global optimal solution in a finite number of iterations in case of the decision makers are tied to each other through a strongly connected communication network.

Main differences between our work and the existing literature can be summarized as follows. Different from [11], our methods allow more than two decision makers to solve their problem collaboratively. Only partial information sharing is required in both methods. The first one relies on sharing dual information while the second one requires primal information sharing. Thus, we offer two choice of methods to the users with respect to the type of information that they want to reveal. Our methods have no restriction on the number of complicating constraints in the global problem unlike [1]. Our methods do not require a distinct role definitions for decision makers as it is the case in [1, 9]. The decision makers in our approaches are equal on hierarchy and task assignment. Furthermore we prove the convergence of the proposed methods to the same optimal solution with centralized methods in finite number of iterations while [1, 11, 14, 19] state near-optimal solution.

The rest of this paper is organized as follows. In Section 2 we introduce the block angular problem that we address and we present notation for the communication network. In Section 3 we apply Classical Benders Decomposition as a solution approach before we present Decentralized Benders Decomposition method. In Section 4, we present Decentralized Dantzig–Wolfe Decomposition method. Section 5 presents experimental results on two groups of test instances with a discussion. Finally, Section 6 concludes the paper with possible future research directions. We provide correctness and convergence proofs of our proposed methods in appendices for brevity.

2. Problem statement and notation

2.1. Problem statement

The problem has N independent decision makers which we call optimization agents (OA). Each OA $i \in \{1, 2, \dots, N\}$ has its own set of decision variables x_{ij}'s for activity $j \in \{1, 2, \dots, n_i\}$. OAs aim to minimize their linear cost $\sum_{j=1}^{n_i} c_{ij}x_{ij}$, where $c_{ij} \in \Re$ and $x_{ij} \in \Re$, while satisfying a block of local constraints given by (2.1c). There is also a set of complicating constraints given by (2.1b) linking the interactions among the OAs. Hence, the resulting problem has primal block angular structure.

BALP

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{2.1a}$$

$$\text{subject to} \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq r^k \qquad \forall k = 1, 2, \dots, K \tag{2.1b}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \geq l_i \qquad \forall i = 1, 2, \dots, N \tag{2.1c}$$

$$x_{ij} \geq 0 \qquad \forall i = 1, 2, \dots, N \quad \forall j = 1, 2, \dots, n_i \tag{2.1d}$$

where a_{ij}^k, b_{ij}, r^k and l_i are all in $\Re$. Note that, the local constraint set given by (2.1c) may consist of more than one equation. Furthermore, the number of local constraints does not necessary to equal to each other for all OAs.

We may present an example of BALP as a supply chain problem that we address in Section 1. We use the following description:

Indices

$i = 1, 2, \dots, N$: Entities (either manufacturer or supplier) in the supply chain network.
$j = 1, 2, \dots, n_i$: Activities/operations associated with entity i.
$k = 1, 2, \dots, K$: Common resources shared by the entities.

Parameters

c_{ij} : unit cost for activity/operation j of entitiy i.
r^k : capacity of common resource k.
l_i : capacity of local resources of entity i.
a_{ij}^k : consumption rate of the common resource k for activity/operation j by entity i.
b_{ij} : consumption rate of local resources for activity/operation j by entity i.

Decision variable

x_{ij} : decision variable controlled by entity i for activity/operation j.

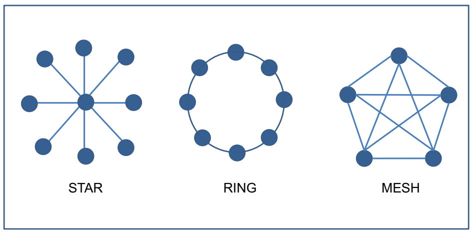

FIGURE 4. Some commonly used network topologies.

In this model, the system-wide objective is to minimize the total cost given by (2.1a). (2.1c) gives the blocks of local constraints of manufacturer and the supplier associated with their own operations. (2.1b) gives the complicating constraints associated with inventory transfer, common capacity, material or budget. Note that r^k values related to inventory transfer constraints are equal to zero to ensure that the number of items delivered by the supplier is equal to the number of items received by the manufacturer.

2.2. Communication network

One possible approach for solving (2.1) is *centralization* where a center builds the problem as a whole, announces the solution after solving it with Simplex Algorithm or Interior Point methods [6, 13]. However, in this work we propose decentralized decomposition methods to solve the problem without a central coordination unit. Thus, individual OAs need to communicate and exchange information with each other in order to solve the problem collaboratively.

We consider a directed graph, *G(N,A)*, to model the communication network among N decision makers. In this setting, OAs are the nodes of the graph and the arcs represent a communication link between two OAs. OA t is called a *neighbour* of OA i if there is an arc from OA i to OA t and $N_i \subseteq N$ denotes the set of neighbours of OA i. Note that communication between any two nodes may be uni-directional, *i.e.*, if OA i is a neighbour of OA t, it does not necessarily mean that OA t is a neighbour of OA i. In order for the problem to reach global optimality, any pair of the optimization agents should communicate to each other. OA i can communicate with OA t where $t \notin N_i$ if there exists a path from node i to node t. Hence, we assume a *strongly connected* graph for a communication network where there exists a path that goes from i to t for every pair of the nodes of the graph.

There are several common communication network topologies in the literature. Figure 4 illustrates the topologies which we consider in our work. In Star Topology, there is a node in the middle and all other nodes are directly connected to it. Connection among the nodes are achieved indirectly through the node in the middle. In Ring Topology, nodes are connected in a closed loop configuration. While adjacent pairs of nodes are directly connected, other nodes are connected indirectly through one or more intermediate nodes. In Mesh Topology, each node is directly connected to all other nodes. We assume that the communication between any two nodes is bi-directional.

3. DECENTRALIZED BENDERS DECOMPOSITION

Classical Benders Decomposition is equivalent to Dantzig–Wolfe decomposition applied to the dual of the linear program in Primal Block Angular structure [20]. Thus, it can be best applied to the linear programs in Dual Block Angular Structure [21]. Figure 5 shows dual block angular structure. Instead of complicating constraints, there is a number of complicating variables that connects independent blocks.

In this section, first we describe an application of Classical Benders Decomposition to primal BALP. Then, we propose Decentralized Benders Decomposition.

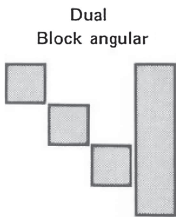

FIGURE 5. Dual block angular structure [4].

3.1. Classical Benders Decomposition Applied to Primal BALP

We propose a reformulation of (2.1) that enables decomposing the complicating constraints in (2.1b) into several subproblems. To this end, we introduce a new variable, r_i^k, denoting the share of OA i for k. Hence we add a new set of constraints (3.1b) satisfying the sum of the shares is equal to the total available amount for each k. Then, instead of each complicating constraint in (2.1b), the following set of constraints is introduced:

$$\textstyle\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq r_i^k \qquad \forall i.$$

Hence, an equivalent formulation for (2.1) after decomposing the complicating constraints can be given as the following:

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{3.1a}$$

$$\text{subject to:} \sum_{i=1}^{N} r_i^k = r^k \qquad \forall k = 1,2,\ldots,K \tag{3.1b}$$

$$\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq r_i^k \qquad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K \tag{3.1c}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \geq l_i \qquad \forall i = 1,2,\ldots,N \tag{3.1d}$$

$$r_i^k \quad \text{unrestricted} \quad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K \tag{3.1e}$$

$$x_{ij} \geq 0 \qquad \forall i = 1,2,\ldots,N \qquad \forall j = 1,2,\ldots,n_i. \tag{3.1f}$$

With this reformulation, (3.1) has a dual block angular structure, where r_i^k are treated as complicating variables. Furthermore, note that (3.1c) can be treated as local constraints in addition to (3.1d). Assume that the complicating variables are fixed to a given value, $\hat{r}_i^k$ while satisfying (3.1b). Then, the centralized problem in (3.1) can be solved as N independent problems only in x_{ij} variables. Given $\hat{r}_i^k$ values, the ith subproblem is formulated as the following:

$SP_i(\hat{r}_i^k)$

$$\text{Minimize} \sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{3.2a}$$

$$\text{subject to:} \sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq \hat{r}_i^k \qquad \forall k = 1,2,\ldots,K \tag{3.2b}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \geq l_i \tag{3.2c}$$

$$x_{ij} \geq 0 \qquad \forall j = 1, 2, \ldots, n_i. \tag{3.2d}$$

Note that (3.2) is a linear program for given $\hat{r}_i^k$ values . If any one of the subproblems is unbounded for $\hat{r}_i^k$, then (3.1) is also unbounded. This implies the unboundedness of the centralized problem in (2.1). Hence, we assume bounded subproblems. By introducing dual variables π_i^k associated with constraints (3.2b) and w_i associated with the constraint (3.2c), the dual subproblem can be formulated as the following:

$Dual - SP_i(\hat{r}_i^k)$

$$\text{Maximize } \sum_{k=1}^{m} \hat{r}_i^k \pi_i^k + l_i w_i \tag{3.3a}$$

$$\text{subject to: } \sum_{k=1}^{K} a_{ij}^k \pi_i^k + b_{ij} w_i \leq c_{ij} \qquad \forall j = 1, 2, \ldots, n_i \tag{3.3b}$$

$$\pi_i^k \geq 0 \qquad \forall k = 1, 2, \ldots, K \tag{3.3c}$$

$$w_i \geq 0. \tag{3.3d}$$

Note that only the objective function depends on the values of $\hat{r}_i^k$. If the feasible region in (3.3) is empty, then for any $\hat{r}_i^k$ either (3.2) is unbounded or infeasible. Hence, we assume that the feasible region of dual subproblem is non-empty. Thus, the extreme points of the feasible region in (3.3) can be enumerated as $\{\binom{\pi_i^k}{w_i}^p\}$, where p is an element of the set of extreme points, P_i. Similarly, its extreme rays can be enumerated as $\{\binom{\pi_i^k}{w_i}^r\}$, where r is an element of the set of extreme rays, R_i. If dual subproblem is bounded, then there exists an extreme point, $p \in P_i$ that maximizes the objective function value $\hat{r}_i^k(\pi_i^k)^p + l_i w_i^p$. Otherwise, if the the dual subproblem is unbounded, then there exists an extreme ray, $r \in R_i$ such that $\hat{r}_i^k(\pi_i^k)^r + l_i w_i^r > 0$. In this second case, (3.2) is infeasible. Thus, (3.3) can be reformulated as the following:

DSP$_i(\hat{r}_i^k)$

$$\text{Minimize } q_i \tag{3.4a}$$

$$\text{subject to: } \sum_{k=1}^{K} \hat{r}_i^k (\pi_i^k)^p + l_i w_i^p \leq q_i \qquad p \in P \tag{3.4b}$$

$$\sum_{k=1}^{K} \hat{r}_i^k (\pi_i^k)^r + l_i w_i^r \leq 0 \qquad r \in R \tag{3.4c}$$

$$q_i \quad \text{unrestricted.} \tag{3.4d}$$

Constraints of type (3.4b) are called *Benders optimality cuts*, while constraints of type (3.4c) are called *Benders feasibility cuts*. Then, (2.1) can be reformulated equivalently by using the Benders cuts:

MP

$$z_{MP} = \text{Minimize } \sum_{i=1}^{N} q_i \tag{3.5a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k = r^k \qquad \forall k = 1, 2, \ldots, K \tag{3.5b}$$

$$\sum_{k=1}^{K} r_i^k (\pi_i^k)^p + l_i w_i^p \leq q_i \qquad p \in P_i \qquad \forall i = 1, 2, \ldots, N \tag{3.5c}$$

$$\sum_{k=1}^{K} r_i^k (\pi_i^k)^r + l_i w_i^r \leq 0 \qquad r \in R_i \qquad \forall i = 1, 2, \ldots, N \tag{3.5d}$$

$$q_i \quad \text{unrestricted} \tag{3.5e}$$

$$r_i^k \quad \text{unrestricted.} \tag{3.5f}$$

The number of Benders cuts in (3.5) is generally huge. Hence, Classical Benders Decomposition starts with a Relaxed Master Problem (RMP) consisting of a subset of feasibility and optimality cuts. A center solves RMP and announces the $\hat{r}_i^k$ values. Each OA updates (3.3) with $\hat{r}_i^k$ and solves to optimality to produce either a feasibility or an optimality cut. Then, RMP is re-solved after the addition of the cuts. The algorithm terminates if no new cut is generated. The convergence of Classical Benders decomposition in a finite number of iterations follows since in each time a dual subproblem is solved, a unique Benders cut is generated associated with an extreme point or an extreme ray. Since the sets of extreme points and extreme rays are finite, there are finitely many feasibility or optimality cuts to be added. Efficiency of Classical Benders Decomposition is based on the observation that the algorithm typically reaches optimality after adding a fraction of possible Benders cuts.

3.2. Decentralized Benders Decomposition algorithm

Classical Benders Decomposition requires a center to solve RMP. By Decentralized Benders Decomposition method, our aim is to achieve complete removal of the center from the system. Thus, instead of solving a global master problem by a center, we introduce a local copy of the relaxed master problem for each OA.

MP_i

$$\text{Minimize} \quad q_i \tag{3.6a}$$

$$\text{subject to:} \quad \sum_{i=1}^{N} r_i^k = r_k \qquad \forall k = 1, 2, \ldots, K \tag{3.6b}$$

$$r_i^k \text{ unrestricted} \qquad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{3.6c}$$

$$q_i \geq \text{LB}_i. \tag{3.6d}$$

Note that initially (3.6) consists of linking constraints (3.1b), only. We also initialize a lower bound LB_i for q_i which denotes the objective function value of the ith subproblem to ensure feasibility of local master problem at initial iterations. To find an LB_i, we solve a problem consisting of local constraints given by (3.2c) only.

Algorithm 1: DECENTRALIZED BENDERS DECOMPOSITION $(Dual - SP_i(\hat{r}_i^k),\ MP_i,\ G = (N, A))$.

```
   Input: $Dual - SP_i(\hat{r}_i^k)$ { Dual subproblem for OA $i$}
   Input: $MP_i$ {Local master problem for OA $i$}
   Input: G =(N, A){Cut exchange network}
1  {$r_i^k$ denotes the share of $r^k$ for OA $i$}
2  {$V$ denotes the visited neighbours list for finding a new cut}
3  {$BC$ denotes the Benders cut}
4  for $i = 1$ to $N$ do
5      repeat
6          Solve $MP_i \rightarrow (r_i^k)$
7          $V \leftarrow \varnothing$
8          $BC \leftarrow GetCut(i, V, r_i^k)$
9      until $BC = Null$
```

Algorithm 1 describes Decentralized Benders Decomposition algorithm. It starts with a pair of problems $Dual - SP_i(\hat{r}_i^k) - MP_i$ for each OA and a strongly connected communication graph, $G = (N,A)$. Each OA solves its local master problem, MP_i, and gets the r_i^k values. Algorithm 1 utilizes GETCUT procedure, which

Algorithm 2: GETCUT(t, V, r_i^k).

```
Input: $t$ {Identity of OA}
Input: $V$ {Visited Neighbours List}
Input: $r_i^k \quad \forall k = 1, 2, \ldots, K$ {Share of $r^k$ for OA $i$}
Output: $BC$ {Benders Feasibility or Optimality cut}
1  if $t \in V$ then
2      return $Null$
3  $V \leftarrow V \cup \{t\}$
4  Update objective function of $Dual - SP_i(\hat{r}_i^k)$
5  Solve $Dual - SP_i(\hat{r}_i^k) \rightarrow GetStatus$
6  if $GetStatus = Optimal$ then
7      $BC \leftarrow$ Generate Benders optimality cut according to (3.4b)
8  else if $GetStatus = Unbounded$ then
9      $BC \leftarrow$ Generate Benders feasibility cut according to (3.4c)
10 else
11     $BC \leftarrow$ Null
12 if $BC = Null$ then
13     forall the $n \in N_t$ do
14         $BC \leftarrow$ GETCUT$(n, V, r_i^k)$
15         if $BC \neq Null$ then
16             Break out of the ForAll loop
17 Add $BC$ to $MP_t$
18 return $BC$
```

looks for a cut from neighbours recursively, if there is no cut generated by the OA itself. Hence, Algorithm 1 keeps track of the visited OAs with the list, V. It terminates when no new cut is generated for any of the OAs.

Algorithm 2 gives the details of the GETCUT procedure. It is a recursive procedure that returns a Benders cut, if it exists. First, the procedure adds t, the identity of the current OA, to the visited neighbours list, V, to avoid visiting it more than once. Then the objective function of $Dual - SP_i(\hat{r}_i^k)$ is updated with the given r_i^k values and it is solved. According to the solution status of $Dual - SP_i(\hat{r}_i^k)$, either a Benders feasibility cut or a Benders optimality cut is generated. Otherwise, Algorithm 2 looks for a new cut recursively from all neighbouring OAs until it finds a new cut. Here $N_t \subset N$ denotes the set of all neighbours of OA t. If a new cut is generated then it is added to the local master problem of OA t. Algorithm 2 runs until all OAs are visited. A flowchart of Algorithm 2 is shown in Figure 6. Note that, when a cut is generated by OA t with respect to OA i's allocations through GETCUT procedure, then all the OAs on the path connecting OAs i and t adds that cut to their local master problem by definition of GETCUT procedure.

Notice that the local master problem consists of only a subset of constraints initially. Hence at any iteration, the optimal objective function value of the local master problem is a lower bound on objective function value of (2.1). Also notice that the sum of objective function values of the subproblems is an upper bound for objective function value of (2.1). The advantage of this feature is allowing termination before reaching global optimality if lower and upper bounds are close enough.

Proposition 3.1. *Decentralized Benders Decomposition yields an optimal solution for BALP (if one exists) within a finite number of iterations if communication network is strongly connected.*

Proof. See Appendix A.1. □

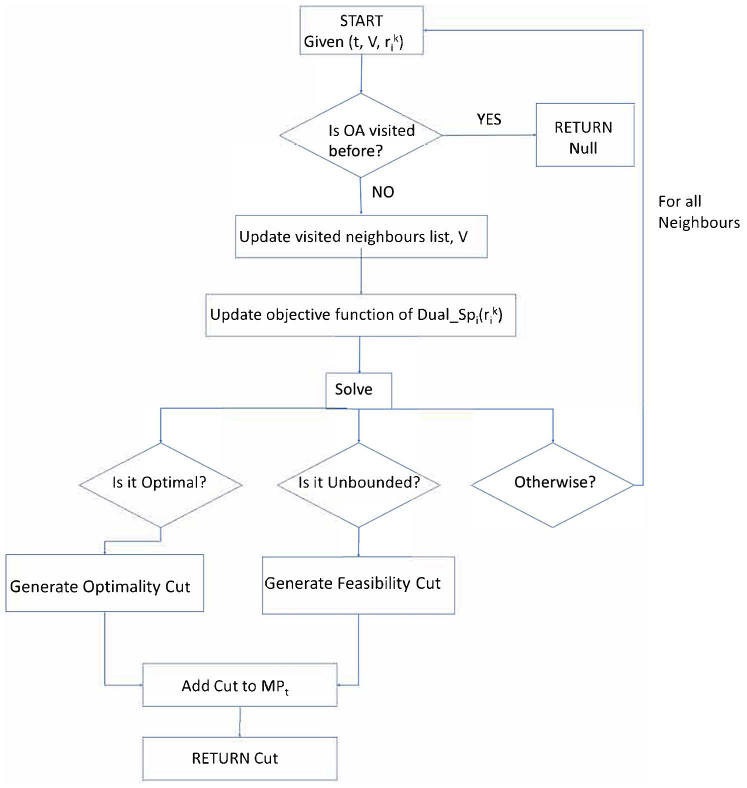

FIGURE 6. Flowchart of GETCUT procedure given in Algorithm 2.

4. DECENTRALIZED DANTZIG–WOLFE DECOMPOSITION

Dantzig–Wolfe Decomposition requires a center for coordinating the information exchange between master problem and subproblems. Hence it is not applicable for peer-to-peer optimization problems. In this section, we describe Decentralized Dantzig–Wolfe Decomposition algorithm that allows decision makers to solve BALP collaboratively without need of a center. Decentralized Dantzig–Wolfe Decomposition algorithm (Algorithm 3) utilizes Phase I algorithm as described in [12] to ensure the feasibility of local master problem. If an initial local master problem is established, Algorithm 3 calls Phase II algorithm (Algorithm 4). Otherwise, it terminates since BALP is infeasible.

Algorithm 3: DECENTRALIZED DANTZIG–WOLFE DECOMPOSITION.

```
1 $IsFeasible$ ←Phase I Algorithm
2 if $IsFeasible$ then
3     Phase II Algorithm
4 else
5     BALP is infeasible.
```

4.1. Decentralized Dantzig–Wolfe Decomposition algorithm

We consider block angular linear programs given by (2.1). Without the complicating constraints in (2.1b), the problem can be decomposed into N smaller subproblems each of which is associated with a block of local constraints in (2.1c). By Minkowski's Representation Theorem, any point x_{ij} in the feasible region of subproblem i can be expressed as sum of a convex combination of its extreme points and non-negative linear combination of its extreme rays. For the sake of simplicity, we assume a bounded feasible region for subproblems. Hence, we can formulate local master problem for OA i by using extreme points, $x_i^p \in P_i$, as the following:

MP_i

$$\text{Minimize} \sum_{j=1}^{n_i}\sum_{p} c_{ij}x_{ij}^p\lambda_i^p \tag{4.1a}$$

$$\text{subject to} \sum_{j=1}^{n_i}\sum_{p} a_{ij}^k x_{ij}^p\lambda_i^p \geq r^k \qquad \forall k=1,2,\ldots,K \tag{4.1b}$$

$$\sum_{p}\lambda_i^p = 1 \tag{4.1c}$$

$$\lambda_i^p \geq 0 \qquad \forall x_i^p \in P_i. \tag{4.1d}$$

Here P_i denotes the set of extreme points of the subproblem i's feasible region. To ensure feasibility of MP_i, we use Phase I method. Once we initialize local master problem MP_i, we solve it by column generation method. We decide whether a variable can be added to MP_i with its corresponding column with respect to its reduced cost:

$$\text{RC}: \quad \sum_{j=1}^{n_i}\left(c_{ij} - \sum_{k=1}^{K}\pi_k a_{ij}^k\right)x_{ij} - w_i, \tag{4.2}$$

where $\pi_k \quad \forall k = 1,2,\ldots,K$ and w_i are dual variables associated with complicating constraints and convexity constraint for (4.1), respectively. The most profitable variable to enter the master problem is the one having the most negative reduced cost. Hence, we solve the following pricing subproblem:

$SP_i(\pi_k, w_i)$

$$z_{SP_i} = \text{Minimize} \sum_{j=1}^{n_i}\left(c_{ij} - \sum_{k=1}^{K}\pi_k a_{ij}^k\right)x_{ij} - w_i \tag{4.3a}$$

$$\text{subject to} \sum_{j=1}^{n_i} b_{ij}x_{ij} \geq l_i \tag{4.3b}$$

$$x_{ij} \geq 0 \quad \forall j = 1,2,\ldots,n_i. \tag{4.3c}$$

Assume that an optimal solution of $SP_i(\pi_k, w_i)$ is x_{ij}^*. The column generated with respect to x_{ij}^* is as the following:

$$C: \begin{bmatrix} \sum_{j=1}^{n_i} c_{ij}x_{ij}^* & \sum_{j=1}^{n_i} a_{ij}^1 x_{ij}^* \cdots \sum_{j=1}^{n_i} a_{ij}^K x_{ij}^* & 1 \end{bmatrix}^T \tag{4.4}$$

where T is the transpose operator.

We describe Phase II algorithm in Algorithm 4. It starts with a pair of problems $SP_i(\pi_k, w_i) - MP_i$ for each OA $i \in \{1,2,\ldots,N\}$ and a strongly connected communication graph, $G = (N,A)$. Here (π_k, w_i) denotes the dual variables of MP_i, V denotes the visited neighbours list for finding a new column from the neighbours and C denotes the column generated. Algorithm 4 utilizes recursive GetColumn procedure to find a new column.

Algorithm 4: PHASE II ALGORITHM $(SP_i(\pi_k, w_i), MP_i, G = \{N, A\})$.

```
Input: $SP_i(\pi_k, w_i)$ { Pricing Subproblem for OA $i$}
Input: $MP_i$ {Local master problem for OA $i$}
Input: $G = \{N, A\}$ {Strongly connected digraph}
1  for $i = 1$ to $N$ do
2      repeat
3          Solve $MP_i \rightarrow (\pi_k, w_i)$
4          $V \leftarrow \varnothing$
5          $C \leftarrow GetColumn(i, V, (\pi_k, w_i))$
6      until $C = Null$
```

It terminates when no new column is generated for any of the OAs. Note that, since a center is lack in the system, each OA get the solution by its own after running the steps (5) and (6).

Algorithm 5 gives the details of recursive GETCOLUMN procedure. Here (π_k, w_i) denotes the dual variables of MP_i, V denotes the visited neighbours list for finding a new column from the neighbours and C denotes the column generated. First GETCOLUMN procedure updates the visited neighbours list. Then it updates $SP_t(\pi_k, w_i)$ with dual variables of OA i to solve subproblem of OA t where $i \neq t$. If reduced cost is negative for a variable, then a new column is generated. Otherwise, Algorithm 5 looks for a new column recursively from neighbours. Note that when a column is generated by OA t for OA i, then by definition of GETCOLUMN procedure, all the OAs on the path connecting the OAs add that column. Algorithm 5 terminates when all neighbours are visited (Fig. 7).

Algorithm 5: GETCOLUMN $(t, V, (\pi_k, w_i))$.

```
Input: $t$ {Identity of OA}
Input: $V$ {Visited Neighbours List}
Input: $(\pi_k, w_i)$ {Dual variables of $MP_i$ }
Output: $C$ {New Column}
1  if $t \in V$ then
2      return Null
3  $V \leftarrow V \cup \{t\}$
4  Update objective function of $SP_t(\pi_k, w_i)$
5  Solve $SP_t(\pi_k, w_i) \rightarrow z^*_{SP_t}$
6  RC $\leftarrow z^*_{SP_t}$ {RC denotes reduced cost in (4.2)}
7  if RC $< 0$ then
8      Generate C according to (4.4)
9  else
10     forall the $n \in N_t$ do
11         $C \leftarrow$ GETCOLUMN$(n, V, (\pi_k, w_i))$
12         if $C \neq Null$ then
13             Break out of the For loop
14 Add $C$ to $MP_t$
15 return $C$
```

Note that local master problem of an OA is initially formulated as (4.1) and it consists of a few columns. Then the local master problem grows gradually by the addition of new columns including the ones coming from the neighbours and becomes as the following:

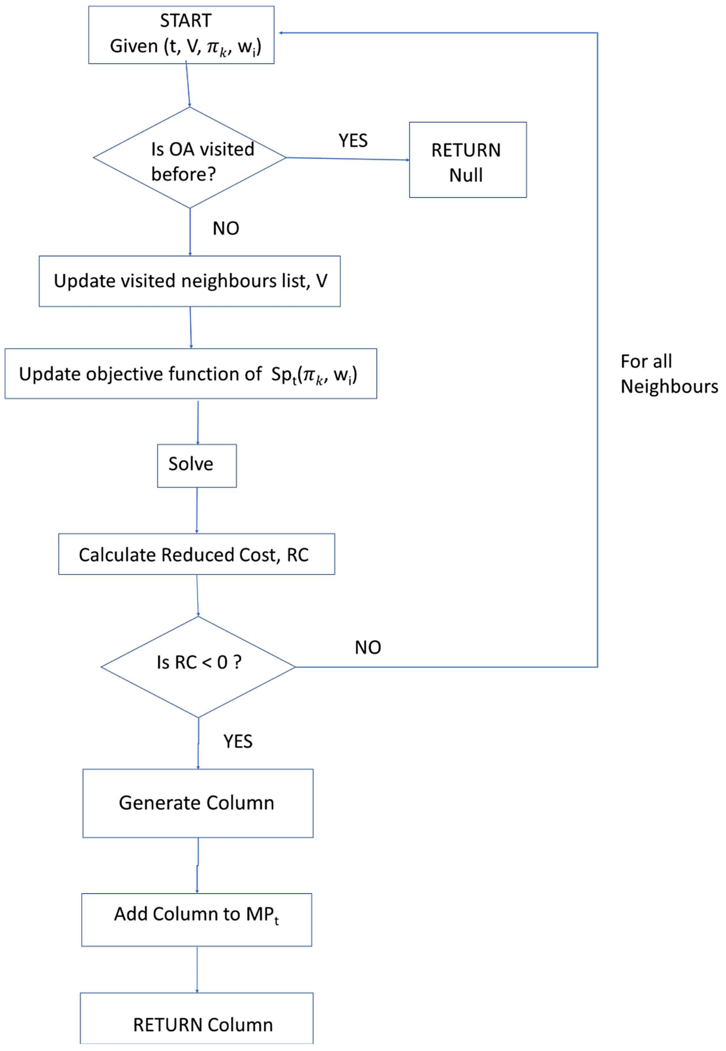
START
Given (t, V, π_k, w_i)
Is OA visited before?
YES
RETURN Null
NO
Update visited neighbours list, V
Update objective function of $Sp_t(\pi_k, w_i)$
Solve
Calculate Reduced Cost, RC
Is RC < 0 ?
NO
For all Neighbours
YES
Generate Column
Add Column to MP_t
RETURN Column

FIGURE 7. Flowchart of GETCOLUMN procedure given in Algorithm 5.

MP_i

$$\text{Minimize} \quad \sum_{i=1}^{N}\sum_{j=1}^{n_i}\sum_{p} c_{ij} x_i^p \lambda_i^p \tag{4.5a}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\sum_{j=1}^{n_i}\sum_{p} a_{ij}^k x_i^p \lambda_i^p \geq r^k \qquad \forall k = 1,2,\ldots,K \tag{4.5b}$$

$$\sum_{p} \lambda_i^p = 1 \qquad \forall i = 1,2,\ldots,N \tag{4.5c}$$

$$\lambda_i^p \geq 0 \qquad \forall x_i^p \in P_i \qquad \forall i = 1,2,\ldots,N. \tag{4.5d}$$

At termination, optimal objective function value of any local master problem gives the optimal objective value for the global problem. In the worst case, local master problem may eventually converge to the classical master problem of centralized approach. OA i can derive global optimal solution value of its own variables x_{ij}^{opt} by using the following equation:

$$(x_{ij}^{opt}) = \left(\sum_{p \in \bar{P}_i} x_{ij}^p \lambda_i^p \right), \tag{4.6}$$

where $\bar{P}_i$ denotes the set of extreme points which are used to generate columns in OA i's local master problem. Then optimal solution of the global problem is obtained by combining all OA's individual global optimal solutions.

Proposition 4.1. *Decentralized Dantzig–Wolfe Decomposition yields an optimal solution for BALP (if one exists) within a finite number of iterations if communication network is strongly connected.*

Proof. See Appendix A.2. □

5. Numerical experiments

In this section, we apply Decentralized Benders Decomposition and Decentralized Dantzig–Wolfe Decomposition to solve randomly generated block angular linear problems and Multi-Commodity Network Flow problems. We present test results with a discussion.

5.1. Test sets

We use two groups of problems to test the correctness and performance of the proposed methods. For the first one, we generate random block angular linear problems by using the same strategy of the authors in [10]. According to this, the constraint matrix A_i of the problems consists of non-negative random numbers in the range [0,10] with density 30%. The objective function coefficients c_i are generated from the range [10, 20] while right hand side values are selected from [100, 500]. Table 1 presents the dimensions of randomly generated problems in three sets. In the first set, problems has fixed size of 500 × 1000 while the number of OAs varies. In the second set, each problem has twenty OAs with varying size. In the third set, the problems has varying size with varying number of OAs, however each block has same size of 20 × 30.

The second group is Multi-commodity network flow (MCNF) problems that are one of the well-known problem types representing primal block angular structure. We use random generator *Mnetgen* [2] for MCNF problems that can be retrieved from [16]. These set of problems can be characterized by the number of nodes n and the number of commodities k where $n \in \{64, 128, 256\}$ and $k \in \{4, 8, 16, \ldots, n\}$. For any pair of *(n,k)*, Mnetgen randomly generates twelve problems such that six of the problems are *dense* with $m/n \approx 8$ and the other six problems are *sparse* with $m/n \approx 3$. Within each group of six problems, three problems are *easy* and the other three problems are *hard*.

5.2. Results and discussion

In this section, we present computational results performed on test instances. We implemented proposed algorithms with C# utilizing CPLEX 12.5 running on a Windows 10 PC with a 3.6 GHz CPU and 32 GB RAM. We use Star, Ring and Mesh topologies in Figure 4 as strongly connected communication graph among OAs. For Decentralized Benders Decomposition and Decentralized Dantzig–Wolfe Decomposition we allow equal run time for each OA that sum up to two hours for each problem. Also we report the results of Classical Benders Decomposition and Classical Dantzig–Wolfe Decomposition of as a benchmark. We find lower and upper bounds on the objective function value if the algorithm does not converge to optimal solution within the allowed time and report the gap percent.

For the first group of problems in Table 1, we generate five instances randomly for each problem type and report the average as the result. Table 2 presents results for these problems on Decentralized Benders

Table 1. Dimensions for randomly generated problems.

Instance no	Number of blocks	Number of variables in each block	Number of rows in each block constraints	Number of constraints constraints	Percent of complicating
1	2	500	200	100	20%
2	5	100	80	100	20%
3	10	50	40	100	20%
4	20	25	20	100	20%
5	50	10	8	100	20%
6	20	10	4	20	20%
7	20	20	8	40	20%
8	20	30	12	60	20%
9	20	40	16	80	20%
10	20	50	20	100	20%
11	5	30	20	25	20%
12	10	30	20	50	20%
13	20	30	20	100	20%
14	40	30	20	200	20%

Decomposition. For the first set of problems, the size of the overall problem is fixed. Thus, subproblem size becomes smaller as the number of OAs increases. However, *Gap%* increases with the number of OAs because of allowing less time for each OA in a problem having more OAs. For the second set, we can observe the effect of the subproblem size on convergence. Harder subproblems results in higher *Gap%*. For the third set, we can observe the effect of the number of OAs since the subproblem size is fixed. Problem having more OAs need more time to converge. Ring topology outperforms the others almost in all instances. Mesh topology results in smaller *Gap%* than Star topology.

Table 3 presents results for first group of problems on Decentralized Dantzig–Wolfe Decomposition. Results in the first set presents the effect of communication network. Convergence time in Star topology increases first because time spent for communication is more than time spent for solving the subproblems. Convergence time decreases whenever the subproblems becomes easy enough to solve. For Ring topology, convergence time decreases because the subproblems are getting easier to solve. For Mesh topology, convergence time decreases initially as the subproblems are getting easier to solve, however more OAs results in higher convergence time. There is an increase in convergence time for the second set and the third set. However, the algorithm reacts more to size of the subproblems than the number of agents for all topology types. While Ring topology outperforms the others, Mesh topology converges faster than Star topology.

Tables 4 and 5 present results for Mnetgen instances for Decentralized Benders Decomposition. While eight out of twelve M64.4.* problems converge to optimal solution, six out of twelve M64.8.* problems converge. The results shows that there is not a clear dominance of any topology to the others. However, in most of the problems Star topology has longer convergence time than the others Table 6 presents results for Centralized Benders Decomposition. As expected, Centralized Benders Decomposition outperforms Decentralized Benders Decomposition in all instences. The main reason for this is the existence of the center in Centralized Benders Decomposition while it lacks in Decentralized Dantzig–Wolfe Decomposition. The center solves the problem only once and announce the results to the others in Centralized Benders Decomposition, but in Decentralized Benders Decomposition the problem is solved many times since each OA solves the problem by itself. Also, M64.4.5 and M64.8.7 converge to optimum within allowed time while the best reported *Gap%* for this problems are %2 and %0.6, respectively for Decentralized Benders Decomposition. Centralized Benders Decomposition performs better because we allow same amount of time for both methods to converge. While Decentralized Benders

TABLE 2. Results for Decentralized Benders Decomposition on random set of Table 1.

Instance no	Optimal Obj.Fn. value	STAR				RING				MESH			
		LB	UB	*Gap*%	CPU Time(s)	LB	UB	*Gap*%	CPU Time(s)	LB	UB	*Gap*%	CPU Time(s)
1	292.73	281.22	296.51	5.22	7200	281.22	294.42	4.50	7200	284.72	296.72	4.03	7200
2	776.96	758.73	1156.31	51.17	7200	758.73	913.39	19.90	7200	758.73	1156.31	51.17	7200
3	1535.00	1469.35	2696.85	79.96	7200	1469.35	2576.85	72.14	7200	1469.35	2648.42	76.81	7200
4	3066.16	2984.32	7736.83	154.99	7200	2984.32	5993.92	98.15	7200	2984.32	7490.82	146.97	7200
5	7416.69	7389.58	23 587.04	218.39	7200	7389.58	21 000.68	183.51	7200	7389.58	23 587.04	218.39	7200
6	2170.44	2170.44	2170.44	–	1585	2170.44	2170.44	–	1040	2170.44	2170.44	–	4699
7	2257.37	2032.35	2752.59	31.90	7200	2153.29	2427.62	12.15	7200	2032.35	2752.59	31.90	7200
8	2406.45	2169.42	4494.05	96.59	7200	2169.42	4511.44	97.32	7200	2169.42	4480.24	96.02	7200
9	2510.25	2138.83	4848.40	107.94	7200	2138.83	4848.40	107.94	7200	2138.83	4848.40	107.94	7200
10	2498.82	2276.55	5335.55	122.41	7200	2276.55	5335.55	122.41	7200	2276.55	5335.55	122.41	7200
11	732.75	732.75	732.75	–	230	732.75	732.75	–	196	732.75	732.75	–	229
12	1397.83	1369.66	1493.95	8.89	7200	1369.66	1493.95	8.89	7200	1369.66	1493.95	8.89	7200
13	2950.30	2852.13	6463.40	122.40	7200	2852.13	5610.44	93.49	7200	2852.13	6463.40	122.40	7200
14	5642.89	5519.44	14,880.13	165.88	7200	5519.44	14 820.44	164.82	7200	5519.44	14 880.13	165.88	7200

Table 3. Results for Decentralized Dantzig–Wolfe Decomposition on random set of Table 1.

Instance no	Optimal Obj.Fn. value	STAR	RING	MESH
		CPU Time(s)	CPU Time(s)	CPU Time(s)
1	292.73	36.29	36.47	37.02
2	776.96	37.02	19.98	21.65
3	1535.00	39.63	17.21	20.06
4	3066.16	30.42	11.86	16.58
5	7416.69	28.55	3.65	18.63
6	2170.44	1.58	0.38	1.06
7	2257.37	7.88	2.84	4.81
8	2406.45	38.84	15.62	19.67
9	2510.25	222.02	40.48	74.78
10	2498.82	611.71	38.91	142.01
11	732.75	0.49	0.22	0.30
12	1397.83	5.69	0.97	2.16
13	2950.30	61.75	4.84	18.49
14	5642.89	666.40	24.64	185.32

Decomposition allocates this time to each OA to solve its own problem, in Centralized Benders Decomposition, the center uses whole allowed time by itself only to solve the problem.

Table 7 shows the results for MNetgen Instances for Decentralized Dantzig–Wolfe Decomposition. Decentralized Dantzig–Wolfe Decomposition method converges to optimal solution under one second for M64.4.* instances and within seconds for M64.8.* instances. Ring topology has the smallest convergence time for most of the instances while results for Ring topology and Mesh topology are very close to each other. Star topology requires more computational time for convergence than the others. Table 7 also shows the results for MNetgen Instances for Centralized Dantzig–Wolfe Decomposition as a benchmark. The results support our assumption that Centralized Dantzig–Wolfe Decomposition outperforms Decentralized Dantzig–Wolfe Decomposition for similiar reasons that we explained for Centralized Benders Decomposition and Decentralized Benders Decomposition.

We conclude this section with a summary of observations under the following headings.

Comparison of the methods

In Linear Programming case, Benders Decomposition is defined as Dantzig–Wolfe Decomposition applied to the dual [7]. So, from theoretical point of view, Benders Decomposition and Dantzig–Wolfe Decomposition are equivalent to each other. However, they may differ when looking at the computational side. Generally, Benders decomposition is appropriate for problems with complicating variables, while Dantzig–Wolfe Decomposition is suitable for problems with complicating constraints.

We observe the similar properties for Decentralized Benders Decomposition and Decentralized Dantzig–Wolfe Decomposition. Experimental results show that Decentralized Dantzig–Wolfe Decomposition outperforms Decentralized Benders Decomposition in all instances. The main reason for this is the problem structure. This work is mainly focused on Linear Programming Problems with Primal Block Angular Structure. Recall that this structure is characterized by complicating constraints linking the local constraints on the diagonal. To apply Decentralized Benders Decomposition to this problem structure, we introduce a variable for each OA for a single common resource. By this way, we convert the primal BALP problem to a dual one. However, the complicating constraints in local master problem and local constraint set in each subproblem gets larger and

TABLE 4. Results for Decentralized Benders Decomposition for M64.4.* Instances.

Instance no	Optimal Obj.Fn. value	STAR				RING				MESH			
		LB	UB	*Gap*%	CPU Time(s)	LB	UB	*Gap*%	CPU Time(s)	LB	UB	*Gap*%	CPU Time(s)
M64.4.1	290 806.3	290 806.3	290 806.3	–	176.1	290 806.3	290 806.3	–	118.3	290 806.3	290 806.3	–	97.8
M64.4.2	336 019.9	336 019.9	336 019.9	–	41.8	336 019.9	336 019.9	–	21.2	336 019.9	336 019.9	–	21.1
M64.4.3	348 966.6	348 966.6	348 966.6	–	1	348 966.6	348 966.6	–	0.8	348 966.6	348,966.6	–	0.8
M64.4.4	412 475.8	390 540.3	412 795	5.39	7200	390 540.3	412 707.2	5.37	7200	390 540.3	413 212	5.49	7200
M64.4.5	390 578.5	378 690.2	390 605.1	3.05	7200	382 769.2	390 612.6	2	7200	382 204.2	390 594.3	2.1	7200
M64.4.6	506 554.4	506 554.4	506 554.4	–	5.6	506 554.4	506 554.4	–	5.1	506 554.4	506 554.4	-	4.8
M64.4.7	147 862.1	147 862.1	147 862.1	–	71	147 862.1	147 862.1	–	50.8	147 862.1	147 862.1	–	40.9
M64.4.8	165 185.3	165 185.3	165 185.3	–	343	165 185.3	165 185.3	–	265.2	165 185.3	165 185.3	–	254.6
M64.4.9	192 119.4	192 119.4	192 119.4	–	2.4	192 119.4	192 119.4	–	1.6	192 119.4	192 119.4	–	1.7
M64.4.10	167 479.5	165 710.0	167 755.4	1.22	7200	165 710.0	167 755.0	1.22	7200	165 710.0	167 502.5	1.07	7200
M64.4.11	193 238.4	192 338.7	193 562.6	0.63	7200	192 338.7	194 410.8	1.07	7200	192 338.7	197 863.5	0.78	7200
M64.4.12	192 400.1	192 400.1	192 400.1	–	378.6	192 400.1	192 400.1	–	250.9	192 400.1	192 400.1	–	313.4

TABLE 5. Results for Decentralized Benders Decomposition for M64.8.* Instances.

Instance no	Optimal Obj.Fn. value	STAR				RING				MESH			
		LB	UB	*Gap*%	CPU Time(s)	LB	UB	*Gap*%	CPU Time(s)	LB	UB	*Gap*%	CPU Time(s)
M64.8.1	622 280.4	622 280.4	622 280.4	–	607	622 280.4	622 280.4	–	247.8	622 280.4	622 280.4	–	292.2
M64.8.2	649 767.0	649 767.0	649 767.0	–	375.1	649 767.0	649 767.0	–	168.6	649 767.0	649 767.0	–	299.9
M64.8.3	750 938.0	750 938.0	750 938.0	–	44.6	750 938.0	750 938.0	–	19.3	750 938.0	750 938.0	–	29.9
M64.8.4	761 862.7	740 240.1	808 421.5	8.9	7200	741 868	835 831.6	12.3	7200	737 493.5	835 831.6	12.9	7200
M64.8.5	753 927.6	728 256.5	808 359.9	10.6	7200	729 050.5	808 360	10.5	7200	728 744.3	808 360	10.5	7200
M64.8.6	929 066.8	929 066.8	929 066.8	–	3 221.3	929 066.8	929 066.8	–	1171	929 066.8	929 066.8	–	1272.8
M64.8.7	304 045.0	298 939.3	306 481.5	2.4	7200	302367.1	304 912.2	0.8	7200	304 045.5	304 237	0.6	7200
M64.8.8	355 699.7	355 699.7	355 699.7	–	1096.4	355 699.7	355 699.7	–	1605.4	355 699.7	355 699.7	–	555.9
M64.8.9	357 649.1	357 649.1	357 649.1	–	1161.8	357 649.1	357 649.1	–	883.2	357 649.1	357 649.1	–	1084.1
M64.8.10	361 802.0	357 717.8	511 124.9	42.4	7200	357 717.8	542 878	51.1	7200	357 717.8	451 518	25.9	7200
M64.8.11	418 824.0	408 052.3	652 132.5	58.2	7200	407 577.5	652 132.5	58.3	7200	405 916.3	652 132.5	58.7	7200
M64.8.12	394 051.0	383 193	522 198.5	35.2	7200	382 840.1	558 636.9	44.6	7200	383 193	569 272	47.2	7200

Table 6. Results for Centralized Benders Decomposition for MNetgen Instances.

Instance no	Optimal Obj.Fn. value	LB	UB	*Gap*%	CPU Time(s)
M64.4.1	290 806.3	290 806.3	290 806.3	–	20.7
M64.4.2	336 019.9	336 019.9	336 019.9	–	5.3
M64.4.3	348 966.6	348 966.6	348 966.6	–	0.3
M64.4.4	412 475.8	400 086	412 683.8	3.05	7200
M64.4.5	390 578.5	390 578.5	390 578.5	–	1279.3
M64.4.6	506 554.4	506 554.4	506 554.4	–	1
M64.4.7	147 862.1	147 862.1	147 862.1	–	600.7
M64.4.8	165 185.3	165 185.3	165 185.3	–	8.4
M64.4.9	192 119.4	192 119.4	192 119.4	–	0.6
M64.4.10	167 479.5	165 710.0	167 668.6	1.16	7200
M64.4.11	193 238.4	192 338.7	207 624.2	7.9	7200
M64.4.12	192 400.1	192 400.1	192 400.1	–	24.5
M64.8.1	622 280.4	622 280.4	622 280.4	–	41
M64.8.2	649 767.0	649 767.0	649 767.0	–	22.2
M64.8.3	750 938.0	750 938.0	750 938.0	–	3.3
M64.8.4	761 862.7	737 920.4	808 421.5	9.2	7200
M64.8.5	753 927.6	728 018.8	783 858.5	7.4	7200
M64.8.6	929 066.8	929 066.8	929 066.8	–	549.7
M64.8.7	304 045.0	304 045.0	304 045.0	–	1829.2
M64.8.8	355 699.7	355 699.7	355 699.7	–	365.3
M64.8.9	357 649.1	357 649.1	357 649.1	–	4440.4
M64.8.10	361 802.0	356 884.6	387 304.7	8.4	7200
M64.8.11	418 824	407 130.6	534 460.3	30.4	7200
M64.8.12	394 051	381 255.8	474 594.2	23.6	7200

becomes computationally harder to solve. This results in worse experimental results when compared to the Decentralized Dantzig–Wolfe Decomposition.

Size of blocks

We observe that the size of the blocks influences the performance of the methods. Larger block size results in larger and harder to solve subproblems.

Number of blocks

We observe that as the number of blocks increase, the time until termination gets longer. This can be explained mainly with the fact that each block is associated with an OA. Both methods solve the overall problem for each OA before termination. Hence as the number of blocks increase, proposed methods solve the overall problem for more times.

Another reason is the increase in the amount of communication among OAs. As the number of OAs increases, each OA can have more neighbours. This results in having more communication rounds to get a cut or column.

Type of communication network

We can observe the effect of communication network on performance of the proposed methods while we solve problems having larger size. When the problem size gets larger, Ring topology outperforms the other topologies for Decentralized Dantzig–Wolfe decomposition. Mesh topology has quicker convergence than Star topology almost in all instances. This result holds for Decentralized Benders Decomposition too. Although Mesh topology outperforms Ring topology on small problems, for the larger problems Ring topology converges faster.

TABLE 7. Results for Decentralized Dantzig–Wolfe Decomposition and Centralized Dantzig–Wolfe Decomposition for MNetgen Instances.

Instance no	Optimal Obj.Fn. value	Decentralized DW			Centralized DW
		STAR	RING	MESH	
M64.4.1	290 806.3	0.081	0.053	0.058	0.024
M64.4.2	336 019.9	0.082	0.071	0.071	0.021
M64.4.3	348 966.6	0.010	0.013	0.010	0.002
M64.4.4	412 475.8	0.429	0.227	0.308	0.100
M64.4.5	390 578.5	0.465	0.332	0.329	0.168
M64.4.6	506 554.4	0.036	0.028	0.039	0.011
M64.4.7	147 862.1	0.231	0.146	0.169	0.057
M64.4.8	165 185.3	0.182	0.098	0.105	0.060
M64.4.9	192 119.4	0.018	0.015	0.017	0.003
M64.4.10	167 479.5	0.619	0.433	0.446	0.162
M64.4.11	193 238.4	0.274	0.215	0.222	0.069
M64.4.12	192 400.1	0.123	0.074	0.085	0.032
M64.8.1	622 280.4	0.122	0.062	0.064	0.018
M64.8.2	649 767.0	0.021	0.017	0.015	0.003
M64.8.3	750 938.0	0.103	0.046	0.067	0.020
M64.8.4	761 862.7	1.020	0.345	0.538	0.134
M64.8.5	753 927.6	2.799	0.850	1.206	0.380
M64.8.6	929 066.8	0.323	0.120	0.196	0.048
M64.8.7	304 045.0	7.645	2.997	4.174	0.977
M64.8.8	355 699.7	0.585	0.226	0.330	0.074
M64.8.9	357 649.1	1.067	0.417	0.542	0.152
M64.8.10	361 802.0	11.785	4.946	7.426	1.339
M64.8.11	418 824.0	13.789	4.392	6.261	1.855
M64.8.12	394 051.0	10.213	3.436	4.860	1.434
M128.32.1	11 186 573.8	320.685	21.224	115.798	9.944
M128.32.2	118 663 936.6	342.258	22.965	115.491	9.980
M128.32.3	122 476 676.8	143.936	12.030	58.701	4.158
M128.32.4	12 715 040.2	7094.051	355.515	2,403.788	187.800
M128.32.5	13 582 810.6	5390.729	208.135	985.815	125.418
M128.32.6	14 617 437.1	337.414	27.348	127.854	9.513

The reason for this is the cut or column exchange strategy of the proposed methods. While OA i gets cut or column from a neighbour t, both agents i and t adds that cut or column to its local master problem. Thus, while getting a cut or column from a further neighbour, that cut or column is added to all OAs in the path connecting two communicating OAs. In Ring topology, an OA i can reach any other OA t indirectly. Since the length of the path can be largest in Ring topology, any cut or column can be added to the local master of more OAs at a time. This results in fast convergence rate.

Cut or column exchange strategy also affects the number of communication rounds among OAs. To give an example, Figure 8 illustrates the total number of cuts/columns generated for solving Mnetgen M64.4.* instances by Decentralized Benders Decomposition and Decentralized Dantzig–Wolfe Decomposition, respectively. We can observe that the number of cuts or columns generated in Ring topology is less than the number of cuts/columns generated in other topologies almost in all cases for both methods. The main reason for this is appending a cut or column to the local master of all OAs in the path connecting two communicating OAs. In an iteration a cut or column may be added local master problem of more OAs in Ring topology and this results in faster

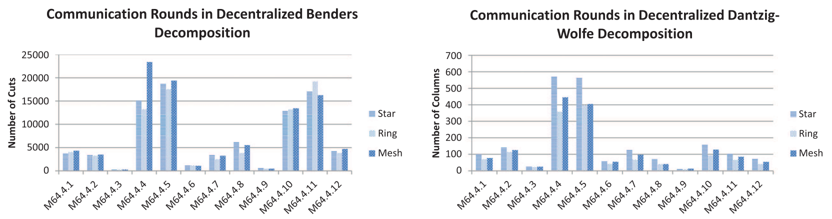

FIGURE 8. Communication rounds for Mnetgen M64.4.* Instances.

convergence. In most of the problems, Mesh topology generates fewer cuts or columns than Star topology, which confirms the speed of convergence of these methods.

6. Conclusions and future work

In this work, we propose Decentralized Benders Decomposition and Decentralized Dantzig–Wolfe Decomposition for large-scale linear block angular problems. We remark that, our main goal is not competing with the computational speed of a centralized algorithm. Instead, we primarily aim to propose decentralized solution approaches for decision makers that are unwilling to disclose their local data, but want to solve the global problem collaboratively in a peer-to-peer fashion.

From an organizational point of view, in Decentralized Benders Decomposition, local master problem shares common resources among OAs. In return, each subproblem finds its best solution for given allocations and generates a cut which includes implicit information of dual prices for common resources. On the other hand, in Decentralized Dantzig–Wolfe Decomposition local master problem shares prices on common resources. In return, each subproblem generates a column indicating explicit information that disclose optimal way of using common resources in terms of cost, profit or specific proposal.

We prove that the proposed methods can reach global optimal solution in a finite number of iterations. We confirm theoretical results with computational experiments on randomly generated test instances and also on Multi-commodity Network Flow Problems. We observe that Decentralized Dantzig–Wolfe Decomposition shows faster convergence rate than Decentralized Benders decomposition.

There are three main alternatives for multiple decision makers to solve an overall block angular linear optimization problem. The first one is the centralization approach, which converges to optimal solution rapidly but requires a central coordination unit having full access to managerial information of all decision makers. The second one is Decentralized Benders Decomposition, which requires revealing dual information however has slower convergence rate. Finally, the third one is Decentralized Dantzig–Wolfe Decomposition, which has faster convergence rate but requires revealing primal information. Thus, we propose two decentralized methods for decision makers to make a choice with a trade-off between the degree of the information that they want to disclose and the speed of the convergence time.

Decentralized methods for optimization problems are inherently suited to parallel or distributed computing opportunity. Thus, as a future research direction, parallelization of proposed algorithms will be desirable. Moreover, we intend to explore decentralization in integer programs since decentralized decision making is rarely applied to integer programs in the literature.

Appendix A.

A.1. Convergence proof for Decentralized Benders Decomposition

We give a formal proof for convergence of Decentralized Benders Decomposition for linear programs in three parts. In the first part we show that there are finitely many cuts that can be generated. The second part shows that each cut can be generated at most once. Finally, in the third part, we show that any violated cut is detected and added to the relaxed local master problem. Then, convergence of Decentralized Benders Decomposition for linear programs follows.

PART I: There is a finite number of Benders Cuts that can be generated.

Proof of the first part is based on projection theory which will be assumed to fulfill conditions similar to those utilized in the convergence analysis of Classical Benders Decomposition in [3]. First we state the following theorem which defines the multipliers obtained by projecting out the x_{ij} variables in (3.2) and also the projection of the polyhedron.

Theorem A.1. *If*

$$P_i = \left\{ (x,r) \in \Re^{n_i} \times \Re^k | \sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq r_i^k \quad \forall k = 1,2,\dots,K, \quad \sum_{j=1}^{n_i} b_{ij} x_{ij} \geq l_i \right\} \tag{A.1}$$

then projecting out the x_{ij} variables from the system generates the nonnegative multipliers $\{(\pi_i^k, w_i), \forall k = 1,2,\dots,K\}$ such that

$$\sum_{k=1}^{K} \sum_{j=1}^{n_i} a_{ij}^k \pi_i^k + \sum_{j=1}^{n_i} b_{ij} w_i = 0. \tag{A.2}$$

Also the projection of the polyhedron is

$$proj_r(P_i) = \left\{ r \in \Re^k | \pi_i^k r_i^k \geq 0 \quad \forall k = 1,2,\dots,K \right\}. \tag{A.3}$$

Then we state two propositions which are adapted from [15] for BALP. One can relate the multipliers obtained by projection and the extreme rays of the projection cone as a result of these propositions.

Proposition A.2. *If P_i is given in (A.1) then $proj_r(P_i)$ is given in (A.3), where $\{(\pi_i^k, w_i), \forall k = 1,2,\dots,K\}$ are the extreme rays of the projection cone*

$$C_x(P_i) = \left\{ (\pi, w) | \sum_{k=1}^{K} \sum_{j=1}^{n_i} a_{ij}^k \pi_i^k + \sum_{j=1}^{n_i} b_{ij} w_i = 0, \pi_i^k \geq 0 \quad \forall k = 1,2,\dots,K, \quad w_i \geq 0 \right\}. \tag{A.4}$$

Proposition A.3. *If P_i is given in (A.1) and $\{(\pi_i^k, w_i), \forall k = 1,2,\dots,K\}$ are the multipliers that are generated using projection, then the extreme rays of the projection cone in (A.4) are contained in this set of multipliers.*

We omit the proof of Propositions A.2 and A.3 here. Refer to [15] for proof of generalized cases as Propositions 2.22 and 2.23, respectively.

Theorem A.1 states that the inequalities $\pi_i^k r_i^k \geq 0 \quad \forall k = 1,2,\dots,K$ that results from projecting out the x_{ij} variables from the system defines $proj_r(P_i)$. By Proposition A.2, $proj_r(P_i)$ can also be generated by the extreme rays of the projection cone $C_x(P_i)$. Proposition A.3 defines the relationship between the multipliers $\{(\pi_i^k, w_i), \forall k = 1,2,\dots,K\}$ generated using projection and the extreme rays of $C_x(P_i)$. We use this relationship to conclude that finite number of cuts are generated. Thus, we apply projection to BALP. The aggregate problem

(3.1) can be equivalently stated as the following:

$$\text{Minimize } z_0 \tag{A.5a}$$

$$\text{subject to: } z_0 - \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \geq 0 \tag{A.5b}$$

$$\sum_{i=1}^{N} r_i^k - r^k = 0 \qquad \forall k = 1,2,\ldots,K \tag{A.5c}$$

$$\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq r_i^k \qquad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K \tag{A.5d}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \geq l_i \qquad \forall i = 1,2,\ldots,N \tag{A.5e}$$

$$x_{ij} \geq 0 \qquad \forall i = 1,2,\ldots,N \qquad \forall j = 1,2,\ldots,n_i \tag{A.5f}$$

$$r_i^k \text{ unrestricted} \qquad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K. \tag{A.5g}$$

Let us assign the multiplier u_0 to the constraint (A.5b) and the vector of multipliers π, w, u to the constraints (A.5d)–(A.5f), respectively. Using these multipliers, we project out the x_{ij} variables. This gives the following:

$$\text{Minimize } z_0 \tag{A.6a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k = r^k \qquad \forall k = 1,2,\ldots,K \tag{A.6b}$$

$$u_0^p z_0 \geq (\pi_i^k)^p r_i^k + {w_i}^p l_i \qquad \forall i \quad \forall k \quad \forall p \tag{A.6c}$$

$$r_i^k \text{ unrestricted} \qquad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K. \tag{A.6d}$$

Without loss of generality, multipliers $(u_0^p, (\pi_i^k)^p, {w_i}^p, (u_{ij})^p)$ can be re-scaled. Assume $u_0^p = 1$ for $p = 1,2,\ldots,t$, and $u_0^p = 0$ for $p = t+1,\ldots,P$. Hence the resulting problem is:

$$\text{Minimize } z_0 \tag{A.7a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k = r_k \qquad \forall k = 1,2,\ldots,K \tag{A.7b}$$

$$z_0 \geq (\pi_i^k)^p r_i^k + {w_i}^p l_i \qquad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K \qquad \forall p = 1,2,\ldots,t \tag{A.7c}$$

$$0 \geq (\pi_i^k)^p r_i^k + {w_i}^p l_i \qquad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K \qquad \forall p = t+1,\ldots,P \tag{A.7d}$$

$$r_i^k \text{ unrestricted} \qquad \forall i = 1,2,\ldots,N \qquad \forall k = 1,2,\ldots,K. \tag{A.7e}$$

From theory of projection, r^* is an optimal solution to (A.7) if and only if there is an x^* such that (x^*, r^*) is an optimal solution to the aggregate problem. By Proposition A.3, the extreme rays of the projection cones

$$C_x(P_i) = \left\{ (u_0, \pi, w, u) | \sum_{k=1}^{K} a_{ijk}\pi_i^k + b_{ij}w_i + u_{ij} - u_0 c_{ij} = 0 \quad \forall j = 1,2,\ldots,n_i, (u_0, \pi, w, u) \geq 0 \right\} \tag{A.8}$$

are contained in the set of multipliers

$$\{(u_0^p, (\pi_i^k)^p, {w_i}^p, (u_{ij})^p) | \forall j = 1,2,\ldots,n_i \quad \forall k = 1,2,\ldots,K \quad \forall p = 1,2,\ldots,P\} \tag{A.9}$$

generated by projection $\forall i = 1,2,\ldots,N$. By Proposition A.2, only the extreme rays of the projection cones $C_x(P_i) \quad \forall i = 1,2,\ldots,N$ are needed to generate constraints (A.7c) and (A.7d) which characterize the projection into r space. Therefore, we can conclude that the constraints (A.7c) and (A.7d) are generated from the extreme rays of the projection cones.

Proposition A.4. *If the multipliers in* (A.9) *are the extreme rays of the projection cone in* (A.8) *scaled so that* $u_0^p = 1$, *for* $p = 1, 2, \dots, t$ *and* $u_0^p = 0$, *for* $p = t+1, 2, \dots, P$, *then* $((\pi_i^k)^p, (w_i)^p)$, *for* $p = 1, 2, \dots, t$ *are the extreme points of the polyhedron*

$$\left\{ (\pi_i^k, w_i) | \sum_{k=1}^{K} a_{ijk}\pi_i^k + b_{ij}w_i \le c_{ij}, (\pi_i^k, w_i) \ge 0 \qquad \forall j = 1, 2, \dots, n_i \right\} \tag{A.10}$$

and $((\pi_i^k)^p, (w_i)^p)$, *for* $p = t+1, 2, \dots, P$ *are the extreme rays of the associated recession cone*

$$\left\{ (\pi_i^k, w_i) | \sum_{k=1}^{K} a_{ijk}\pi_i^k + b_{ij}w_i \le 0, (\pi_i^k, w_i) \ge 0 \qquad \forall j = 1, 2, \dots, n_i \right\}. \tag{A.11}$$

If all the constraints associated with the extreme rays are generated, then (A.7) becomes *full master problem.* Since solving full master problem is not practical, a *relaxed master problem* having a subset of the constraints (A.7c) and(A.7d) is solved. If there is a constraint that violates the relaxed master problem's solution, then that constraint is added to the relaxed master. By Proposition A.4, one can find a constraint that violates the relaxed master problem by solving the following subproblem:

$$\text{Maximize} \sum_{k=1}^{K} \hat{r}_i^k \pi_i^k + l_i w_i \tag{A.12a}$$

$$\text{subject to:} \sum_{k=1}^{K} a_{ij}^k \pi_i^k + b_{ij} w_i \le c_{ij} \qquad \forall j = 1, 2, \dots, n_i \tag{A.12b}$$

$$\pi_i^k \ge 0 \qquad \forall i = 1, 2, \dots, N \qquad \forall k = 1, 2, \dots, K \tag{A.12c}$$

$$w_i \ge 0 \qquad \forall i = 1, 2, \dots, N. \tag{A.12d}$$

Assume $\bar{r}$ is a feasible solution to the relaxed master problem with objective function value $\bar{z}_0$. If (π_i^k, w_i) is an optimal solution to the subproblem and $\bar{z}_0 < \hat{r}_i^k \pi_i^k + l_i w_i$ then add the constraint $\bar{z}_0 \ge \hat{r}_i^k \pi_i^k + l_i w_i$ to the relaxed master problem. If the subproblem is unbounded, then there exists an extreme ray (π, w) in the recession cone such that $\hat{r}_i^k \pi_i^k + l_i w_i > 0$. In this case, add the constraint $\hat{r}_i^k \pi_i^k + l_i w_i \le 0$ to the relaxed master problem. Therefore, there is only one constraint associated with each extreme ray or extreme point. Since number of extreme rays and extreme points is finite, one can conclude that the number of cuts that can be generated is also finite.

PART II: A new cut is generated at each iteration.

Proposition A.5. *At each iteration, the constraint added to the relaxed master problem is unique.*

Proof. We proved that each cut generated by the subproblem is associated with either an extreme ray or an extreme point. When a new cut is generated and added to the relaxed master problem, then relaxed master problem excludes the associated extreme ray or extreme point in the solution set for subsequent iterations. Hence each cut can be generated and added to the relaxed master at most once. □

PART III: Any violated cut can be detected and added to any local master problem.

Proposition A.6. *Decentralized Benders Decomposition yields an optimal solution for BALP (if one exists) within a finite number of iterations if communication network is strongly connected.*

Proof. Assume that the cut exchange network is *strongly connected*. Then, by definition of strong connectivity, there exist a directed path between any pair of the nodes. Hence any cut generated by any node can reach all the nodes in the graph along the directed path *via* recursive GETCUT procedure. In other words, Benders Cut generated by any one of the OAs can be added to relaxed local master of any other OA. Decentralized Benders Decomposition algorithm terminates when no new cut is generated for any one of OAs. Hence the convergence of Decentralized Benders Decomposition to the global optimal solution in a finite number of iterations follows from having finite number of cuts, each of which is generated and added at most once to any OA's local master problem. □

A.2. Convergence proof for Decentralized Dantzig–Wolfe Decomposition

We prove the finite convergence of Decentralized Dantzig–Wolfe decomposition in three parts. In the first part we show that there are finitely many columns that can be generated. The second part shows that each column can be generated at most once. Finally, in the third part, we show that when a new column is generated, it can be added to any relaxed local master problem. Then, convergence of Decentralized Benders Decomposition for linear programs follows.

PART I: There are finitely many columns to be generated.

Proposition A.7. *The number of columns that can be generated in Decentralized Dantzig–Wolfe Decomposition Algorithm is finite.*

Proof. Let $S_i = \{x_{ij} | \sum_{j=1}^{n_i} b_{ij} x_{ij} \geq l_i\}$ denotes the feasible region of ith pricing subproblem (SP_i). Then S_i has finitely many extreme points and extreme rays since it is a polyhedron and any point can be expressed as sum of a convex combination of extreme points and a non-negative linear combination of extreme rays by Minkowski's Representation theorem. For a (SP_i) having bounded feasible region, optimal solution is at one of its extreme points since it is an linear programming problem. A new column can be generated with respect to any optimal solution is given by (4.4). Hence, each extreme point is associated with exactly one column. For a (SP_i) having unbounded feasible region, the solution attains at one of the extreme rays. Hence, similar results holds for an extreme ray. Therefore, finiteness of the number of columns follows. □

PART II: A new column is generated at each iteration.

Proposition A.8. *Decentralized Dantzig–Wolfe Decomposition yields an unique column at each iteration.*

Proof. Local master problem for OA i given in (4.1) is a linear programming problem. Thus, one can calculate reduced cost of any variable x_{ij} by (4.2). Since MP_i is a minimization problem, at optimality, the reduced cost of any variable is non-negative. A variable having negative reduced cost may improve the objective function value of MP_i if it enters the basis. Pricing subproblem (SP_i) searches for the variable having most negative reduced cost and adds associated column to the MP_i. Hence if a column has already added to the MP_i, pricing subproblem cannot generate the same column again since its reduced cost is non-negative. Therefore, each column can be generated and added to the any local relaxed master problem at most once. □

PART III: If a new column is generated, then it can be added to any local master problem.

Proposition A.9. *Decentralized Dantzig–Wolfe Decomposition yields an optimal solution for BALP (if one exists) within a finite number of iterations if communication network is strongly connected.*

Proof. Assume a strongly connected communication network for exchanging columns among OAs. Thus, any column generated by any OA can be added to relaxed local master of any other OA in the network along the directed path *via* recursive GETCOLUMN procedure. Decentralized Dantzig–Wolfe Decomposition algorithm terminates when there is no variable having negative reduced cost for any one of OAs. Therefore, finite convergence of Decentralized Dantzig–Wolfe Decomposition Algorithm for BALP follows from Propositions A.7 and A.8. □

Acknowledgements. This work was supported by Boğaziçi University Research Project under contract no. 11A03D6.

References

[1] M. Albrecht and H. Stadtler, Coordinating decentralized linear programs by exchange of primal information. *Eur. J. Oper. Res.* **247** (2015) 788–796.

[2] A. Ali and J.L. Kennington, Mnetgen program documentation. Technical report, Department of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, TX (1977).

[3] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* **4** (1962) 238–252.

[4] S.P. Bradley, A.C. Hax and T.L. Magnanti, Applied Mathematical Programming. Addison-Wesley Publishing Company, Reading, MA (1977).

[5] M. Bürger, G. Notarstefano and F. Allgöwer, Locally constrained decision making via two-stage distributed simplex. In: *Decision and Control and European Control Conference (CDC-ECC)* (2011) 5911–5916.

[6] G. Dantzig, Linear Programming and Extensions. Princeton Landmarks in Mathematics and Physics. Princeton University Press (2016).

[7] G.B. Dantzig and M.N. Thapa, Linear programming: 2: Theory and Extensions. Springer Science & Business Media (2006).

[8] G.B. Dantzig and P. Wolfe, Decomposition principle for linear programs. *Oper. Res.* **8** (1960) 101–111.

[9] H. Dutta and H. Kargupta, Distributed linear programming and resource management for data mining in distributed environments. In: *IEEE International Conference on Data Mining Workshops* (2008) 543–552.

[10] Y. Hong, J. Vaidya and H. Lu, Secure and efficient distributed linear programming. *J. Comput. Secur.* **20** (2012) 583–634.

[11] I.-J. Jeong and V. Jorge Leon, Distributed allocation of the capacity of a single-facility using cooperative interaction via coupling agents. *Int. J. Prod. Res.* **41** (2003) 15–30.

[12] E. Kalvelagen, Column Generation with Gams. Gams Development Corp. Washinton, DC (2003).

[13] N. Karmarkar, A new polynomial-time algorithm for linear programming. *Combinatorica* **4** (1984) 373–395.

[14] S.Y.P. Lu, H.Y.K. Lau and C.K.F. Yiu, A hybrid solution to collaborative decision-making in a decentralized supply-chain. *J. Eng. Technol. Manage.* **29** (2012) 95–111.

[15] R.K. Martin, Large Scale Linear and Integer Optimization: A Unified Approach. Springer US (2012).

[16] Multicommodity problems. Available from: http://www.di.unipi.it/optimize/Data/MMCF.html. (2020).

[17] Y. Pochet and L.A. Wolsey, Production Planning by Mixed Integer Programming. *Springer Series in Operations Research and Financial Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006).

[18] J.B. Rosen, Primal partition programming for block diagonal matrices. *Numer. Math.* **6** (1964) 250–260.

[19] S. Schleicher, Decentralized optimization of linear economic systems with minimum information exchange of the subsystems. *Z. Nationalökonomie* **31** (1971) 33–44.

[20] S.J. Stoyan, M.M. Dessouky and X. Wang, Introduction to large scale linear programming and applications. In: Wiley Encyclopedia of Operations Research and Management Science (2010).

[21] Z.C. Taşkın, Benders decomposition. In: Wiley Encyclopedia of Operations Research and Management Science (2010).

[22] Z.C. Taşkın, S. Ağralı, A. Tamer Unal, V. Belada and F. Gokten-Yilmaz, Mathematical programming-based sales and operations planning at Vestel Electronics. *Interfaces* **45** (2015) 325–340.