DECENTRALIZED DECOMPOSITION METHODS FOR BLOCK ANGULAR LINEAR AND INTEGER PROGRAMMING PROBLEMS

by

M. Aslı Aydın

B.S., Mathematics, Boğaziçi University, 2003

M.S., Industrial Engineering, Bahçeşehir University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering
Boğaziçi University
2016

DECENTRALIZED DECOMPOSITION METHODS FOR BLOCK ANGULAR LINEAR AND INTEGER PROGRAMMING PROBLEMS

APPROVED BY:

Assoc. Prof. Z. Caner Taşkın (Thesis Supervisor)	....
Prof. Serpil Sayın	....
Prof. Kuban Altınel	....
Assoc. Prof. Ali Tamer Ünal	....
Assoc. Prof. Barış Selçuk	....

DATE OF APPROVAL: 18.07.2016

ACKNOWLEDGEMENTS

There are many people who have contributed me to complete this dissertation. First and foremost, I would like to thank my thesis advisor Assoc. Prof. Z. Caner Taşkın for his kind and patient guidance. He provided me by spending his precious time to advise me for research, guide me for writing scientific papers and encourage me to participate in scientific organizations. I was very fortunate to work with him.

I would like to thank my committee members Prof. Serpil Sayın and Prof. Kuban Altınel for their advice and helpful comments on my thesis. I also thank to Assoc. Prof. Ali Tamer Ünal and Assoc. Prof. Barış Selçuk for taking a part in my committee.

I thank to my parents for their endless love, continuous support and encouragement they show not only during this long period but throughout my life. I gratefully thank to my husband Tarkan Aydın for always being there for me. I could never thank him enough. My little daughter Ayşe Nil, I thank you for being such a wise and angel like baby.

Finally, I acknowledge the support of my work by Boğaziçi University Research Project under contract no. 11A03D6.

ABSTRACT

DECENTRALIZED DECOMPOSITION METHODS FOR BLOCK ANGULAR LINEAR AND INTEGER PROGRAMMING PROBLEMS

In this thesis, we propose *Decentralized Benders decomposition* and *Decentralized Dantzig-Wolfe decomposition* for block angular linear programs and *Decentralized L-Shaped Method* for block angular integer programs. We exploit the block angular structure of the problem to decompose the overall problem into several subproblem-local master problem pairs, each of which is associated with an independent decision maker. Then the decision makers of equal hierarchy level solve the overall problem cooperatively by exchanging minimal required information through a peer-to-peer communication network without need of a central coordination unit. The main difference between the proposed Decentralized Benders Decomposition and Decentralized Dantzig-Wolfe Decomposition is the type of the information disclosed. While Decentralized Benders Decomposition requires exchange of dual information, in Decentralized Dantzig-Wolfe Decomposition primal information is shared. We remark that our goal is not competing with the computational speed of a centralized algorithm. Instead, we primarily aim to propose a decentralized coordination scheme for decision makers that are unwilling to reveal their local data while solving the overall problem for a mutual benefit in such a case a central coordination unit is unavailable or not accepted.

We prove that the proposed methods converge to a global optimal solution in a finite number of iterations. Then we conduct computational experiments to evaluate the performance of the proposed methods. Also we investigate the impact of the underlying communication network computationally.

ÖZET

BLOK KÖŞEGEN YAPI GÖSTEREN DOĞRUSAL VE TAMSAYILI PROBLEMLER İÇİN DAĞITIK PARÇALAMA YÖNTEMLERİ

Bu tezde blok köşegen yapı gösteren doğrusal programlama problemleri için Dağıtık Benders Ayıştırma Yöntemi ve Dağıtık Dantzig-Wolfe Ayrıştırma Yöntemi, tam sayılı programlama problemleri içinse Dağıtık L-Şekil Yöntemi sunuyoruz. Bu yöntemler bütünsel problemi, gösterdiği blok köşegen yapı özelliğinden faydalanarak herbir karar verici için altproblem-yerel ana problem ikililerine ayrıştırır. Karar vericiler bütünsel problemi kendi aralarında merkezi bir yönetime ihtiyaç duymadan, kurdukları iletişim ağı üzerinden gerekli minimum bilgi paylaşımı yaparak işbirliği ile çözerler. Bu tezdeki amacımız, merkezileşmiş bir yöntemden daha hızlı bir yöntem önermek değildir. Amacımız bütünsel problemi merkezi koordinasyon biriminin bulunmadığı ya da kabul edilmediği bir durumda yerel bilgi paylaşmadan, ortak bir fayda için diğerleri ile işbirliği yaparak çözmek isteyen karar vericiler için dağıtık bir koordinasyon yapısı sunmaktır.

Doğrusal programlama problemleri için önerilen metodlar arasındaki temel fark paylaşılan bilginin niteliğidir. Dağıtık Benders Ayıştırma Yönteminde ikincil bilgi paylaşımı varken Dağıtık Dantzig-Wolfe Ayrıştırma Yönteminde birincil bilgi paylaşılır. Önerilen yöntemlerin merkezileştirme ile bulunan en iyi çözüme sonlu sayıda döngü ile yakınsadığını ispatlanmıştır. Metodların performans değerlendirmeleri yapılmıştır. Aynı zamanda karar vericiler arasındaki iletişim ağının etkileri de incelenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Problems in block angular structure may appear when there is multiple decision makers in the environment. For example, consider divisions of an organization sharing some common resources. Primal block angular structure in Figure 1.1 is the resulting structure since capacity constraints on common resources appear as the uppermost complicating constraints that links the blocks of divisions' local constraints. Alternatively, consider independent organizations having own activities engaging only for activities to reduce pollution. Then dual block angular structure in Figure 1.1 is the resulting structure since common mission activities appear as the complicating variables that link blocks of separate activities of the organizations.

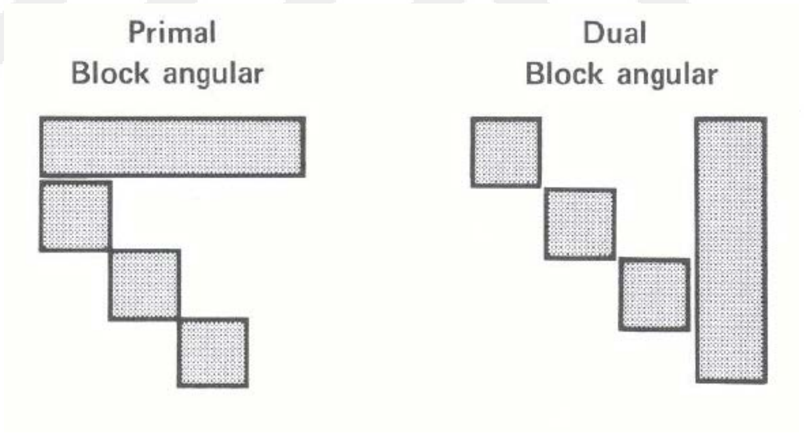


Figure 1.1. Primal and dual block angular problem structures [1].

Centralization is one approach for solving problems in block angular structure. In this case, one of the decision makers acting as a center gathers all managerial information from the others and builds the aggregate problem. The problem is solved by a standard optimization algorithm such as Simplex Method, Interior Point Algorithms or Branch and Bound Algorithm. Then the center announces the solution to the other decision makers. Centralization can be desirable if it is possible to build the aggregate model. However, building the aggregate model may not be practical from computational point of view. Moreover, decision makers may not be willing to share their own managerial information with a center.

Existing decomposition methods [2, 3, 4] present an alternative approach that exploits the block angular structure for computational efficiency. For instance, without the coupling constraints, a primal block angular problem can be partitioned into independent *subproblems* associated with each block. Instead of one large-scale optimization problem, there are many smaller, easy 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 global problem because of the violation of the coupling constraints. Thus, a center is required to solve *master problem* to combine individual solutions of subproblems to form a solution to the global problem.

Decomposition methods allow for *decentralization* to a certain extent since the master problem acts as a center having access to other decision makers' information to direct them. However, many real-world optimization problems involve decision makers that are unwilling to share their private data, but want to collaborate for solving the global problem for a mutual benefit. Consider the following examples: (i) A production planning problem arising in the supply chain of an enterprise where multiple decision makers are sharing some scarce resources to serve their own customers. The resulting problem represents primal block angular structure since common resource availability constraints link blocks associated with production requirement constraints of each product. In this case, decision makers realize their own production facilities in isolation, but they reduce the performance losses on shared resources collaboratively. (ii) A machine scheduling problem where machines having its own block of precedence constraints are tied to each other by a set of complicating constraints imposing completion of the jobs. Thus, the global problem of minimizing total completion time of the jobs is in primal block angular structure. Machines, acting as independent decision makers, should cooperate to achieve global optimal solution since any one of them has access to the local information of the others. (iii) A problem arising in logistics when two companies producing and serving the same region decide to share their vehicles to minimize global transportation cost by reducing empty vehicle movements. The resulting problem represents primal block angular structure since the restrictions on shared vehicles such as capacity constraints links the blocks of local constraints associated with the supply and demand information of companies. (iv) An energy distribution problem where some microgrids demand more energy from peers having excess energy instead of connecting to a main power grid. The resulting problem represents primal block angular structure since microgrids aim to minimize global energy loss in the system while keeping their energy demand or storage capacity private. These examples reveals the need for decentralized decomposition methods since it is impossible or not practical to solve the global problem with existing decomposition methods.

Our main motivation for this research is to develop decentralized decomposition methods for block angular linear and integer programs. We propose two methods, *Decentralized Benders Decomposition* and *Decentralized Dantzig-Wolfe Decomposition* for linear programs and *Decentralized L-Shaped Method* for integer programs. We exploit the primal block angular structure of the problem to decompose the problem into a subproblem-local master problem pair for each decision maker which we call *optimization agent* (OA). Then by allowing minimum required information sharing among the collaborative OAs through a strongly connected communication network, we solve the overall problem by reaching a global optimal solution without need of a central coordination unit. We prove that the proposed methods can reach a global optimal solution in a finite number of iterations.

1.1. Contributions

In this section, we summarize the contributions of our research.

Development of Decentralized Benders Decomposition Algorithm for BALP: We introduce a decentralized algorithm based on Benders Decomposition. Although Benders Decomposition can be best applied to the dual block angular structures, our algorithm exploits primal block angularity in problem structure. Main goal of the algorithm is solving BALP without a central coordination unit. Decentralized Benders Decomposition Algorithm allows multiple decision makers of same hierarchy to solve the linear optimization problem while sharing dual information only among themselves. We prove the convergence to the global optimal solution in a finite number of iterations.

Development of Decentralized Dantzig-Wolfe Decomposition Algorithm for BALP: We introduce a decentralized algorithm based on Dantzig-Wolfe Decomposition as an alternative for Decentralized Benders Decomposition Algorithm. It allows multiple decision makers of same hierarchy to solve the linear optimization problem by sharing primal information as a difference. We present proof of convergence to the global optimal solution in a finite number of iterations.

Development of Decentralized Integer L-shaped Method for BAIP: Our research also includes an extension to block angular integer programs based on Integer L-shaped method [5].

The strengths of our algorithms are listed below:

- Allowing multiple decision makers communicating among themselves through a communication network to solve problems in block angular structure collaboratively without need of a center.
- Being blind to the communication network among decision makers. Proposed methods solve problems in block angular structure to optimality in case of decision makers are tied to each other through any strongly connected communication network. The algorithm does not need to know the structure of the communication network.
- Design of the methods allow for parallelization and distributed computing as a future research direction.

1.2. Structure Of The Thesis

This thesis is organized as follows. In Chapter 2, we present an overview of the common approaches. Then, we review the literature on decentralized decision making specifically for both linear and integer problems in block angular structure.

In Chapter 3, we present preliminary concepts to lay a groundwork for describing the proposed methods. First we formally define the problem that we address. We give definitions and notations for the communication network among the decision makers. We state the main algorithm for the proposed methods and give sketch of the convergence proof.

In Chapter 4, first we apply Classical Benders Decomposition as a solution approach. Then we present the proposed Decentralized Benders Decomposition method for block angular linear problems and we give the convergence proof.

In Chapter 5, we present the proposed Decentralized Dantzig-Wolfe Decomposition method for linear programs having block angular structure and prove its convergence to optimality.

In Chapter 6, we give an extension of the proposed method to integer programming problems in block angular structure. We describe the algorithm with differences in feasibility and optimality cut generation strategy, then we prove the convergence of the method.

In Chapter 7, we test the computational performance of Decentralized Benders Decomposition and Decentralized Dantzig-Wolfe Decomposition on two groups of test instances: randomly generated linear block angular problems and linear Multicommodity Network Flow problems. We present experimental results with a discussion. Also, we present experimental results for Decentralized L-shaped Method on small test instances and discuss preliminary results. We test our algorithms by using Star, Ring and Mesh network topologies and represent the effect of the topology of the communication network on the performance of the algorithms.

Finally, Chapter 8 concludes the thesis with a summary of the content and possible future research directions.

2. LITERATURE REVIEW

This chapter reviews common approaches for solving large-scale optimization problems. Then, we present the literature survey on decentralized decision making specifically for both linear and integer problems in block angular structure. We also discuss issues regarding distributed decision making where applicable. Finally, we present the definitions and notations for the communication network.

2.1. Common Approaches for Large-Scale Optimization

2.1.1. Column Generation

Column generation method is introduced by Dantzig and Wolfe [3] in the context of solving large-scale linear problems by decomposition methods. Column generation method can be best applied to the problems that has extremely large number of columns (i.e., variables). Including all of the columns for building the aggregate problem and solving it as a single large problem may be either impossible or computationally not practical. Hence, the key idea of column generation method is starting with a subset of columns at the beginning and then adding most profitable column when it is needed.

Consider the following problem which is called *master problem*(CG-MP):

CG-MP

$$\text{Minimize} \sum_{j=1}^{N} c_j x_j \tag{2.1a}$$

$$\text{subject to} \sum_{j=1}^{N} a_{ij} x_j = b_i \quad \forall i = 1, 2, \ldots, M \tag{2.1b}$$

$$x_j \geq 0 \quad \forall j = 1, 2, \ldots, N \tag{2.1c}$$

where x_j denotes decision variables for $j \in \{1, 2, 3, ..., N\}$ and we assume that N is very large. Building and solving MP directly may be inappropriate. Thus, instead of solving one large problem, column generation method formulates *restricted master problem* (CG-RMP) with a subset, $\bar{N} \subset N$, of variables. The resulting problem is:

CG-RMP

$$\text{Minimize} \sum_{j=1}^{\bar{N}} c_j x_j \tag{2.2a}$$

$$\text{subject to} \sum_{j=1}^{\bar{N}} a_{ij} x_j = b_i \quad \forall i = 1, 2, \ldots, M \tag{2.2b}$$

$$x_j \geq 0 \quad \forall j = 1, 2, \ldots, \bar{N} \tag{2.2c}$$

Let $\pi^* = \{\pi_1, \pi_2, \ldots, \pi_M\}$ be the optimal dual variables for (2.2). An optimal solution of CG-RMP is rarely optimal for CG-MP also. There may be an improving column that has not been considered yet. For a minimization problem, variable $\hat{x}_j$ having minimum *negative reduced cost*, $\bar{r}$, is the most promising variable to generate a column. The reduced cost of a variable with respect to current dual variables is given as follows:

$$r_j = c_j - \sum_{i=1}^{M} \pi_i a_{ij} \tag{2.3}$$

To find the variable having minimum negative reduced cost among a large set of alternatives, the following *subproblem* (CG-SP) is solved:

CG-SP

$$\bar{r} = \min_{1 \leq j \leq N} c_j - \sum_{i=1}^{M} \pi_i a_{ij} \tag{2.4}$$

(2.4) is also called the *pricing problem.* If $\bar{r} < 0$, the variable found and its coefficient column is added to the CG-RMP. Then CG-RMP is re-solved and the entire process is repeated until there is no improving variable that can be added to CG-RMP.

Column generation method avoids solving CG-MP directly. Instead, it solves CG-RMP and CG-SP in an alternating sequence. The main advantage of column generation method is obtaining an optimal solution before many columns are added to CG-RMP. Also in some cases structural properties ensure solving SP effectively. For further reading, refer to [6, 7].

2.1.2. Cutting Plane Algorithm

A large-scale linear problem may consists of extremely large number of constraints. Solving such a problem directly may be impractical. *Cutting Plane Algorithm* is introduced as an appropriate approach. The basic idea is defining a less constrained initial problem and then adding the other constraints when they are needed.

Consider the following aggregate problem consisting of full set of constraints. It is called *master problem* (CP-MP). Note that, CP-MP is the dual of (2.1). Since N is assumed to be a very large number, the number of constraints is very large.

CP-MP

$$\text{Maximize} \sum_{i=1}^{M} b_i \pi_i \tag{2.5a}$$

$$\text{subject to} \sum_{i=1}^{M} a_{ij} \pi_i \leq c_j \quad \forall j = 1, 2, \ldots, N \tag{2.5b}$$

Instead of solving CP-MP as a whole, cutting plane algorithm defines *relaxed master problem* consisting a subset $\bar{N} \subseteq N$ of constraints:

CP-RMP

$$\text{Maximize} \sum_{i=1}^{M} b_i \pi_i \tag{2.6a}$$

$$\text{subject to} \sum_{i=1}^{M} a_{ij} \pi_i \leq c_j \quad \forall j = 1, 2, \ldots, \bar{N} \tag{2.6b}$$

Cutting plane algorithm starts with solving CP-RMP, then checks whether the current optimal solution is feasible for CP-MP. If it is, then that solution is an optimal solution for CP-MP also. Otherwise, there is at least one constraint in the set $N \setminus \bar{N}$ that is violated. Cutting plane algorithm finds the most violated constraint by solving a *subproblem*. Assume that π_i^k is an optimal solution of CP-RMP at iteration k. Then the subproblem is defined as follows:

CP-SP

$$\min_{j \in N \setminus \bar{N}} c_j - \sum_{i=1}^{M} a_{ij} \pi_i^k \tag{2.7}$$

If (2.7) is nonnegative for all constraints in the set $N \setminus \bar{N}$, then π_i^k is an optimal solution for CP-MP. Otherwise, the constraint that satisfies the inequality $c_j > \sum_{i=1}^{M} a_{ij} \pi_i^k$ is added to CP-RMP. The updated CP-RMP is re-solved and the entire process is repeated until there is no new constraint that can be added to CP-RMP.

Cutting plane algorithm decomposes aggregate problem into CP-RMP and CP-SP and solves them alternately. The main advantage of cutting plane algorithm is obtaining an optimal solution before many constraints are added to CP-RMP. For further reading, refer to [8].

2.1.3. Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition is introduced by Dantzig and Wolfe [3] to solve large scale linear programs especially in primal block angular structure. To take the advantage of block angularity, Dantzig-Wolfe decomposition relaxes the complicating constraints from BALP and considers them as master problem. Then the remaining blocks of constraints constitute the subproblems. The master problem can be reformulated in terms of extreme points and extreme rays of the subproblems. The master problem has fewer constraints than the aggregate problem, however it may be still intractable because of the large number of columns. Hence Dantzig-Wolfe decomposition starts with a restricted master problem instead of a full master problem. The subproblems generate the remaining columns of full master problem as they are needed.

Consider the following primal BALP:

BALP

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{2.8a}$$

$$\text{subject to} \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ijk}x_{ij} = r_k \quad \forall k = 1,2,\ldots,K \tag{2.8b}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \quad \forall i = 1,2,\ldots,N \tag{2.8c}$$

$$x_{ij} \geq 0 \quad \forall i = 1,2,\ldots,N \quad \forall j = 1,2,...,n_i \tag{2.8d}$$

where (2.8b) are the *complicating constraints* whereas (2.8c) are the blocks of *local constraints*. Let's define the polyhedron of each block of local constraints by the set $X_i = \{x_{ij} | \sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i, x_{ij} \geq 0 \quad \forall j = 1,2,...,n_i\}$ for $i = 1,2,\ldots,N$. One can rewrite (2.8) as follows:

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{2.9a}$$

$$\text{subject to} \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ijk}x_{ij} = r_k \quad \forall k = 1,2,\ldots,m \tag{2.9b}$$

$$x_{ij} \in X_i \quad \forall i = 1,2,\ldots,N \tag{2.9c}$$

Note that (2.9) contains complicating constraints only and local constraints are considered implicitly. Dantzig-Wolfe decomposition applies *Minkowski's Representation Theorem* [7, 9] here which is stated below:

Theorem 1. If P is the polyhedron $P = \{x|Ax \geq b\}$, then any $x \in P$ is the sum of a convex combination of extreme points of P and a nonnegative linear combination of extreme rays of P.

Using Theorem 1, any $x_{ij} \in X_i$ can be written as the following:

$$x_{ij} = \sum_{q \in Q_i} \lambda_i^q v_i^q + \sum_{s \in S_i} \mu_i^s d_i^s \tag{2.10a}$$

$$\sum_{q \in Q_i} \lambda_i^q = 1 \tag{2.10b}$$

$$\lambda_i^q \geq 0, \quad \mu_i^s \geq 0 \quad \forall v_i^q \in Q_i, \quad \forall d_i^s \in S_i \tag{2.10c}$$

where Q_i is the set of extreme points and S_i is the set of extreme rays of polyhedron X_i. By substituting x_{ij} variables, (2.9) can be equivalently reformulated as follows:

DW-FMP

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{n_i} c_{ij} (\sum_{q \in Q_i} \lambda_i^q v_i^q + \sum_{s \in S_i} \mu_i^s d_i^s) \tag{2.11a}$$

$$\text{subject to} \sum_{i=1}^{N} \sum_{j=1}^{n_i} a_{ijk} (\sum_{q \in Q_i} \lambda_i^q v_i^q + \sum_{s \in S_i} \mu_i^s d_i^s) = r_k \quad \forall k = 1, 2, \ldots, K \tag{2.11b}$$

$$\sum_{q \in Q_i} \lambda_i^q = 1 \quad \forall i = 1, 2, \ldots, N \tag{2.11c}$$

$$\lambda_i^q \geq 0, \quad \mu_i^s \geq 0 \quad \forall v_i^q \in Q_i, \quad \forall d_i^s \in S_i \quad \forall i = 1, 2, \ldots, N \tag{2.11d}$$

(2.11) is called *Dantzig-Wolfe full master problem*, (DW-FMP). It has many variables since each polyhedron X_i may have great number of extreme points and extreme rays. Hence Dantzig-Wolfe decomposition applies column generation idea here. It starts with solving *restricted master problem* (DW-RMP) given in (2.12) that has a subset of extreme points $\bar{Q}_i \subset Q_i$ and extreme rays $\bar{S}_i \subset S_i$.

DW-RMP

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}(\sum_{q\in\bar{Q}_i} \lambda_i^q v_i^q + \sum_{s\in\bar{S}_i} \mu_i^s d_i^s) \tag{2.12a}$$

$$\text{subject to} \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ijk}(\sum_{q\in\bar{Q}_i} \lambda_i^q v_i^q + \sum_{s\in\bar{S}_i} \mu_i^s d_i^s) = r_k \quad \forall k = 1,2,\ldots,K \tag{2.12b}$$

$$\sum_{q\in\bar{Q}_i} \lambda_i^q = 1 \quad \forall i = 1,2,\ldots,N \tag{2.12c}$$

$$\lambda_i^q \geq 0, \quad \mu_i^s \geq 0 \quad \forall v_i^q \in \bar{Q}_i, \quad \forall d_i^s \in \bar{S}_i \quad \forall i = 1,2,\ldots,N \tag{2.12d}$$

Dantzig-Wolfe decomposition solves (2.12) and obtains dual variables π_k and α_i associated with the constraint sets (2.12b) and (2.12c), respectively. In order to check whether the current solution can be improved, Dantzig-Wolfe decomposition searches for a new variable which has negative reduced cost. Then, the reduced cost, r_i^q, of variable v_i^q can be calculated as the following:

$$r_i^q = \sum_{i=1}^{N}\sum_{j=1}^{n_i}(c_{ij} - a_{ijk}\pi_k)v_i^q - \alpha_i \tag{2.13}$$

Similarly, the reduced cost r_i^s of variable μ_i^s can be calculated as the following:

$$r_i^s = \sum_{i=1}^{N}\sum_{j=1}^{n_i}(c_{ij} - a_{ijk}\pi_k)\mu_i^s \tag{2.14}$$

To check whether there exists a variable with negative reduced cost in $Q_i \setminus \bar{Q}_i$, a *subproblem* (DW-SP1) given in (2.15) is solved:

DW-SP1

$$\min_{q\in Q_i\setminus\bar{Q}_i} \sum_{i=1}^{N}\sum_{j=1}^{n_i}(c_{ij} - a_{ijk}\pi_k)v_i^q - \alpha_i \tag{2.15}$$

Similarly, to verify whether there exists a variable with negative reduced cost in $S_i \setminus \bar{S}_i$, a *subproblem* (DW-SP2) given in (2.16) is solved:

DW-SP2

$$\min_{s \in S_i \setminus \bar{S}_i} \sum_{i=1}^{N} \sum_{j=1}^{n_i} (c_{ij} - a_{ijk}\pi_k)\mu_i^s \tag{2.16}$$

If (2.15) has a negative objective function value, then there exists an improving variable associated with an extreme point. On the other hand, if (2.16) is unbounded, then there exists an improving variable associated with an extreme ray. In both cases, improving variable is added to DW-RMP with its corresponding column. DW-RMP is resolved with added columns and this process is repeated until there is no improving variable.

The details of the Dantzig-Wolfe decomposition can be found at any standard textbook such as [9, 7]. Also a summary on computational efficiency of Dantzig-Wolfe decomposition especially for large-scale BALP is given in [10]. According to this, Dantzig-Wolfe decomposition has slow convergence when it is compared to the methods that solve one single problem. However, one of its advantages is applying decomposition approach when the aggregate problem is too large to solve at a time.

Another advantage of Dantzig-Wolfe decomposition is from an economic point of view. In an organizational setting, subproblems can be interpreted as divisions whose goal is optimizing their own objective while sharing some common resources. Master problem acts as a center who coordinates the subproblems through setting (dual) prices for common resources. Given the prices, divisions announce their proposal of common resource consumption to center. The center checks whether there is a set of proposals improving the overall objective. If it finds one, then updates prices and process continues. Otherwise, it announces optimal (dual) prices on common resources. As a result, there is two levels of decision making in Dantzig-Wolfe decomposition; a master problem acting as the center and subproblems representing divisions. The center knows all corporate constraints while divisions only know their own constraints. To reach overall optimality, individual proposals of divisions are required to be coordinated by a center.

2.1.4. Benders Decomposition

Benders Decomposition is introduced by Benders [2] especially to solve mixed integer programming problems. However it has applications in a broader class of problems [11, 12].A formal description on Benders Decomposition with some extensions can be found in [13]. In this section, we will describe how Benders Decomposition is applied to large-scale linear programs in dual block angular structure.

The main feature of Benders decomposition is having *complicating variables* which link blocks (See Figure 1.1). Hence the main idea of Benders decomposition is splitting the aggregate problem with respect to its variables. By fixing the complicating variables, it is easier to solve independent subproblems. Also a relaxed master problem is constructed in order to decide the values for complicating variables. At each iteration subproblems generate new constraints which are added to the relaxed master problem for determining new values for complicating variables.

Benders decomposition also defined as Dantzig-Wolfe decomposition applied to the dual problem. Equivalence of the methods is shown in [14]. Dual of BALP can be formulated as follows by associating π_k and α_i variables with constraints (2.8b) and (2.8c), respectively.

Dual-BALP

$$\text{Maximize } \sum_{k=1}^{K} r_k \pi_k + \sum_{i=1}^{N} \alpha_i l_i \tag{2.17a}$$

$$\text{subject to } \sum_{k=1}^{K} a_{ijk}\pi_k + b_{ij}\alpha_i \leq c_{ij} \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i \tag{2.17b}$$

$$\pi_k \text{ } \textit{unrestricted} \quad \forall k = 1, 2, \ldots, K \tag{2.17c}$$

$$\alpha_i \text{ } \textit{unrestricted} \quad \forall i = 1, 2, \ldots, N \tag{2.17d}$$

In this formulation, π_k variables represents complicating variables. Dual-BALP can be formulated in terms of the complicating variables as follows:

$$\text{Maximize } \sum_{k=1}^{K} r_k \pi_k + \sum_{i=1}^{N} z_i(\pi_k) \tag{2.18a}$$

$$\pi_k \text{ } \textit{unrestricted} \quad \forall k = 1, 2, \ldots, K \tag{2.18b}$$

where $z_i(\pi_k)$ is defined as optimal objective value of the following problem:

$$z_i(\pi_k) = \text{Maximize } \sum_{i=1}^{N} \alpha_i l_i \tag{2.19a}$$

$$\text{subject to } b_{ij}\alpha_i \leq c_{ij} - \sum_{k=1}^{K} a_{ijk}\pi_k \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i \tag{2.19b}$$

$$\alpha_i \text{ } \textit{unrestricted} \quad \forall i = 1, 2, \ldots, N \tag{2.19c}$$

Associating the dual variables x_{ij} with constraints (2.19b), the dual of (2.19) is as follows for given $\hat{\pi}_k$ variables:

$$\text{Minimize } \sum_{i=1}^{N} \sum_{j=1}^{n_i} (c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi}_k) x_{ij} \tag{2.20a}$$

$$\text{subject to} \sum_{j=1}^{n_i} b_{ij} x_{ij} = l_i \quad \forall i = 1, 2, \ldots, N \tag{2.20b}$$

$$x_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i \tag{2.20c}$$

Notice that (2.20) can be separated into N independent smaller problems which are called as *Benders subproblem*. The resulting i^{th} subproblem is:

$$\text{Minimize } \sum_{j=1}^{n_i}(c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi}_k)x_{ij} \tag{2.21a}$$

$$\text{subject to} \sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \tag{2.21b}$$

$$x_{ij} \geq 0 \quad \forall j = 1, 2, ..., n_i \tag{2.21c}$$

Note that the feasible region of (2.21) does not depend on $\hat{\pi}_k$ values. Assuming a non-empty feasible region, one can enumerate all extreme points and extreme rays of the feasible region. Let us define P_i and R_i as sets of extreme points and extreme rays of the i^{th} problem, respectively. If (2.21) is unbounded for $\hat{\pi}_k$, then there exists an extreme ray x_{ij}^r in R_i such that

$$(c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi}_k)x_{ij}^r < 0 \tag{2.22}$$

To exclude this extreme ray solution, $(c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi}_k)x_{ij}^r \geq 0$ should be satisfied. On the other hand, if (2.21) has finite optimal solution for $\hat{\pi}_k$, then there exists an extreme point x_{ij}^p in P_i that minimizes the objective function value, that we call z_i^* where

$$z_i^* \leq (c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi}_k)x_{ij}^p \tag{2.23}$$

In this case, the aim of Benders Decomposition is to find that extreme point solution. Based on (2.22) and (2.23), (2.21) can be equivalently reformulated as follows:

$$\text{Maximize } z_i \tag{2.24a}$$

$$\text{subject to } (c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi}_k)x_{ij}^r \geq 0 \quad \forall r \in R_i \tag{2.24b}$$

$$(c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi}_k)x_{ij}^p \geq z_i \quad \forall p \in P_i \tag{2.24c}$$

$$z_i \textit{ unrestricted} \tag{2.24d}$$

Dual-BALP can be reformulated in terms of π_k and α_i variables by replacing $z_i(\pi_k)$ in (2.18) with (2.24):

$$\text{Maximize } \sum_{k=1}^{K} r_k\pi_k + \sum_{i=1}^{N} z_i \tag{2.25a}$$

$$\text{subject to } (c_{ij} - \sum_{k=1}^{K} a_{ijk}\pi_k)x_{ij}^r \geq 0 \quad \forall r \in R_i \quad \forall i = 1, 2, \ldots, N \tag{2.25b}$$

$$(c_{ij} - \sum_{k=1}^{K} a_{ijk}\pi_k)x_{ij}^p \geq z_i \quad \forall p \in P_i \quad \forall i = 1, 2, \ldots, N \tag{2.25c}$$

$$\pi_k \textit{ unrestricted} \quad \forall k = 1, 2, \ldots, K \tag{2.25d}$$

$$z_i \textit{ unrestricted} \quad \forall i = 1, 2, \ldots, N \tag{2.25e}$$

(2.25) is called *Benders master problem.* In general, it may have an exponential number of constraints since there may be exponentially many extreme points and extreme rays for each subproblem. Hence Benders Decomposition starts with solving a *relaxed master problem* which contains a subset of constraints. The solution of restricted master problem includes $\hat{\pi}_k$ values to announce to the subproblems and a candidate solution value z_i on subproblem's objective function value, $z_i(\hat{\pi}_k)$. If $z_i = z_i(\hat{\pi}_k)$ for each i, then algorithm terminates. Otherwise, if dual subproblem is unbounded with respect to $\hat{\pi}_k$ values, a *feasibility cut* represented as (2.24b) is generated and added to the restricted master problem. If the subproblem has an optimal solution, then an *optimality cut* represented as (2.24c) is generated and added to the restricted master problem. Restricted master problem is resolved to get new $\hat{\pi}_k$ and z_i values. This process is repeated until termination. The convergence of the algorithm in finite number of iterations follows since there are finitely many extreme points and extreme rays, and a *new* feasibility or optimality cut is generated at each iteration [2].

Notice that at any iteration, Benders relaxed master problem gives an upper bound for objective function value of Dual-BALP since it consists of only a subset of constraints associated with extreme rays or extreme points. Also notice that, Benders subproblem is a restriction on Dual-BALP, hence objective function value of (2.19) in addition with the constant term $\sum_{k=1}^{K} r_k \hat{\pi}_k$ gives a lower bound for objective function value of Dual-BALP. The advantage of this feature is allowing termination before reaching overall optimality if lower and upper bounds are close enough.

From an economic point of view, in Benders Decomposition master problem acting as a center allocates common resources to divisional subproblems by fixing values of complicating variables. Given the allocations, subproblems solve their problem and each makes an offer for common resource prices. Master problem collects the prices and updates the allocations with respect to them. This process continues until center and divisions reach overall optimality. Although Benders decomposition allows decentralization to a certain extent, there is a need for a center to direct the divisions.

2.1.5. Lagrangian Relaxation

Lagrangian relaxation method is introduced by Held and Karp [15] in their work on relaxation algorithms for Traveling Salesman Problem. Afterwards it is applied especially to integer programming problems and nonlinear programming problems [16, 17, 18]. In this research, we focus on how Lagrangian relaxation can be applied to large-scale linear programs in primal block angular structure.

Without complicating constraints, primal BALP can be broken into smaller independent problems. Hence the key idea of Lagrangian relaxation is omitting complicating constraints from the constraint set by placing them in the objective function by multiplying with a penalty term. These multipliers are called *Lagrangian duals* since they are dual variables associated with the complicating constraints. This process is referred to *dualizing* the complicating constraints and the resulting problem is called *Lagrangian Function* (LF). If BALP is a minimization problem, Lagrangian duals that maximizes LF yield an optimal objective function value that is equal to optimal objective function value of the original problem. However, this holds for the convex case. For integer programming or nonlinear programming problems, there may be difference between two objective function values which is called *duality gap*.

We consider primal BALP in (2.8) to describe Lagrangian relaxation method mathematically. Here (2.8b) are the complicating constraints which are dualized. Lagrangian function is as follows for associated Lagrangian duals, π_k:

LF

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} + \sum_{k=1}^{K}\pi_k(r_k - \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ijk}x_{ij}) \tag{2.26a}$$

$$\text{subject to} \sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \quad \forall i = 1,2,\ldots,N \tag{2.26b}$$

$$x_{ij} \geq 0 \quad \forall i = 1,2,\ldots,N \quad \forall j = 1,2,...,n_i \tag{2.26c}$$

With appropriate arrangement in objective function, (2.26) can be rewritten equivalently as

$$\sum_{k=1}^{K}\pi_k r_k + \text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i}(c_{ij} - \sum_{k=1}^{K}\pi_k a_{ijk})x_{ij} \tag{2.27}$$

Note that LF is a relaxation, hence it provides a lower bound on BALP which depends on the Lagrangian duals. To obtain the tightest lower bound, one can solve LF as a maximization problem over all Lagrangian duals since BALP is a minimization problem. This yields the following *Lagrangian dual problem* (LDP) :

LDP

$$\underset{\pi_k}{Max}\, L(\pi_k) \tag{2.28}$$

where $L(\pi_k)$ is defined as

$$\sum_{k=1}^{K} \pi_k r_k + \text{Minimize } \sum_{i=1}^{N}\sum_{j=1}^{n_i}(c_{ij} - \sum_{k=1}^{K} \pi_k a_{ijk})x_{ij} \tag{2.29a}$$

$$\text{subject to } \sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \quad \forall i = 1, 2, \ldots, N \tag{2.29b}$$

$$x_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i \tag{2.29c}$$

There are various solution methods for solving LDP [19]. Among them Subgradient Optimization, Column Generation and Cutting Plane methods are the leading ones. Here, we describe the cutting plane method for solving LDP. Thus, we refer to (2.28) as *Lagrangian master problem.* Note that, without the term $\sum_{k=1}^{K} \pi_k r_k$, it can be separated with respect to i. Hence $L(\pi_k) = \sum_{k=1}^{K} \pi_k r_k + \sum_{i=1}^{N} z_i(\pi_k)$, where $z_i(\pi_k)$ is

$$z_i(\pi_k) = \text{Minimize } \sum_{j=1}^{n_i}(c_{ij} - \sum_{k=1}^{K} \hat{\pi_k} a_{ijk})x_{ij} \tag{2.30a}$$

$$\text{subject to } \sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \tag{2.30b}$$

$$x_{ij} \geq 0 \quad \forall j = 1, 2, ..., n_i \tag{2.30c}$$

The resulting problem, (2.30), is called the *Lagrangian subproblem.* Note that the feasible region of the Lagrangian subproblem is independent of π_k variables. Assuming a non-empty feasible region, the extreme rays and the extreme points can be enumerated. Let us define P_i as the set of extreme points of i^{th} subproblem that consists of $\{x_{ij}^p\}$ for $p \in P_i$. Similarly, let R_i be the set of the extreme rays that consists of $\{x_{ij}^r\}$ for

$r \in R_i$. Then (2.30) can be equivalently formulated for given $\hat{\pi_k}$ as the following:

$$\begin{aligned}
&\text{Maximize } \sum_{i=1}^{N} z_i && \text{(2.31a)}\\
&\text{subject to } (c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi_k})x_{ij}^{r} \geq 0 \quad \forall r \in R_i \quad \forall i = 1,2,\ldots,N && \text{(2.31b)}\\
&(c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi_k})x_{ij}^{p} \geq z_i \quad \forall p \in P_i \quad \forall i = 1,2,\ldots,N && \text{(2.31c)}\\
&z_i \textit{ unrestricted} \quad \forall i = 1,2,\ldots,N && \text{(2.31d)}
\end{aligned}$$

If (2.31) is bounded, then it generates an optimality cut as the following:

$$z_i \leq (c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi_k})x_{ij}^{p} \tag{2.32}$$

In case of unboundedness, (2.30) generates a feasibility cut as the following:

$$(c_{ij} - \sum_{k=1}^{K} a_{ijk}\hat{\pi_k})x_{ij}^{r} \leq 0 \tag{2.33}$$

Hence, full master problem can be formulated in terms of extreme rays and extreme points as the following:

$$\begin{aligned}
&\text{Maximize } \sum_{k=1}^{K} r_k\pi_k + \sum_{i=1}^{N} z_i && \text{(2.34a)}\\
&\text{subject to } (c_{ij} - \sum_{k=1}^{K} a_{ijk}\pi_k)x_{ij}^{r} \geq 0 \quad \forall r \in R_i \quad \forall i = 1,2,\ldots,N && \text{(2.34b)}\\
&(c_{ij} - \sum_{k=1}^{K} a_{ijk}\pi_k)x_{ij}^{p} \geq z_i \quad \forall p \in P_i \quad \forall i = 1,2,\ldots,N && \text{(2.34c)}\\
&\pi_k \textit{ unrestricted} \quad \forall k = 1,2,\ldots,K && \text{(2.34d)}\\
&z_i \textit{ unrestricted} \quad \forall i = 1,2,\ldots,N && \text{(2.34e)}
\end{aligned}$$

Lagrangian relaxation starts with relaxed master problem for computational convenience and the algorithm alternates between solving master problem and subproblems. Algorithm terminates when value of $L(\pi_k)$ at iteration t equals to $\sum_{i=1}^{N} z_i$ at iteration $t+1$. For an application of subgradient optimization as solution approach for Lagrangian dual function, refer [20].

2.1.6. Rosen's Primal Partitioning Algorithm

Rosen proposed a Primal Partitioning Algorithm [4] for large-scale linear programming problems in dual block angular structure. Consider the following problem:

$$\text{Maximize } b_0^t y + \sum_{i=1}^{l} b_i^t x_i \tag{2.35a}$$

$$\text{subject to } D_i^t y + A_i^t x_i \leq c_i \quad \forall i = 1, 2, \ldots, l \tag{2.35b}$$

$$x_i \quad \textit{unrestricted} \quad \forall i = 1, 2, \ldots, l \tag{2.35c}$$

$$y \quad \textit{unrestricted} \tag{2.35d}$$

where t denotes the transpose operator. Then the primal block angular problem is as follows:

$$\text{Minimize } \sum_{i=1}^{l} c_i^t u_i \tag{2.36a}$$

$$\text{subject to } \sum_{i=1}^{l} D_i u_i = b_0 \tag{2.36b}$$

$$A_i u_i = b_i \quad \forall i = 1, 2, \ldots, l \tag{2.36c}$$

$$u_i \geq 0 \tag{2.36d}$$

In (2.35), y variables are the complicating variables since they link the blocks of local constraints. Hence, by fixing the values of the complicating variables, (2.35) can be decomposed into l subproblems. Assume $\hat{y}$ is feasible for (2.35). Then i^{th} subproblem in the primal form can be formulated as the following:

$$\text{Minimize } (c_i^t - \hat{y}^t D_i)u_i \tag{2.37a}$$

$$\text{subject to } A_i u_i = b_i \tag{2.37b}$$

$$u_i \geq 0 \tag{2.37c}$$

Then each subproblem i can be solved independently. Let u_i^* be an optimal solution for (2.37) with corresponding basis matrix $\underline{A_i}$. Then assume that the non-basic columns of A_i constitute a matrix B_i. Assume a similar partition for the cost vector c_i into $\underline{c_i}$ and e_i, also for D_i into $\underline{D_i}$ and E_i for basic and non-basic columns respectively. Then the following equation can be derived from (2.36c) for the basic variables, $\underline{u_i}^*$, where non-basic variables are denoted by v_i:

$$\underline{u_i^*} = \underline{A_i}^{-1} b_i - \underline{A_i}^{-1} B_i v_i \tag{2.38}$$

Eliminating the basic variables in (2.36) by substituting (2.38) in the objective function and complicating constraints results in the following problem in terms of the non-basic variables:

$$\text{Minimize } \sum_{i=1}^{l} {q_i}^t v_i \tag{2.39a}$$

$$\text{subject to } R_i v_i = b \tag{2.39b}$$

$$v_i \geq 0 \tag{2.39c}$$

(2.39) is called the master problem where $q_i = e_i - (\underline{A_i^{-1} B_i})^t \underline{c_i}$, $R_i = E_i - \underline{D_i} A_i^{-1} B_i$ and $b = b_0 - \sum_{i=1}^{l} \underline{D_i} A_i^{-1} b_i$. Let v_i^* be optimal solution for (2.39). Substituting v_i^* in (2.38) gives the new values for $\underline{u_i}^{new}$. The blocks for which $\underline{u_i}^{new} \geq 0$ are called the optimal blocks. If all blocks are optimal, then the overall optimality is reached. Otherwise, if $\underline{u_i}^{new}$ has negative elements for some i, then for block i, a positive component of v_i^* is changed by a negative component of $\underline{u_i}^{new}$ by performing a set of pivot operations. This gives a new basis matrix for i^{th} subproblem. Then a new master problem is formulated with the new basis matrices for each non-optimal blocks to start the next iteration. The algorithm terminates with a finite optimal solution in finite number of iterations if an overall optimal solution exists.

2.1.7. Kornai-Lipták Method

Kornai-Lipták method is suggested by the authors [21] to prepare the macroeconomic plans of Hungarian National Planning Bureau. The problem of interest is a single, large-scale problem that can be decomposed into mutually independent subproblems solved by *sectors* coordinated by the *center* which allocates the resources. The idea is transforming the problem into a *two-level problem* at first where *central problem* develops an allocation pattern to maximize the sum of the maximal yields of *sector problems*. Then the two-level problem is transformed into a *polyhedral game*.

Consider the following problem:

$$\text{Maximize } c^t x \tag{2.40a}$$

$$\text{subject to } Ax \leq b \tag{2.40b}$$

$$x \geq 0 \tag{2.40c}$$

Here, small letters denote vectors while capital letters denote matrices of proper size. (^{t}) notation is used for transposition. Then the dual problem is:

$$\text{Minimize } y^t b \tag{2.41a}$$

$$\text{subject to } y^t A \geq c^t \tag{2.41b}$$

$$y \geq 0 \tag{2.41c}$$

From duality principle, the optimum value, ϕ, of both problems must be equal. In other words, $\phi = c^t x^* = y^{t*} b$, where x^* and y^{t*} denote optimal solutions of (2.40) and (2.41), respectively. If the sum of the vectors $u_1, u_2, ...u_n$ is b, then the vector $u = [u_1, u_2, \dots, u_n]$ is called *central program* and (2.40) can be decomposed into n independently solvable sector problems. Hence i^{th} sector problem is:

$$\text{Maximize } c_i^t x_i \tag{2.42a}$$

$$\text{subject to } A_i x_i \leq u_i \quad \forall i = 1, 2, \dots, n \tag{2.42b}$$

$$x_i \geq 0 \quad \forall i = 1, 2, \dots, n \tag{2.42c}$$

with dual problem

$$\text{Minimize } y_i^t b_i \tag{2.43a}$$

$$\text{subject to } y_i^t A_i \geq c_i^t \quad \forall i = 1, 2, \dots, n \tag{2.43b}$$

$$y_i \geq 0 \quad \forall i = 1, 2, \dots, n \tag{2.43c}$$

In [21], authors state the conditions for feasibility of all sector problems. Thus, for any feasible central program $u' = [u_1', u_2', \dots, u_n']$, the i^{th} sector optima is defined as

$$\varphi_i(u_i') = \underset{x_i \in X_i(u_i')}{max} \; c_i^t x_i = \underset{y_i \in Y_i(u_i')}{min} \; y_i^t u_i' \tag{2.44}$$

where $X_i(u_i') = \{x_i \mid A_i x_i \leq u_i', x_i \geq 0\}$ and $Y_i(u_i') = \{y_i \mid y_i^t A_i \geq c_i^t, y_i \geq 0\}$. Hence overall optimum under u' is:

$$\varphi(u') = \varphi_1(u_1') + \varphi_2(u_2') + \ldots + \varphi_n(u_n') \tag{2.45}$$

The authors give a proof for the equivalence of $\phi = \max_u \varphi(u)$ [21]. Then, the following equality holds where $v^t = [y_1, y_2, \dots, y_n]$ denotes the vector of shadow prices.

$$\varphi(u) = \sum_{i=1}^{n} \varphi_i(u_i) = \sum_{i=1}^{n} (\min_{y_i \in Y_i} y_i^t u_i) = \min_{y_i \in Y_i} \sum_{i=1}^{n} y_i^t u_i = \min v^t u \tag{2.46}$$

Hence one can conclude the following equation:

$$\phi = \max_{u \in U} \min_{v \in V} v^t u \tag{2.47}$$

(2.47) defines a polyhedral game where U is the set of maximizing strategies while V is the set of minimizing strategies. While maximizing player is the center who distributes the resources among the sectors in an optimal way, the minimizing player is the sectors who try to minimize the prices they should pay for the allocated resources. At each iteration, a lower and an upper bound is obtained for the payoff function, $v^t u$. Convergence to overall optimum in finite number of iterations is not guaranteed. Instead, an approximate solution is obtained under certain regularity conditions. Ten Kate [22] proposes a similar decomposition technique with improvements on convergence issues.

2.1.8. Primal-Dual Decomposition

Decomposition methods can be classified into *primal decomposition* and *dual decomposition*. While the original problem is decomposed with respect to variables in primal decomposition, decomposition is based on complicating constraints in dual decomposition. Role of the master problem also differs. In primal decomposition, master problem directly allocates common resources to the subproblems. Given the allocations, subproblems return prices for common resources. The role of the master problem is deciding the optimum allocation of common resources with respect to prices. From this point of view, primal decomposition is also referred to as *resource directive decomposition*.

On the other hand, in dual decomposition, master problem sets prices for common resources. In response, subproblems determine the amount of common resources that is to be used. In this case, the role of the master problem is deciding the optimum pricing. Hence, dual decomposition is also referred to as *price directive decomposition*. Benders decomposition and Dantzig-Wolfe decomposition can be given as examples for primal decomposition and dual decomposition, respectively.

Apart from these, there are also *primal-dual decomposition* methods which combines properties of both primal and dual decomposition methods. Primal-dual decomposition methods can be classified into *mixed decomposition*, *cross decomposition* and *mean value decomposition* [23].

Mixed decomposition is based on exploiting primal and dual decomposition at the same time. In an early example by Obel [24], a combination of Dantzig-Wolfe decomposition (a price-directive method) and Ten Kate method (a resource-directive method) is proposed for block angular linear problems. Master problem directs some divisions by setting prices and some others by allocating common resources. Convergence in finite number of iterations is proved. Meijboom [25] also proposes mixed decomposition approach for block angular linear problems. The key characteristic of this method is dividing complicating constraints into two groups. First group of complicating constraints is directed by prices while the second group of complicating constraints is directed by allocations simultaneously. Master problem obtains a lower bound and an upper bound for the objective function value of the original problem according to the responses coming from subproblems. The algorithm terminates when lower and upper bounds are close enough.

Cross decomposition is introduced by Van Roy [26] especially for mixed integer problems. The key idea of this method is using primal and dual decomposition methods alternatingly. Compared to the other methods, master problem is more passive in cross decomposition. Instead, there is *subproblem phase* where subproblems are solved iteratively until *convergence tests* fail. In that case, master problem takes action to ensure improvement of the method.

Mean value cross decomposition is similar to cross decomposition. The main difference between two methods is that in mean value cross decomposition the role of the master problem lessens. The method mainly based on subproblem phase in cross decomposition. There is no convergence test, hence to ensure convergence, mean value of all previous solutions is used as input instead of current solution. Master problem only pass information among subproblems and obtain mean values. This method is a generalization of Kornai-Lipták method for linear programming problems [27].

2.2. Decentralization in Linear Programs

The key idea in decomposition methods is to partition the overall problem into a master problem and several subproblems. Then the master problem collects information from the subproblems to direct them and obtain an optimal solution to the global problem. Thus, decomposition methods allow for decentralized decision making to a certain extent since the master problem acts as a center having access to the system wide information. However, several real-world optimization problems arising in many areas such as logistics, production and scheduling involve decision makers/parties/entities/agents that are unwilling to disclose their private data but want to collaborate for solving global problem for a mutual benefit. Consider an example of two companies producing and serving goods to the same region [28]. Each company has its own goals and products. Their supply and demand information is private. Without collaboration, there may be empty vehicle movements in optimal delivery route of both companies. However, by sharing their vehicles, they can reduce empty vehicle movement dramatically in their delivery routes and obtain minimum global cost of transportation while having their own production facilities.

Another example of decentralized decision making can be found in an supply chain environment where several production lines having their own private information such as production capacity, production and storage costs are sharing the same equipment. Hence the resulting problem may be a large-scale aggregate planning problem. In this case, building the global problem and solving it in a centralized way is impossible. However, decentralized decision making allows coordination of individual production lines by themselves to reach global solution with partial information exchange.

The solution approaches for decentralized decision making can be classified into three classes [29]: (i) cryptographic methods, (ii) transformation based methods, (iii) decomposition based methods. Cryptographic methods essentially rely on using cryptographic tools to hide the private data while performing each iteration in simplex method [30, 31] or interior point algorithms [32]. Although these methods guarantee data privacy strongly, they are inefficient and impractical for large-scale problems [32]. Moreover, from a decentralized point of view, relationship among the decision makers is insignificant since data privacy is ensured by encryption. Hence, cryptographic methods are not covered in the scope of this research. However, refer [33] to investigate cryptography-based optimization models. In this chapter, we review transformation based methods and decomposition based methods while we restrict ourselves to the prominent papers that address linear programming problems in block angular structure.

Transformation based methods are introduced by Du [34]. The main idea of transformation based methods is to *disguise* the linear programming problem into an equivalent problem by applying algebraic transformations that make use of random matrices. The main difference from the cryptographic methods is that the algebraic transformation is used once at the beginning of the algorithm instead of at each iteration. Then the disguised problem is solved and the solution is transformed back to the original problem domain. From a decentralized point of view, transformation based methods allow more than one decision makers to solve linear programming problems since the private data is enclosed by algebraic transformations.

The authors address vertically partitioned linear programs in [35] where columns of the constraint matrix and associated objective function coefficients are partitioned into p independent entities. Thus, one can view this kind of problems as block angular linear programs consisting of only the complicating constraints. The authors propose an exact transformation based approach where each entity first generates its own random transformation matrix. Then each entity shares its transformed constraint matrix and also transformed objective function coefficient vector with others. The authors formulate a *secure linear program* with public data revealed by the entities. Any one of the entities solves the disguised problem. Then each entity derives its own optimal solution from public optimal solution of the secure linear program.

Transformation based methods allows using recent advances that is proposed to solve linear programming problems once the problem is disguised by algebraic transformations. Thus, transformation based methods are not restricted to use simplex method or interior point algorithms unlike cryptographic methods. For instance, in [36], the authors make use of a decomposition method. They propose a secure and efficient algorithm that allows K collaborative agents to solve K-Agent linear programming problem (K-LP). Any K-LP problem can be partitioned into a problem with block angular structure. Thus, it can be distributed to the individual agents such that each agent only knows its block of local constraints B_i, its share in the complicating constraints A_i and associated vector of objective function coefficients c_i. First, the agents locally anonymize right hand side values of the constraints. Then each agent transforms A_i, B_i and c_i by its own transformation matrix Q_i. The authors use the appropriate choice of transformation matrix that is given in [37]. Then transformed K-LP problem is solved by Dantzig-Wolfe decomposition which utilize column generation method. An arbitrary agent is chosen to solve the master problem and obtain the dual values. Pricing subproblem of any agent i is formulated and solved not by its own but by a peer-agent j. If it exists, agent j sends agent i's new column to the agent solving the master problem. After updating the master problem with generated columns, new dual values distributed to the agents. Once the algorithm reaches global optimum, each agent uses back-transformation to find its part in optimal solution of the original problem. The authors state slow convergence for multiple agents as a main drawback of their approach. Thus their algorithm may stop with a near-optimal solution. On the other hand, the agents share all the information since privacy is ensured by transforming the constraint matrices and objective function vector.

A recent work by Hong *et al.* [38] applies transformation based methods to energy exchange optimization problems. The authors first formulate Microgrid Energy Exchange opTimization (MEET) problem for different scenarios arising in smart grid power distribution network. MEET is a block angular linear problem which minimizes global energy loss emanating while individual microgrids exchange their local energy instead of requesting it from a main grid. The authors propose an efficient solution procedure for MEET problem following the transformation based approach in [36] without revealing private information of any microgrid.

There are some drawbacks of transformation based methods. In addition to the computational cost of algebraic transformation, providing *heuristic security* is stated as the one of the main disadvantages [30, 39]. Hence, it is difficult to prove that the private data is protected in case of attacks [32]. Thus, the risk of disclosing whole private data is an issue in transformation based methods since it is revealed in a disguised way.

Decomposition based methods are preferred by the researchers since only the partial information is revealed in case of an attack. They differ from cryptographic methods and transformation based methods since they do not need any kind of encryption or transformation. The private data is inherently enclosed. Although they are not as secure as cryptographic methods, the decomposition based methods are more efficient than the others from computational point of view.

Decomposition based methods are the main focus of this research since they can serve as a model of decentralized decision making [40]. An early work is proposed by [41] which focus on block angular linear problems. The aim is to find the optimum partition for the global problem with minimal amount of information exchange among the subproblems. By partitioning the right hand side of the complicating constraints, they decompose the global problem into smaller subproblems. Thus, each subproblem consists of the block of local constraints and its share in the complicating constraints with its associated right hand side. Then each subproblem solves its problem and shares optimal dual variables with the other subproblems. New partition of the right hand side vector is found with a *coordination rule* utilizing the optimal dual variables of the subproblems. The process terminates with a near-optimal solution when some stopping condition is fulfilled.

In recent studies, decentralization that exploits decomposition methods has attained significant attention. The authors in [42] address capacity planning problem where finite capacity of a single facility is allocated among organizations to satisfy demand constraints. Hence the resulting linear programming problem represents block angular structure since the capacity constraints are linking the organizations' demand satisfaction constraints. The authors propose Cooperative Interaction via Coupling Agents (CICA) algorithm where the facility and organizations act as coupling agents. The CICA algorithm is mainly based on Lagrangian Relaxation. Thus the facility minimizes total cost of deviations from each organization's specified demand quantities subject to capacity constraints while each organization optimize local objective and deviations from facility's recommended capacity allocations subject to its demand constraints. A convex combination rule is utilized while updating Lagrangian multipliers to avoid oscillation of local solution among extreme points of the local feasible region. Numerical experiments are limited to include two coupling agents. Even in this case, the CICA algorithm does not guarantee global optimal solution. The authors represent the effect of amount of disclosed information to the solution quality. The CICA algorithm converges to near-optimal solution more quickly when organizations know partial information about capacity of the facility.

A hybrid method especially for solving cross-facility capacity allocation problem that combines Lagrangian relaxation and immunity-inspired coordination scheme is proposed in [43]. The problem shows block angular structure since it arises in a decentralized supply chain where multiple agents sharing some common resources manage manufacturing facilities. First, the authors decompose the problem among the agents. An agent's problem consists of the local constraints and the complicating constraints corresponding to its common resource usage. Agents sharing the same common resource are the neighbours. Then each agent relaxes its complicating constraints by adding them to the objective function by associated Lagrangian multipliers. Each agent solves its relaxed problem and communicates the optimal solution with its neighbours. Lagrangian multipliers are updated locally with respect to a biological immune system model by utilizing neighbours' stimulatory or suppressive effect. The algorithm terminates when an acceptable capacity violation is reached for all common resources. Thus, the algorithm does not guarantees to reach global optimum solution. And also, the formula for updating Lagrangian multipliers may become more complicated if there are more than two agents sharing same common resource.

A distributed simplex method allowing more than two decision makers to solve linear problems is proposed in [44]. The authors especially address the the security and access control issues arising in distributed data mining environments. They introduce a graph where each node acts as an individual decision maker and the edges denote the communication between the nodes. Each node having its own local constraints and also data processing capacity constraints. Instead of building a centralized model, the authors aim to obtain a partition of the global problem at each node corresponding to the local constraints. However, to reach global optimality, each node needs to add the relevant basic variables. Hence, an arbitrary node s is selected as an initiator to build a minimum spanning tree in the network. The initiator learns the total number of constraints in the tree and informs any other node i such that it adds the corresponding basic variables to its constraints. Then each node optimizes the same global objective function with respect to its own constraint set using simplex method. To do so, first each node i finds $row-pivot_i$ in its simplex tableau which denotes the minimum ratio of right hand side of a constraint to the corresponding number in the pivot column. Then each node sends its $row - pivot_i$ to the neighbours. The node having the minimum value of $row - pivot_i$ sends its row to all other nodes in the network. Then each node updates its simplex tableau with respect to the new row. Algorithm terminates when each node has only positive coefficients in the its objective function. However, these are some the drawbacks of this method from a decentralized point of view. First, to reach global solution, the algorithm requires that the nodes optimize the common global objective function. In addition to this, an initiator node who knows all the constraints in the centralized problem is required to initialize the algorithm.

Another distributed simplex based algorithm is proposed for block angular problems in a multi-agent setup in [45]. The aim of Two-Stage Distributed Simplex Algorithm is to solve the problem with information exchange only among the agents. They develop an algorithm that utilizes column generation method. The algorithm decomposes the aggregate model in a way that each agent has a local master problem and a subproblem. Each agent initially knows only the associated local constraints, its part in the objective function and right hand side of the complicating constraints. Local constraints set up the subproblem's feasible region. Each agent constructs an initial basis for local master problem by using Big-M method. The idea is allowing subproblems to generate new columns for the master problem. The authors assume a strongly connected communication network for the agents. In this set up, each agent consecutively performs three tasks: First, it transmits its current basis to its out-neighbours irregularly at least after a time interval having a maximal length. Second, whenever it gets a basis from its in-neighbours, it sorts all columns (in the existing basis and in the received basis) lexicographically. Then performs simplex algorithm. Third, it updates its basis with the optimal basis computed by simplex algorithm. The algorithm terminates when all agents converge to the same optimal basis. [46] gives an application of the algorithm to multi-agent assignment problem which has exact linear programming formulation.

In [47], the authors propose a decentralized coordination algorithm especially for Sales and Operations Planning problems in supply chain environments. The problem shows block angular structure since independent decision makers share some common resources while having private decision problems. There are two types of decision makers: One Informed Party (IP) and several Reporting Parties (RP). IP is arbitrarily chosen and it leads the others through calculating and announcing the common resource usage proposals. It solves a linear programming problem $CS2_I$ which is closely related to the Dantzig-Wolfe master problem. However, $CS2_I$ consists of local constraints of IP also. Thus IP determines common resource allocations and its local decisions simultaneously. On the other hand, RP i has two problems: $CS1_i$ and $CS - E_i$. Each RP evaluates any given common resource allocation by solving $CS - E_i$ and generates a new common resource usage proposal which is sent to IP. Moreover, optimal solution of $CS - E_i$ gives a starting solution for $CS1_i$. By solving $CS1_i$, RP i maximizes profit increase for new solution by comparing it with the last outcome of $CS - E_i$ and assigns a penalty or bonus term u_i for corresponding change in common resource usage. Since u_i is obtained within $CS1_i$, the objective function of $CS1_i$ is nonlinear. However, for fixed value of u_i, $CS1_i$ corresponds to Dantzig-Wolfe subproblem and authors propose an approximate solution approach. The resulting profit increase is sent to IP. Then IP updates common resource allocations by solving $CS2_I$. The algorithm terminates when optimal solution of $CS1_i$ is zero for all RP i and IP can no longer produce new common resource allocation. The authors state that their algorithm's computational time is comparable to global LP of same size if there is only one common resource.

In this research, we propose two decomposition based methods serving decentralized decision making for block angular linear programs: Decentralized Benders decomposition and Decentralized Dantzig-Wolfe decomposition. Differently from [42], 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 decision makers with respect to the type of information that they want to reveal. To the best of our knowledge, we are first to use Benders Decomposition method in decentralized decision making. We have no restriction on the number of complicating constraints in the global problem unlike [47]. Our methods do not require a distinct role definitions for decision makers as it is the case in [47] and [44]. The decision makers are equal on hierarchy and task assignment. Both methods are proved to converge to global optimal solution in finite number of iterations while [47], [42], [43] and [41] states near-optimal solution.

2.3. Decentralization in Integer Programs

Many decomposition methods that are proposed for linear programming problems have mixed integer or integer programming extensions. However they do not address decentralization allowing collaborative decision making which lacks central coordination.

There are relatively few studies in literature that address decentralization for integer programming problems. Among them, [48] addresses collaborative planning problem arising in supply chain environments and private e-marketplaces where each player has a private objective function and unique local problem depending on local variables. Also, players have a global objective function depending on some common variables. Hence the resulting problem represents dual block angular structure. Players' aim is to reach global Pareto-optimal solution collaboratively while hiding local objective functions by using some masked variables. The solution procedure is mainly based on integer L-shaped method. Thus, the players construct a mixed integer master problem jointly by using common variables and constraints associated with expected costs for players. While master problem is solved by branch and bound algorithm, players add optimality or feasibility cuts to the current master problem. The cuts are generated by solving each player's local linear subproblem using dual simplex method. The algorithm terminates when a global solution of the master problem is acceptable for all the players.

In [48], one can observe the case where linear subproblems allow making use of linear programming duality for cut generation. However, in some cases, subproblems are also integer programming problems. For instance, [49] addresses decentralized decision making for zero-one integer programming case. The problem is scheduling N jobs on a single shared machine while minimizing a linear function of completion times subject to some precedence constraints on jobs and machine capacity constraints. The problem represents block angular structure as the capacity constraints couple the local precedence constraints. The proposed solution method is mainly based on Lagrangian relaxation with some modifications on the way of relaxing coupling constraints and Lagrangian multiplier updating to reduce the amount of information shared. The capacity constraints of the shared machine is associated with the master problem as each disjoint subset of jobs is associated with a coupling agent having individual subproblems. The problem is solved by partial information exchange between master problem and subproblems. Authors report close-to-optimal solution for experiments including two coupling agents and consider including multiple coupling agents.

In this research, we address decentralized decision making for pure integer programs in block angular structure. Our method differs from the solution approach presented in [48] since subproblems are also integer programs. Different from [49], our solution approach is based on Integer L-shaped method allowing multiple decision makers solve the global problem exactly.

3. PRELIMINARIES

In this chapter, we present basic concepts that help for describing the proposed methods in detail in the following three chapters. First we formally define the problem that we are interested in. Then we give definitions and notation from graph theory to describe the structure of the communication scheme among decision makers. We state the main algorithm that is common for the proposed methods. Finally, we give sketch of convergence proof for the proposed methods.

3.1. Problem Statement

The problem has N decision makers which we call optimization agents (OA). We design OA as an individual, independent decision unit. The problem that we focus on can be defined as follows. Each OA $i \in \{1, 2, ..., N\}$ has its own set of decision variables x_{ij}'s for activity $j \in \{1, 2, ..., n_i\}$. OAs aim to minimize own linear cost $c_{ij}x_{ij}$, where $c_{ij} \in \Re$ and $x_{ij} \in \Re$ while satisfying a block of local constraints given by $\sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i$. Here b_{ij} denotes the local resource usage of OA i for activity j while l_i is the total available amount of the local resource for OA i. There is also a set of complicating constraints given by $\sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ij}^k x_{ij} = r^k$, where a_{ij}^k is the j^{th} entry of global constraint matrix of OA i for common resource k and r^k is the available amount of common resource $k \in \{1, 2, ..., K\}$. Hence, the resulting problem has primal block angular structure. It can be formulated as follows:

BALP

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

$$\text{subject to} \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \le r^k \quad \forall k = 1, 2, \ldots, m \tag{3.1b}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \le l_i \quad \forall i = 1, 2, \ldots, N \tag{3.1c}$$

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

We call this formulation as *aggregate problem.* The constraints given by (3.1b) are the complicating constraints that link OAs for each common resource k. Also, (3.1c) describes the blocks of local constraints that are private to each OA itself. For integer programming case, we refer 3.1 with integer x_{ij} variables. In centralization approach, the problem is built and solved as a whole by a center. However, we propose Decentralized Benders Decomposition and Decentralized Dantzig-Wolfe Decomposition for linear programs while we propose Decentralized Integer L-shaped Method for integer programs to solve the problem without a central coordination unit.

3.2. Communication Network

In decentralized methods, decision makers need to communicate among themselves to achieve global optimum since a central coordination is lacking. Usually information exchange among decision makers is provided through a communication network. In this section, we present some definitions and notation from graph theory to define the communication networks that we utilize. Then we introduce some common network topologies in literature.

A *graph* consists of a set of nodes N and a set of edges $E \subseteq N \times N$ and is denoted by *G(N,E)*. Graphs can be divided into two groups. *Undirected graphs* are the first group of graphs where the edges are unordered pairs of nodes such that *(i,j)* and *(j,i)* denotes the same edge between the nodes i and j, hence $(i, j) \in E$ if and only if $(j, i) \in E$. The other group of graphs is called *directed graphs* if any edge *(i,j)* of the graph is directed from node i to node j. Two nodes are called to be *adjacent* if there is an edge between them. The *neighbours* of a node i are the ones that have an incoming edge from node i. A *path* is a list of edges that connect a sequence of distinct nodes. A graph is called *strongly connected* if there exists a path of directed edges that goes from i to j for every pair of the nodes of the graph.

As mentioned before, in this research we want to propose decentralized methods which allow multiple optimization agents of same hierarchy to solve a large-scale block angular problem collaboratively. Since a central coordinating unit does not exist, individual optimization agents need to communicate and exchange information with each other in order to solve the problem. Thus, we consider a directed graph of N nodes to model the communication network among N decision makers in the sense that optimization agents are the nodes of the graph and the edges represent a communication link between two optimization agents. Then optimization agent i can only communicate with optimization agent j if there exists a path of directed edges from node i to the node j in the graph. In order for the problem to be reach global optimality, any pair of the optimization agents should communicate to each other. Hence, we assume a strongly connected graph for a communication network.

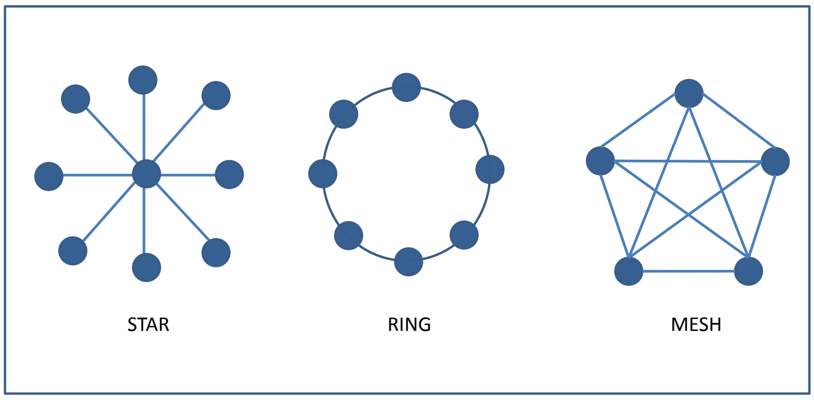


Figure 3.1. Common Network Topologies

There are several common communication network topologies in the literature as illustrated in Figure 3.1. Among them, Star Topology, Mesh Topology and Ring Topology are taken into consideration for our research. In the Star Network Topology, there is a node at the center and all the other nodes are directly connected to it. There is no direct connection between any one of the non-central nodes, however a node is connected to any other node through central node. In the Ring Network Topology, the nodes are connected in a closed loop configuration. While adjacent pairs of nodes are directly connected, the other nodes are connected indirectly through one or more intermediate nodes. In the Mesh Network Topology, each node is connected directly to all the other nodes. There is not a central node in Ring or Mesh network topologies.

3.3. Main Algorithm

Existing decomposition methods serve decentralization to a certain extent since they involve a center associated with the master problem to direct the other decision makers of each having a subproblem.

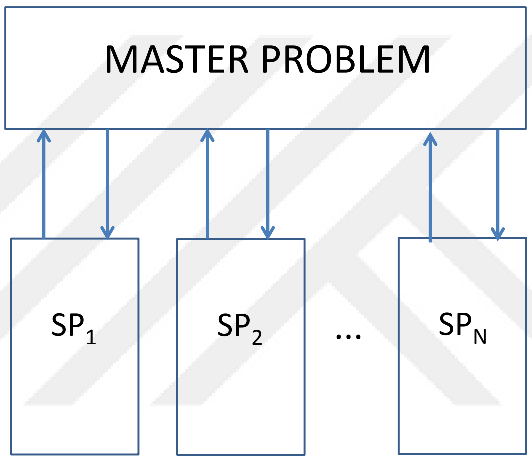


Figure 3.2. Structure and information exchange scheme for existing decomposition methods

Figure 3.2 illustrates the structure and information exchange scheme for decomposition methods. According to this, the master problem is on top of the subproblems collecting their private information to guide them either setting prices or setting allocations for common resources. However, many optimization problems involve decision makers of equal hierarchy level who want to solve the aggregate problem collaboratively. In this case, existing decomposition methods are not applicable or inefficient as a solution approach.

In this research, we propose Decentralized Benders Decomposition and Decentralized Dantzig-Wolfe Decomposition for block angular linear programs while we propose Decentralized Integer L-shaped method for block angular integer programs. Our aim is eliminating the center completely and allowing multiple decision makers to solve aggregate problem cooperatively by exchanging partial information through a peer-to-peer communication network.

The main assumptions of proposed methods are as follows. First, we assume a strongly connected communication graph that allows any OA to reach any cut/column generated by any other OA. We ignore communication lags and assume that OAs share partial information asynchronously only when it is required. Second, we assume an environment based on trust. OAs jointly solve the problem without conspiring each other to reach other's private information or changing problem data in time for ones own benefit.

Main algorithm is common for the proposed methods. The inputs are a subproblem - local master problem pair for each OA and a strongly connected graph to ensure the information exchange among OAs. The algorithm solves the aggregate problem for each OA in serial. First the algorithm solves the local master problem to get allocations/dual variables. Then it calls recursive GetCut/GetColumn function to find a new cut/column if it is available. The algorithm terminates when there is no new cut/column to be added to local master problem.

Recursive GetCut/GetColumn function takes current OA's identity, visited neighbours list and allocations/dual variables shared by the current OA's master problem as inputs. The algorithm first adds the current OA to the visited neighbours list to avoiding cycling. Then it updates the subproblem with given allocations/dual variables and solves it. A cut/column is generated with respect to the subproblem solution. If there is no new cut/column generated, then algorithm asks for a cut/column from all neighbours recursively until it finds one. If there is no new cut/column coming from the neighbours, then algorithm terminates.

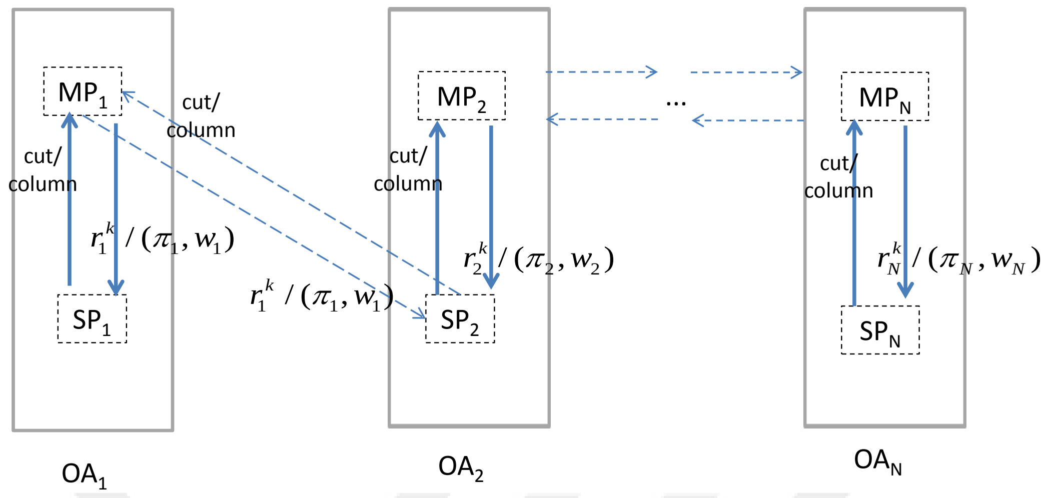


Figure 3.3. Structure and information exchange scheme for proposed methods

The main aspects of the algorithm are the following. First, communication among OAs occur whenever an OA's own subproblem cannot generate a new cut/column for given allocations/dual variables. In this case, local master problem communicates with the subproblem of a neighbouring OA by sharing allocations/dual variables for a new cut/column. Figure 3.3 illustrates the structure and information exchange scheme for the proposed methods. Second, if a new cut/column is generated by a neighbouring OA, then that cut/column is added to the local master problem of each OA in the path connecting the two communicating OAs. Third, at any iteration an upper bound and a lower bound on objective function value can be obtained. These bounds are of interest to terminate the algorithm when they are close enough. Finally, proposed methods can also be applicable in case of having hierarchy levels among the OAs provided that the communication is ensured via a strongly connected communication graph which is the same requirement as before.

3.4. Sketch of Convergence Proof

In the proposed methods, main algorithm terminates when there is no new cut or column generation. Hence the convergence proof of the proposed methods mainly based on the cut or column generation strategy. We prove convergence of the proposed methods in three parts. In the first part, we show that there are finitely many cuts or columns that can be generated. This implies that there is an upper bound on the number of iterations. The second part shows that each cut or column can be generated and added to the relaxed local master problem at most once. This implies that cycling is not allowed in proposed methods. Finally, in the third part, we show that the algorithm can reach any cut or column and add it to the relaxed local master problem by utilizing a recursive function. This is required to reach a global optimal solution. As a result, convergence of the proposed methods to a global optimal solution follows.

4. DECENTRALIZED BENDERS DECOMPOSITION FOR BLOCK ANGULAR LINEAR PROGRAMS

This chapter gives the details of Decentralized Benders Decomposition method for primal block angular linear programs. First, we describe Classical Benders Decomposition as a solution approach since it can be typically applied to dual block angular problems. Next, we propose Decentralized Benders Decomposition method. Finally, we prove the convergence of Decentralized Benders Decomposition method.

4.1. Classical Benders Decomposition Applied for Primal BALP

In this section, we briefly describe Classical Benders Decomposition approach for solving (3.1). Classical Benders Decomposition is originally defined as a Dantzig-Wolfe decomposition algorithm applied to the dual of the linear program in Primal Block Angular structure [8]. Thus, it can be best applied to the linear programs in Dual Block Angular Structure [13]. However, we propose a reformulation of (3.1), which allows us to use decentralized solution approach mainly based on Classical Benders Decomposition. According to this, (3.1) can be decomposed into several smaller subproblems by removing the complicating constraints given by (3.1b). However, we want to decompose the complicating constraints also. Thus, we introduce a new variable, r_i^k, for each k and i such that it denotes the amount of common resource k allocated to the OA i. Then, instead of each complicating constraint linking OAs for common resource k, the following set of constraints is introduced:

$$\sum_{j=1}^{n_i} a_{ij}^k x_{ij} = r_i^k \quad \forall i \tag{4.1}$$

Note that, sum of any allocation of common resource k should be equal to the available amount of that resource. Thus, the constraint $\sum_{i=1}^{N} r_i^k = r^k$ should be satisfied for each k. Hence, equivalent formulation for (3.1) after decomposing the complicating constraints can be given as the following:

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{4.2a}$$

$$\text{subject to:} \sum_{i=1}^{N} r_i^k = r^k \quad \forall k = 1,2,\ldots,K \tag{4.2b}$$

$$\sum_{j=1}^{n_i} a_{ij}^k x_{ij} = r_i^k \quad \forall i = 1,2,\ldots,N \quad \forall k = 1,2,\ldots,K \tag{4.2c}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \quad \forall i = 1,2,\ldots,N \tag{4.2d}$$

$$r_i^k \quad \textit{unrestricted} \quad \forall i = 1,2,\ldots,N \quad \forall k = 1,2,\ldots,K \tag{4.2e}$$

$$x_{ij} \geq 0 \quad \forall i = 1,2,\ldots,N \quad \forall j = 1,2,...,n_i \tag{4.2f}$$

With this reformulation, (4.2) has a dual block angular structure where r_i^k are treated as complicating variables. Furthermore, note that (4.2c) can be treated as local constraints in addition to (4.2d). Assume that complicating variables are fixed to a given value, $\hat{r}_i^k$. Then, the centralized problem in (4.2) can be solved as N independent problems only in x_{ij} variables. Each independent problem is called a subproblem. Given $\hat{r}_i^k$ values, the i^{th} subproblem, SP_i is formulated as the following:

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

$$\text{Minimize} \sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{4.3a}$$

$$\text{subject to:} \sum_{j=1}^{n_i} a_{ij}^k x_{ij} = \hat{r}_i^k \quad \forall k = 1,2,\ldots,K \tag{4.3b}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \tag{4.3c}$$

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

Note that, (4.3) is a linear program for given allocations. If any one of the subproblems is unbounded for $\hat{r}_i^k$, then (4.2) is also unbounded. This implies the unboundedness of the centralized problem in (3.1). Hence, we assume the boundedness of the subproblems. By introducing dual variables π_i^k associated with constraints (4.3b) and w_i associated with the constraint (4.3c), dual subproblem can be formulated as the following:

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

$$\begin{aligned}
&\text{Maximize } \sum_{k=1}^{m} \hat{r}_i^k \pi_i^k + l_i w_i && (4.4a)\\
&\text{subject to: } \sum_{k=1}^{K} a_{ij}^k \pi_i^k + b_{ij} w_i \leq c_{ij} \quad \forall j = 1, 2, ..., n_i && (4.4b)\\
&\pi_i^k \quad \textit{unrestricted} \quad \forall k = 1, 2, \ldots, K && (4.4c)\\
&w_i \quad \textit{unrestricted} && (4.4d)
\end{aligned}$$

Note that, only the objective function depends on the values of $\hat{r}_i^k$. The feasible region of dual subproblem does not depend on the given allocations, $\hat{r}_i^k$. If the feasible region of the dual subproblem is empty, then either SP_i is unbounded for some $\hat{r}_i^k$ or the feasible region of SP_i is also empty. Hence, we assume that the feasible region of dual subproblem is non-empty. Thus, one can enumerate all extreme points and extreme rays of the feasible region. All extreme points of the feasible region in (4.4) 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, all 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. For given $\hat{r}_i^k$, a bounded solution of dual subproblem corresponds to an extreme point of the feasible region. On the other hand, an unbounded solution of dual subproblem corresponds to an extreme ray of the feasible region. Thus, if dual subproblem is bounded, then there exists an extreme point, $p \in P_i$ that maximizes the objective function value $\hat{r}_i^k(\pi_i^k)^p + l_i w_i^p > q_i$. Otherwise, if dual subproblem is unbounded, then there exists an extreme ray, $r \in R_i$ such that $\hat{r}_i^k(\pi_i^k)^r + l_i w_i^r > 0$. Then (4.4) can be reformulated as the following:

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

$$\text{Minimize } q_i \tag{4.5a}$$

$$\text{subject to: } \hat{r}_i^k(\pi_i^k)^p + l_i w_i^p \leq q_i \quad p \in P_i \tag{4.5b}$$

$$\hat{r}_i^k(\pi_i^k)^r + l_i w_i^r \leq 0 \quad r \in R_i \tag{4.5c}$$

$$q_i \quad \textit{unrestricted} \tag{4.5d}$$

The constraints in (4.5) are called *Benders cuts.* Constraints of type (4.5b) are *Benders optimality cuts*, while constraints of type (4.5c) are *Benders feasibility cuts.* Then, (3.1) can be reformulated equivalently by using the Benders cuts:

MP

$$\text{Minimize } \sum_{i=1}^{N} q_i \tag{4.6a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k = r^k \quad \forall k = 1, 2, \ldots, K \tag{4.6b}$$

$$r_i^k(\pi_i^k)^p + l_i w_i^p \leq q_i \quad p \in P_i \quad \forall i = 1, 2, \ldots, N \tag{4.6c}$$

$$r_i^k(\pi_i^k)^r + l_i w_i^r \leq 0 \quad r \in R_i \quad \forall i = 1, 2, \ldots, N \tag{4.6d}$$

$$q_i \quad \textit{unrestricted} \tag{4.6e}$$

$$r_i^k \quad \textit{unrestricted} \tag{4.6f}$$

The number of Benders cuts in (4.6) is generally huge, since there are exponential number of extreme rays and extreme points of the feasible region of the dual subproblems. Since generating all Benders cuts is not practical, Classical Benders Decomposition starts with a Relaxed Master Problem (RMP) consisting of a subset of feasibility and optimality cuts. A central coordination unit solves RMP and announces the initial common resource allocations to the OAs. In each iteration, OAs update the objective function of its own dual subproblem by fixing the $\hat{r}_i^k$ to a value obtained by

RMP. Then dual subproblems are solved to optimality to produce either a feasibility cut or an optimality cut which is added to the RMP. Re-solving RMP gives new values for $\hat{r}_i^k$ to update dual subproblems in the next iteration. Since the sets of extreme points and extreme rays are finite, there are finitely many feasibility or optimality cuts to be added to the RMP. The convergence of Benders decomposition in a finite number of iterations follows since in each iteration a new Benders cut is generated. Efficiency of Benders Decomposition is based on the observation that the algorithm typically reaches optimality before addition of the all Benders cuts.

4.2. Decentralized Benders Decomposition Algorithm

Classical Benders Decomposition requires a center to solve RMP to determine common resource allocations. By Decentralized Benders Decomposition method, our aim is achieving 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

$$\text{Minimize} \quad q_i \tag{4.7a}$$

$$\text{subject to:} \quad \sum_{i=1}^{N} r_i^k = r_k \quad \forall k = 1, 2, \ldots, m \tag{4.7b}$$

$$r_i^k \text{ unrestricted} \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{4.7c}$$

$$q_i \geq LB_i \tag{4.7d}$$

Note that initially (4.7) consists of linking constraints (4.2b), only. We also initialize a lower bound LB_i for q_i which denotes the objective function value of the subproblem given by SP_i. Thus, we need to find a lower bound for the objective function value of the subproblem. Recall that SP_i consists of the complicating constraints given by (4.3b) and the local constraints given by (4.3c). Complicating constraints may shift as their right hand side value, $\hat{r}_i^k$, changes. However, the local constraints define a fixed feasible region. Hence, we formulate the following problem as a relaxation of (4.3) by eliminating the complicating constraints.

$$\text{Minimize } \sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{4.8a}$$

$$\text{subject to: } \sum_{j=1}^{n_i} b_{ij}x_{ij} = l_i \tag{4.8b}$$

$$x_{ij} \geq 0 \quad \forall j = 1, 2, ..., n_i \tag{4.8c}$$

Recall that we assume boundedness of the subproblems. Hence, the feasible region of local constraints is also bounded. The optimal objective function value of (4.8) gives a lower bound for q_i since it is a relaxation. (4.7d) ensures the boundedness of the local master problem at initial iterations.

```
  Input: $DSP_i(r_i^k)$ { Dual subproblem for OA $i$}
  Input: $MP_i$ {Local master problem for OA $i$}
  Input: $G = \{N, A\}$ {Strongly connected digraph}
1 {$r_i^k$ denotes the amount of common resource $k$ allocated to 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 $BC = Null$
```

Figure 4.1. Pseudo-code for Decentralized Benders Decomposition Algorithm

We describe Decentralized Benders Decomposition Algorithm in Figure 4.1. It starts with a pair of problems $DSP_i(r_i^k) - MP_i$ for each OA $i \in \{1, 2, ..., N\}$ and a strongly connected communication graph, $G = \{N, A\}$. Each OA i solves its local master problem, MP_i and gets r_i^k that denotes the amount of common resource k

allocated to OA i (line 6). Decentralized Benders Decomposition Algorithm utilizes GetCut function (line 8) which looks for a cut from neighbours recursively, if there is no cut generated by the OA itself. Hence, Decentralized benders Decomposition Algorithm keeps track of the visited OAs with the list, V. It terminates when there is no new cut is generated for any of the OAs (line 9).

```
   Input: $j$ {Identity of OA}
   Input: $V$ {Visited Neighbours List}
   Input: $r_i^k \quad \forall k = 1, 2, \ldots, K$ {Allocation of common resource $k$ in OA $i$'s
          proposal}
   Output: $BC$ {Benders Feasibility or Optimality cut}
 1 {This is a recursive function that returns a Benders cut, if it exists}
 2 if $j \in V$ then
 3     return $Null$
 4 $V \leftarrow V \cup \{j\}$
 5 Update objective function of $DSP_j(r_i^k)$
 6 Solve $DSP_j(r_i^k) \rightarrow GetStatus$
 7 if $GetStatus = Optimal$ then
 8     $BC \leftarrow$ Generate Benders optimality cut according to (4.5b)
 9 else if $GetStatus = Unbounded$ then
10     $BC \leftarrow$ Generate Benders feasibility cut according to (4.5c)
11 if $BC = Null$ then
12     forall $n \in N_j$ do
13         $BC \leftarrow$ GetCut$(n, V, r_i^k)$
14         if $BC \neq Null$ then
15             Break out of the For loop
16 Add $BC$ to $MP_j$
17 return $BC$
```

Figure 4.2. Pseudo-code for GetCut Function

Figure 4.2 gives the pseudo-code of the GetCut function that returns a Benders cut. It adds the current OA to the list to avoid visiting it more than once (line 4). Then objective function of $DSP_j(r_i^k)$ is updated with the given allocations and it is solved (line 5). According to the solution status of $DSP_j(r_i^k)$, either a Benders feasibility cut or a Benders optimality cut is generated (line 6, line 10). Otherwise, GetCut function looks for a new cut recursively from all neighbouring OAs (line 11, line 13) until it finds a new cut (line 14). If a new cut is generated then it is added to the local master problem of OA j (line 16). GetCut function runs until all OAs are visited (line 3). Note that, GetCut is a recursive function. Hence, by definition, when a cut is generated by OA j with respect to OA i's allocations through GetCut function, then all the OAs on the path connecting OAs i and j adds that cut to their local master problem.

Notice that at any iteration, optimal objective function value of local master problem is a lower bound on objective function value of (3.1) since it consists of only a subset of constraints. Also notice that the sum of objective function values of the subproblems is an upper bound for objective function value of (3.1). The advantage of this feature is allowing termination before reaching global optimality if lower and upper bounds are close enough.

4.3. Convergence

In this section, 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. As a result, 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 [2]. First we state the following theorem which defines the multipliers obtained by projecting out the x_{ij} variables in (4.3) and also the projection of the polyhedron.

Theorem 2. If

$$P_i = \{(x,r) \in \Re^{n_i} \times \Re^k | \sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq r_i^k \quad \forall k = 1,2,\ldots,K, \quad \sum_{j=1}^{n_i} b_{ij} x_{ij} \geq l_i\} \tag{4.9}$$

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

$$\sum_{k=1}^{K} \sum_{j=1}^{n_i} a_{ij}^k \pi_i^k + \sum_{j=1}^{n_i} b_{ij} w_i = 0 \tag{4.10}$$

Also the projection of the polyhedron is

$$proj_r(P_i) = \{r \in \Re^k | \pi_i^k r_i^k \geq 0 \quad \forall k = 1,2,\ldots,K\} \tag{4.11}$$

Then we state two propositions which are adapted from [50] 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 1. If P_i is given in (4.9) then $proj_r(P_i)$ is given in (4.11) where $\{(\pi_i^k, w_i), \forall k = 1,2,\ldots,K\}$ are the extreme rays of the projection cone

$$C_x(P_i) = \{(\pi, w) | \sum_{k=1}^{K} \sum_{j=1}^{n_i} a_{ij}^k \pi_i^k + \sum_{j=1}^{n_i} b_{ij} w_i = 0, \pi_i^k \geq 0 \quad \forall k = 1,2,\ldots,K, \quad w_i \geq 0\} \tag{4.12}$$

Proposition 2. If P_i is given in (4.9) and $\{(\pi_i^k, w_i), \forall k = 1,2,\ldots,K\}$ are the multipliers that are generated using projection, then the extreme rays of the projection cone in (4.12) are contained in this set of multipliers.

We omit the proof of Proposition 1 and Proposition 2 here. Refer to [50] for proof of generalized cases as Proposition 2.22 and Proposition 2.23, respectively.

Theorem 2 states that the inequalities $\pi_i^k r_i^k \geq 0 \quad \forall k = 1, 2, \ldots, K$ that results from projecting out the x_{ij} variables from the system defines $proj_r(P_i)$. By Proposition 1, $proj_r(P_i)$ can also be generated by the extreme rays of the projection cone $C_x(P_i)$. Proposition 2 defines the relationship between the multipliers $\{(\pi_i^k, w_i), \forall k = 1, 2, \ldots, K\}$ 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:

$$\text{Minimize } z_0 \tag{4.13a}$$

$$\text{subject to: } z_0 - \sum_{i=1}^{N} \sum_{j=1}^{n_i} c_{ij} x_{ij} \geq 0 \tag{4.13b}$$

$$\sum_{i=1}^{N} r_i^k - r^k = 0 \quad \forall k = 1, 2, \ldots, K \tag{4.13c}$$

$$\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \geq r_i^k \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{4.13d}$$

$$\sum_{j=1}^{n_i} b_{ij} x_{ij} \geq l_i \quad \forall i = 1, 2, \ldots, N \tag{4.13e}$$

$$x_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i \tag{4.13f}$$

$$r_i^k \text{ unrestricted} \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{4.13g}$$

Let us assign the multiplier u_0 to the constraint (4.13b) and the vector of multipliers π, w, u to the constraints (4.13d), (4.13e) and (4.13f), respectively. Using these multipliers, we project out the x_{ij} variables. This gives the following:

$$\text{Minimize } z_0 \tag{4.14a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k - r^k = 0 \quad \forall k = 1, 2, \ldots, K \tag{4.14b}$$

$$u_0^p z_0 \geq (\pi_i^k)^p r_i^k + {w_i}^p l_i \quad \forall i \quad \forall k \quad \forall p \tag{4.14c}$$

$$r_i^k \text{ unrestricted} \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{4.14d}$$

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 $i = t + 1, \ldots, P$. Hence the resulting problem is:

$$\text{Minimize } z_0 \tag{4.15a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k = r_k \quad \forall k = 1, 2, \ldots, K \tag{4.15b}$$

$$z_0 \geq (\pi_i^k)^p r_i^k + {w_i}^p l_i \qquad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \quad \forall p = 1, 2, \ldots, t \tag{4.15c}$$

$$0 \geq (\pi_i^k)^p r_i^k + {w_i}^p l_i \qquad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \quad \forall p = t + 1, \ldots, P \tag{4.15d}$$

$$r_i^k \text{ unrestricted} \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{4.15e}$$

From theory of projection, r^* is an optimal solution to (4.15) if and only if there is an x^* such that (x^*, r^*) is an optimal solution to the aggregate problem. By Proposition 2, the extreme rays of the projection cones

$$C_x(P_i) = \{(u_0, \pi, w, u) | \sum_{k=1}^{K} a_{ijk} \pi_i^k + b_{ij} w_i + u_{ij} - u_0 c_{ij} = 0 \qquad \forall j = 1, 2, ..., n_i, (u_0, \pi, w, u) \geq 0\} \tag{4.16}$$

are contained in the set of multipliers

$$\{(u_0^p, (\pi_i^k)^p, {w_i}^p, (u_{ij})^p) | \forall j = 1, 2, ..., n_i \quad \forall k = 1, 2, \ldots, K \quad \forall p = 1, 2, \ldots, P\} \tag{4.17}$$

generated by projection $\forall i = 1, 2, \ldots, N$. By Proposition 1, only the extreme rays of the projection cones $C_x(P_i) \quad \forall i = 1, 2, \ldots, N$ are needed to generate constraints

(4.15c)−−(4.15d) which characterize the projection into r space. Therefore, we can conclude that the constraints (4.15c)−−(4.15d) are generated from the extreme rays of the projection cones.

Proposition 3. If the multipliers in (4.17) are the extreme rays of the projection cone in (4.16) scaled so that $u_0^p = 1$, for $p = 1, 2, \ldots, t$ and $u_0^p = 0$, for $p = t + 1, 2, \ldots, P$, then $((\pi_i^k)^p, (w_i)^p)$, for $p = 1, 2, \ldots, t$ are the extreme points of the polyhedron

$$\{(\pi_i^k, w_i) | \sum_{k=1}^{K} a_{ijk}\pi_i^k + b_{ij}w_i \leq c_{ij}, (\pi_i^k, w_i) \geq 0 \quad \forall j = 1, 2, \ldots, n_i\} \tag{4.18}$$

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

$$\{(\pi_i^k, w_i) | \sum_{k=1}^{K} a_{ijk}\pi_i^k + b_{ij}w_i \leq 0, (\pi_i^k, w_i) \geq 0 \quad \forall j = 1, 2, ..., n_i\} \tag{4.19}$$

If all the constraints associated with the extreme rays are generated, then (4.15) becomes *full master problem*. Since solving full master problem is not practical, a *relaxed master problem* having a subset of the constraints (4.15c)−−(4.15d) 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 3, one can find a constraint that violates the relaxed master problem by solving the following subproblem:

$$\text{Maximize } \sum_{k=1}^{K} \hat{r_i^k}\pi_i^k + l_i w_i \tag{4.20a}$$

$$\text{subject to: } \sum_{k=1}^{K} a_{ij}^k \pi_i^k + b_{ij}w_i \leq c_{ij} \quad \forall j = 1, 2, ..., n_i \tag{4.20b}$$

$$\pi_i^k \geq 0 \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{4.20c}$$

$$w_i \geq 0 \quad \forall i = 1, 2, \ldots, N \tag{4.20d}$$

Proposition 4. The number of Benders cuts that can be generated in Decentralized Benders Decomposition is finite.

Proof. Assume that $\bar{r}$ is a feasible solution to the relaxed master problem with an 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 $\bar{z}_0 \geq \hat{r}_i^k \pi_i^k + l_i w_i$ 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 $\hat{r}_i^k \pi_i^k + l_i w_i \leq 0$ to the relaxed master problem. Therefore, there is only one constraint associated with each extreme ray or extreme point. The number of extreme rays and extreme points is finite since the feasible region of the subproblem is a polyhedron. Hence one can conclude that the number of cuts that can be generated is also finite. □

PART II: A unique cut is generated at each iteration. We state and prove the following proposition for this part.

Proposition 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. We state and prove the following proposition for this part.

Proposition 6. Decentralized Benders Decomposition Algorithm converges in a finite number of iterations if the neighbourhood 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 function. 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. □

5. DECENTRALIZED DANTZIG-WOLFE DECOMPOSITION FOR BLOCK ANGULAR LINEAR PROGRAMS

In this chapter, we introduce Decentralized Dantzig-Wolfe Decomposition Algorithm given in Figure 5.1. Decentralized Danzig-Wolfe Algorithm utilize Phase I algorithm to ensure the feasibility of local master problem. Once we get an initial feasible local master problem, we describe Phase II algorithm that utilize recursive GetColumn function to seek a column either from the OA's own subproblem or subproblem of any neighbouring OA. Finally, we prove the finite convergence of Decentralized Dantzig-Wolfe Decomposition.

```
1 $IsFeasible \leftarrow$ Phase I Algorithm
2 if IsFeasible then
3     Phase II Algorithm
```

Figure 5.1. Pseudo-code for Decentralized Dantzig-Wolfe Decomposition Algorithm

5.1. Constructing Feasible Local Master Problems

We consider block angular linear programs given by (3.1). We exploit the special structure of BALP to decompose the aggregate model into subproblem - local master problem pairs. Without the complicating constraints in (3.1b), the problem can be decomposed into N smaller subproblems. Each block of local constraints in (3.1c) is associated with a subproblem while master problem is given as the following:

MP

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{n_i} c_{ij} x_{ij} \tag{5.1a}$$

$$\text{subject to } \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \le r^k \quad \forall k = 1, 2, \ldots, K \tag{5.1b}$$

$$x_{ij} \ge 0 \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i \tag{5.1c}$$

Using Minkowski's Representation Theorem, any point x_{ij} in the feasible region of subproblem i can be expressed as:

$$x_{ij} = \sum_p \lambda_i^p x_i^p + \sum_r \mu_i^r x_i^r \tag{5.2a}$$

$$\sum_p \lambda_i^p = 1 \tag{5.2b}$$

$$\lambda_i^p \ge 0 \quad \forall x_i^p \in P_i \tag{5.2c}$$

$$\mu_i^r \ge 0 \quad \forall x_i^p \in R_i \tag{5.2d}$$

where P_i and R_i are the sets of extreme points and extreme rays of the subproblem i's feasible region, respectively. For the sake of simplicity, we assume a bounded feasible region. Hence, we can reformulate (MP_i) as the following:

MP_i

$$\text{Minimize } \sum_{j=1}^{n_i}\sum_p c_{ij} x_i^p \lambda_i^p \tag{5.3a}$$

$$\text{subject to } \sum_{j=1}^{n_i}\sum_p a_{ij}^k x_i^p \lambda_i^p \le r^k \quad \forall k = 1, 2, \ldots, K \tag{5.3b}$$

$$\sum_p \lambda_i^p = 1 \tag{5.3c}$$

$$\lambda_i^p \ge 0 \quad \forall x_i^p \in P_i \tag{5.3d}$$

MP_i may be a huge linear program since the set of extreme points may be exponentially large. Hence we initialize restricted master problem having a column associated with an extreme point. To find an initial column we solve the following subproblem once for each OA_i:

SP_i^0

$$\text{Minimize} \sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{5.4a}$$

$$\text{subject to} \sum_{j=1}^{n_i} b_{ij}x_{ij} \leq l_i \tag{5.4b}$$

$$x_{ij} \geq 0 \quad \forall j = 1, 2, ..., n_i \tag{5.4c}$$

Note that, the constraint set in (5.4) is a subset of the constraints in BALP. Hence, if any one of the subproblems is infeasible in the initialization step, then we can conclude that BALP is infeasible. Otherwise we generate a column and add it to MP_i. However, MP_i may be infeasible because complicating constraints are violated. To ensure feasibility of MP_i we use Phase I algorithm in Algorithm **??** as it is described in [51].

According to this, we introduce an artificial variable s_k for each complicating constraint k and minimize their sum. Hence, the resulting master problem, $MP-I$ is as the following:

$MP-I$

$$z_{MP-I} = \text{Minimize} \sum_{k=1}^{K} s_k \tag{5.5a}$$

$$\text{subject to} \sum_{j=1}^{n_i} \sum_{p} a_{ij}^k x_i^p \lambda_i^p - s_k \leq r^k \quad \forall k = 1, 2, \ldots, K \tag{5.5b}$$

$$\sum_{p} \lambda_i^p = 1 \tag{5.5c}$$

$$\lambda_i^p \geq 0 \quad \forall x_i^p \in P_i \tag{5.5d}$$

$$s_k \geq 0 \quad \forall k = 1, 2, \ldots, K \tag{5.5e}$$

We describe Phase I Algorithm in Figure 5.2. First, it starts by solving $(MP-I)$ to get dual variables $\pi_k \quad \forall k = 1, 2, \ldots, K$ associated with complicating constraints and w_i associated with convexity constraint (line 3). Then the algorithm searches each OA for a new column. A variable can be added to $(MP-I)$ with its corresponding column with respect to its reduced cost (RC). The following equation gives the reduced cost of a variable:

$$RC: \quad \sum_{j=1}^{n_i}(c_{ij} - \sum_{k=1}^{K} \pi_k a_{ij}^k) x_{ij} - w_i \tag{5.6}$$

Note that we set $c_{ij} = 0 \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i$ in (5.6) since the objective function of $(MP-I)$ is minimization of $\sum_{k=1}^{K} s_k$. The most profitable variable enter to the master problem is the one having the most negative reduced cost. Hence, the pricing subproblem to choose the most promising variable is as the following:

$SP_i(\pi_k, w_i)$

$$z_{SP_i} = \text{Minimize} \sum_{j=1}^{n_i}(c_{ij} - \sum_{k=1}^{K} \pi_k a_{ij}^k) x_{ij} - w_i \tag{5.7a}$$

$$\text{subject to} \sum_{j=1}^{n_i} b_{ij} x_{ij} \leq l_i \tag{5.7b}$$

$$x_{ij} \geq 0 \quad \forall j = 1, 2, ..., n_i \tag{5.7c}$$

To decide whether there is a new variable enter to the master problem with respect to the given dual variables, Phase I algorithm updates (5.7) with (π_k, w_i) (line 6) and solves. Assume that optimal solution of $SP_i(\pi_k, w_i)$ is x_{ij}^* (line 7). Note that, the objective function value z_{SP_i} of $SP_i(\pi_k, w_i)$ gives the minimum reduced cost (line 8). If RC is negative for a variable, then a column generated by x_{ij}^* with respect to the following equation (line 10):

$$C: [\sum_{j=1}^{n_i} c_{ij} x_{ij}^* \quad \sum_{j=1}^{n_i} a_{ij}^1 x_{ij}^* \cdots \sum_{j=1}^{n_i} a_{ij}^k x_{ij}^* \quad 1]^T \tag{5.8}$$

```
1  {This algorithm finds a feasible local master problem for OAs}
2  repeat
3      $MP - I \rightarrow (\pi_k, w_i)$
4      $hasColumn \leftarrow False$
5      forall $i \in \{1, \ldots, N\}$ do
6          Update objective function of $SP_i$ with $(\pi_k, w_i)$
7          Solve $SP_i(\pi_k, w_i) \rightarrow x^*_{ij}$
8          $RC \leftarrow z^*_{SP_i}$ { RC denotes reduced cost in (5.6)}
9          if $RC < 0$ then
10             Generate C according to (5.8)
11             Add C to $(MP - I)$
12             $hasColumn \leftarrow True$
13     if not $hasColumn$ then
14         return False {$(MP - I)$ is infeasible so is BALP}
15 until $z_{(MP-I)} = 0$
16 return True
```

Figure 5.2. Pseudo-code for Phase I Algorithm

where T is the transpose operator. Here the first entry of (5.8) is the coefficient for the objective function. Since $c_{ij} = 0 \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i$, it is equal to zero for $(MP - I)$. Next k entries are the coefficients with respect to the complicating constraints and the final entry 1 is for convexity constraint.

The new column is added to $(MP - I)$ (line 11). If there is no new column generated by any of the OAs, then Phase I algorithm terminates since $(MP - I)$ is infeasible which means that BALP infeasible (line 14). Otherwise, at least one column is generated and Phase I algorithm terminates with feasible local master problem if objective function value, $z_{(MP-I)}$ is zero (line 16).

5.2. GetColumn Function

Once we get feasibility of master problem for each OA with Phase I algorithm, we change the objective function of master problem in (5.3) from $\sum_{k=1}^{K} s_k$ to $\sum_{j=1}^{n_i} c_{ij} x_i^p \lambda_i^p$ and run Phase II Algorithm.

We describe Phase II algorithm in Figure **??**. It starts with a pair of problems $SP_i(\pi_k, w_i) - MP_i$ for each OA $i \in \{1, 2, ..., N\}$ and a strongly connected communication graph, $G = \{N, A\}$. Each OA i solves its local master problem, MP_i and gets dual variables (π_k, w_i) (line 6). Phase II Algorithm utilizes recursive GetColumn function (line 8) to find a new column. It terminates when there is no new column is generated for any of the OAs (line 9).

```
  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 {$(\pi_k, w_i)$ denotes the dual variables of $MP_i$ }
2 {$V$ denotes the visited neighbours list for finding a new column from the
   neighbours}
3 {$C$ denotes the column generated}
4 for $i = 1$ to $N$ do
5     repeat
6         Solve $MP_i \rightarrow (\pi_k, w_i)$
7         $V \leftarrow \varnothing$
8         $C \leftarrow GetColumn(i, V, (\pi_k, w_i))$
9     until $C = Null$
```
Figure 5.3. Pseudo-code for Phase II Algorithm

Figure 5.4 gives the details of the GetColumn function which returns a new column (line 16). It updates the visited neighbours list (line 4).Then $SP_j(\pi_k, w_i)$ is updated and solved (line 5, line6). If reduced cost is negative for a variable, then a new column is generated(line 9). Otherwise, GetColumn function looks for a new column recursively from neighbours (line 11, line12). A new column is added to the local master problem (line 15) of OA j. The algorithm terminates when all the neighbours are visited. Note that, when a column is generated by OA j for OA i, then by definition of recursive GetColumn function, all the OAs on the path connecting the OAs add that column to their local master problem.

```
1  {This is a recursive function that returns a column, if it exists}
   Input: $j$ {Identity of OA}
   Input: $V$ {Visited Neighbours List}
   Input: $(\pi_k, w_i)$ {Dual variables of $MP_i$ }
   Output: $C$ {New Column}
2  if $j \in V$ then
3      return $Null$
4  $V \leftarrow V \cup \{j\}$
5  Update objective function of $SP_j(\pi_k, w_i)$
6  Solve $SP_j(\pi_k, w_i) \rightarrow z^*_{SP_j}$
7  $RC \leftarrow z^*_{SP_j}$ {RC denotes reduced cost in (5.6)}
8  if $RC < 0$ then
9      Generate C according to (5.8)
10 else
11     forall $n \in N_j$ do
12         $C \leftarrow$ GetColumn$(n, V, r_i^k)$
13         if $C \neq Null$  then
14             Break out of the For loop
15 Add $C$ to $MP_j$
16 return $C$
```

Figure 5.4. Pseudo-code for GetColumn Function

5.3. Convergence

In this section, we prove the finite convergence of Decentralized Dantzig-Wolfe decomposition in three parts. Assume that BALP has a finite optimal solution. Decentralized Dantzig-Wolfe Decomposition algorithm breaks BALP into many local master problems each of which is associated with an OA. Then by exploiting column generation method, each OA solves its local master problem individually. Thus, the convergence of the algorithm mainly depends on the columns added to the relaxed local master problem of any OA.

PART I: There are finitely many columns to be generated. We state and prove the following proposition for this part.

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

Proof. Let $S_i = \{x_{ij} | \sum_{j=1}^{n_i} b_{ij}x_{ij} \leq l_i\}$ denotes the feasible region of i^{th} 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 as (5.2a) 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 (5.8). 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. $\square$

PART II: A unique column is generated at each iteration. We state and prove the following proposition for this part.

Proposition 8. Decentralized Dantzig-Wolfe Decomposition yields an unique column at each iteration.

Proof. Local master problem for OA i given in (5.3) is a linear programming problem. Thus, one can calculate reduced cost of any variable x_{ij} by (5.6). 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: Any violated column can be detected and added to any local master problem. We state and prove the following proposition for this part.

Proposition 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 GET-COLUMN function. 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 Proposition 8 and Proposition 9. □

6. DECENTRALIZED INTEGER L-SHAPED METHOD

In this chapter, we present Decentralized Integer L-shaped method for integer programming problems in block angular structure. First, we state the problem that we address. In this case, both master problem and the subproblems consist of integer variables. Hence we utilize a cut generation procedure inspired from the work done by Laporte and Louveaux [5] for our algorithm. Then we prove that our algorithm converges to global optimal solution in a finite number of iterations.

6.1. Decentralized Integer L-Shaped Algorithm

We focus on primal Block Angular Integer Program (BAIP) that is given in (6.1).

BAIP

$$\text{Minimize } \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{6.1a}$$

$$\text{subject to } \sum_{i=1}^{N}\sum_{j=1}^{n_i} a_{ij}^{k}x_{ij} \leq r^k \quad \forall k = 1,2,\ldots,K \tag{6.1b}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \leq l_i \quad \forall i = 1,2,\ldots,N \tag{6.1c}$$

$$x_{ij} \geq 0 \text{ and integer}, \quad \forall i = 1,2,\ldots,N \quad \forall j = 1,2,...,n_i \tag{6.1d}$$

The aggregate problem in (6.1) can be decomposed into a master problem and independent subproblems by removing the complicating constraints in (6.1b). To decompose the complicating constraints, we introduce a new variable r_i^k for each k and i such that it denotes the amount of common resource k assigned to the optimization agent (OA) i. Hence the resulting BAIP with new constraint set is given as the following:

$$\text{Minimize } \sum_{i=1}^{N}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{6.2a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k = r^k \quad \forall k = 1,2,\ldots,K \tag{6.2b}$$

$$\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \leq r_i^k \quad \forall i = 1,2,\ldots,N \quad \forall k = 1,2,\ldots,K \tag{6.2c}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \leq l_i \quad \forall i = 1,2,\ldots,N \tag{6.2d}$$

$$r_i^k \text{ unrestricted and integer} \quad \forall i = 1,2,\ldots,N \quad \forall k = 1,2,\ldots,K \tag{6.2e}$$

$$x_{ij} \geq 0 \text{ and integer} \quad \forall i = 1,2,\ldots,N \quad \forall j = 1,2,...,n_i \tag{6.2f}$$

The aggregate problem in (6.2) can be decomposed into a subproblem - local master problem pair for each OA. Given the $\hat{r}_i^k$ values, subproblem i can be formulated as the following:

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

$$\text{Minimize } \sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{6.3a}$$

$$\text{subject to: } \sum_{j=1}^{n_i} a_{ij}^k x_{ij} \leq \hat{r}_i^k \quad \forall k = 1,2,\ldots,K \tag{6.3b}$$

$$\sum_{j=1}^{n_i} b_{ij}x_{ij} \leq l_i \tag{6.3c}$$

$$x_{ij} \geq 0 \text{ and integer} \quad \forall j = 1,2,...,n_i \tag{6.3d}$$

Note that (6.3) is an integer programming problem. In this case, we cannot use linear programming duality to generate Benders cuts as we have done in Decentralized Benders Decomposition. Instead, we adopt cut generation strategy of Laporte and Louveaux [5], which they use for stochastic integer programs with binary first stage decision variables. Binary first stage variables are associated with common resource allocation variables r_i^k in our case. Thus, we need to convert integer r_i^k variables to binary variables to utilize the cut generation strategy. It is known that any bounded integer variable can be expressed as a set of binary variables. Let x be any nonnegative integer variable. For $0 \leq x \leq u$, there exists a number S such that

$$2^S \leq u < 2^{S+1}$$

is satisfied. Here S denotes the minimum required number of binary variables to represent x. Hence representation of x as binary variables is

$$x = 2^0 y_0 + 2^1 y_1 + 2^2 y_2 + \cdots + 2^s y_S$$

where $y_i \in \{0, 1\}$. Any bounded but not necessarily positive integer x, where $l \leq x \leq u$, can be transformed in a similar way since $0 \leq x - l \leq u - l$ is a bounded nonnegative integer.

To find a lower bound for each integer variable r_i^k, we solve an linear problem that minimizes left hand side of (6.3b) subject to the local constraint set since the right hand side of this equation defines r_i^k. Hence the resulting problem is as the following:

$$\text{Minimize} \sum_{j=1}^{n_i} a_{ij}^k x_{ij} \tag{6.4a}$$

$$\sum_{j=1}^{n_i} b_{ij} x_{ij} \leq l_i \tag{6.4b}$$

$$x_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, N \quad \forall j = 1, 2, ..., n_i \tag{6.4c}$$

Similarly, for finding upper bound we solve (6.4) as a maximization problem. To find the bounds, we assume that local constraint set has a bounded feasible region.

Assume each r_i^k is represented with a set of binary variables $r_i^k = \{r_{i0}^k, r_{i1}^k, \cdots, r_{iS}^k\}$ such that $r_{is}^k \in \{0, 1\}$ and $r_i^k = 2^0 r_{i0}^k + 2^1 r_{i1}^k + 2^2 r_{i2}^k + \cdots + 2^S r_{iS}^k$. Also assume that $r_i = \{r_i^1, r_i^2, \cdots, r_i^k\}$ denotes the amount of each common resource k allocated to OA i by master problem. With given r_i values, OA i solves (6.3). As in the linear programming case, proposed algorithm generates *feasibility cuts* if the subproblem is infeasible. If subproblem has a finite solution, then the algorithm generates *optimality cuts*. Subproblems do not have unbounded solution, since we assume local constraint set has a bounded feasible region.

6.1.1. Feasibility Cut Generation

At iteration t, let common resource allocation variables r_i^t for OA i are given, then define two sets

$$I_i^t = \{s|(r_{is}^k)^t = 1\} \text{ and } Z_i^t = \{s|(r_{is}^k)^t = 0\}$$

Next define the linear function

$$\delta_i^t(r_i^t) = \left|I_i^t\right| - \left[\sum_{s \in I_i^t} r_{is}^k - \sum_{s \in Z_i^t} r_{is}^k\right] \tag{6.5}$$

If subproblem is infeasible with respect to given r_i^t values, then this solution should be excluded. Thus, subproblem generates a feasibility cut by switching at least one of the binary components' value from one to zero or zero to one. This can be expressed by the following inequality:

$$\left[\sum_{s \in I_i^t} r_{is}^k - \sum_{s \in Z_i^t} r_{is}^k\right] \leq \left|I_i^t\right| - 1, \qquad i.e. \quad \delta_i^t(r_i^t) \geq 1 \tag{6.6}$$

To verify validity of $\delta_i^t(r_i^t) \geq 1$, observe that when $r_i = r_i^t$, $\delta_i^t(r_i^t) = 0$ and excludes current solution. Otherwise when $r_i \neq r_i^t$, $\delta_i^t(r_i^t) \geq 1$ holds.

6.1.2. Optimality Cut Generation

For the optimality cuts, there is a lower bound LB_i, on the subproblem's objective function value since we assume set of local constraints has bounded feasible region. LB_i can be found by solving (6.7) since local constraints are fixed and do not change with respect to common resource allocations.

LB_i

$$\text{Minimize} \sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{6.7a}$$

$$\text{subject to:} \sum_{j=1}^{n_i} b_{ij}x_{ij} \leq l_i \tag{6.7b}$$

$$x_{ij} \geq 0 \text{ and integer} \quad \forall j = 1, 2, ..., n_i \tag{6.7c}$$

Also assume that optimal objective function value of subproblem at iteration t is denoted by z_i^t. Recall that optimality cuts are generated when subproblem has a finite objective function value. Hence in this case, z_i^t exists. Then the following inequality gives the optimality cut:

$$q_i \geq z_i^t - \delta_i^t(r_i^t)\left[z_i^t - LB_i\right] \tag{6.8}$$

To verify its validity, observe that when $r_i = r_i^t$, then $\delta_i^t(r_i^t) = 0$ and master problem recovers the value of subproblem's objective function value. Otherwise, when $r_i \neq r_i^t$, the following inequality and (6.8) holds:

$$\delta_i^t(r_i^t)\left[z_i^t - LB_i\right] \geq \left[z_i^t - LB_i\right] \tag{6.9}$$

6.1.3. The Algorithm

(6.10) gives general appearance of master problem at any iteration after addition of possible feasibility or optimality cuts.

MP_i

$$\text{Minimize } \sum_{i=1}^{N} q_i \tag{6.10a}$$

$$\text{subject to: } \sum_{i=1}^{N} r_i^k = r^k \quad \forall k = 1, 2, \ldots, m \tag{6.10b}$$

$$r_i^k = \sum_{s=0}^{S} 2^s r_{is}^k \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{6.10c}$$

$$q_i \geq z_i - \delta_i(r_i)[z_i - LB_i] \quad \forall i = 1, 2, \ldots, N \tag{6.10d}$$

$$\delta_i(r_i) \geq 1 \quad \forall i = 1, 2, \ldots, N \tag{6.10e}$$

$$q_i \geq LB_i \quad \forall i = 1, 2, \ldots, N \tag{6.10f}$$

$$r_i^k \text{ unrestricted} \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \tag{6.10g}$$

$$r_{is}^k \in \{0, 1\} \quad \forall i = 1, 2, \ldots, N \quad \forall k = 1, 2, \ldots, K \quad \forall s = 1, 2, \ldots, S \tag{6.10h}$$

In integer programming case, local master problem decides common resource allocations and primal subproblem generates cuts. The main difference is in the cut generation strategy. Instead of Benders Cut, we generate Laporte and Louveaux (LL) cuts in this case since linear programming duality is lost. There is no change in the way of communication among OAs. We describe Decentralized Integer L-Shaped Method in Figure 6.1. It starts with a pair of problems $SP_i(r_i^k) - MP_i$ for each OA $i \in \{1, 2, ..., N\}$ and a strongly connected communication graph, $G = \{N, A\}$. Each OA i solves its local master problem, MP_i and gets r_i^k that denotes the amount of common resource k allocated to OA i (line 6). Decentralized Integer L-Shaped Method utilizes GETIPCUT function (line 8) which looks for a cut from neighbours recursively, if there is no cut

```
    Input: $SP_i(r_i^k)$ { Subproblem for OA $i$}
    Input: $MP_i$ {Local master problem for OA $i$}
    Input: $G = \{N, A\}$ {Strongly connected digraph}
1 {$r_i^k$ denotes the amount of common resource $k$ allocated to OA $i$}
2 {$V$ denotes the visited neighbours list for finding a new cut}
3 {$LL$ denotes the Laporte and Louveaux cut}
4 for $i = 1$ to $N$ do
5     repeat
6         Solve $MP_i \rightarrow (r_i^k)$
7         $V \leftarrow \varnothing$
8         $LL \leftarrow GetIPCut(i, V, r_i^k)$
9     until $LL = Null$
```

Figure 6.1. Pseudo-code for Decentralized Integer L-Shaped Method

generated by the OA itself. Hence, Decentralized Integer LShaped Method keeps track of the visited OAs with the list, V. It terminates when there is no new cut is generated for any of the OAs (line 9).

Figure 6.2 gives the pseudo-code for GetIPCut function that returns a Laporte and Louveaux cut. It adds the current OA to the list to avoid visiting it more than once (line 4). Then objective function of $SP_j(r_i^k)$ is updated with the given allocations and it is solved (line 5).

According to the solution status of $SP_j(r_i^k)$, either a Laporte and Louveaux feasibility cut or a Laporte and Louveaux optimality cut is generated (line 6, line 10). Otherwise, GetIPCut function looks for a new cut recursively from all neighbouring OAs (line 11, line 13) until it finds a new cut (line 14). If a new cut is generated then it is added to the local master problem of OA j (line 16). The algorithm for GetIPCut function runs until all OAs are visited (line 3). Note that, GetIPCut is a recursive function. Hence, by definition, when a cut is generated by OA j with respect to OA i's allocations through GetIPCut function, then all the OAs on the path connecting

OAs i and j adds that cut to their local master problem.

```
   Input: $j$ {Identity of OA}
   Input: $V$ {Visited Neighbours List}
   Input: $r_i^k \quad \forall k = 1, 2, \ldots, K$ {Allocation of common resource $k$ in OA $i$'s
          proposal}
   Output: $LL$ {Laporte and Louveaux Feasibility or Optimality cut}
1  {This is a recursive function that returns a LL cut, if it exists}
2  if $j \in V$ then
3      return $Null$
4  $V \leftarrow V \cup \{j\}$
5  Update objective function of $SP_j(r_i^k)$
6  Solve $SP_j(r_i^k) \rightarrow GetStatus$
7  if $GetStatus = Optimal$ then
8      $LL \leftarrow$ Generate LL optimality cut according to (6.8)
9  else if $GetStatus = Infeasible$ then
10     $LL \leftarrow$ Generate LL feasibility cut according to (6.6)
11 if $LL = Null$ then
12     forall $n \in N_j$ do
13         $LL \leftarrow$ GetIPCut$(n, V, r_i^k)$
14         if $LL \neq Null$ then
15             Break out of the For loop
16 Add $LL$ to $MP_j$
17 return $LL$
```

Figure 6.2. Pseudo-code for GetIPCut Function

6.2. Convergence

In this section, we prove the convergence of Decentralized Integer L-Shaped Algorithm in three parts. In the first part, we show that the number of LL cuts that can be generated is finite. The second part shows that each LL 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 if the underlying communication network is strongly connected. As a result, convergence of Decentralized Integer L-Shaped Algorithm follows.

PART I: There is a finite number of Laporte and Louveaux that can be generated. We state and prove the following proposition for this part.

Proposition 10. The number of LL cuts that can be generated in Decentralized Integer L-Shaped Algorithm is finite.

Proof. Assume that $r_i = \{r_i^1, r_i^2, \cdots, r_i^k\}$ denotes the amount of common resource k allocated to optimization agent i by (6.10). Then each $r_i^k = \{r_{i0}^k, r_{i1}^k, \cdots, r_{is}^k\}$ can be represented with a set of binary variables such that $r_i^k = 2^0 r_{i0}^k + 2^1 r_{i1}^k + 2^2 r_{i2}^k + \cdots + 2^s r_{is}^k$ where $r_{is}^k \in \{0, 1\}$. A feasibility cut or an optimality cut is generated in (6.6) or (6.8), respectively with respect to an unique partition I_i^t and Z_i^t of binary variables r_{is}^k used to represent r_i^k. If there are m binary variables in local master problem then there is at most 2^m available partitions for each variable. Hence one can conclude that there are finite number of cuts that can be generated in Decentralized Integer L-Shaped Algorithm. □

PART II: A unique Laporte and Louveaux cut is generated at each iteration. We state and prove the following proposition for this part.

Proposition 11. Decentralized Integer L-Shaped Algorithm yields an unique Laporte and Louveaux cut at each iteration.

Proof. We verify the validity of feasibility and optimality LL cuts. When a new cut is generated and added to the relaxed master problem, then relaxed master problem excludes the associated binary solution 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. We state and prove the following proposition for this part.

Proposition 12. Decentralized Integer L-Shaped Algorithm yields an optimal solution for BAIP (if one exists) within a finite number of iterations if communication network is strongly connected.

Proof. Assume a strongly connected communication network for exchanging cuts among OAs. Thus, any cut generated by any OA can be added to relaxed local master of any other OA in the network along the directed path via recursive GetIPCut function. Decentralized Integer L-Shaped Algorithm terminates when there is no new cut is generated for any one of OAs. Therefore, finite convergence of Decentralized Integer L-Shaped Algorithm for BAIP follows from Proposition 10 and Proposition 11. $\square$

7. APPLICATIONS AND COMPUTATIONAL RESULTS

In this chapter, 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. Also we apply Decentralized L-Shaped Method to solve small random instances and present preliminary results.

7.1. Numerical Experiments and Results for LP Case

7.1.1. Test Problems

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 [52]. 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 b_i are selected from [100,500]. Table 7.1 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.

Multi-commodity network flow (MNCF)problems are the second group of problems since they are one of the well-known problem types that represents primal block angular structure. MNCF problems simply deal with reducing the total cost of transporting commodities from their origins to destinations through a network with capacitated arcs. Some real-world application areas involve production planning, telecommunications and transportation/distribution. A general linear programming model formulation that obtains the minimum cost routing of a set of k commodities through

Table 7.1. Dimensions for Randomly Generated Problems.

Random Instance	# of blocks	# of variables in each block	# of rows in each block	# of complicating constraints	% of complicating constraints
pr2-500-200-100	2	500	200	100	20%
pr5-100-80-100	5	100	80	100	20%
pr10-50-40-100	10	50	40	100	20%
pr20-25-20-100	20	25	20	100	20%
pr50-10-8-100	50	10	8	100	20%
pr20-10-4-20	20	10	4	20	20%
pr20-20-8-40	20	20	8	40	20%
pr20-30-12-60	20	30	12	60	20%
pr20-40-16-80	20	40	16	80	20%
pr20-50-20-100	20	50	20	100	20%
pr5-30-20-25	5	30	20	25	20%
pr10-30-20-50	10	30	20	50	20%
pr20-30-20-100	20	30	20	100	20%
pr40-30-20-200	40	30	20	200	20%

a network given by a directed graph *G(N,A)* having n nodes and m arcs can be formulated as the following:

$$\text{Minimize} \sum_k \sum_{ij} c_{ij}^k x_{ij}^k \tag{7.1a}$$

$$\text{subject to:} \sum_j x_{ij}^k - \sum_j x_{ji}^k = b_i^k \quad \forall i, k \tag{7.1b}$$

$$0 \le x_{ij}^k \le u_{ij}^k \quad \forall i, j, k \tag{7.1c}$$

$$\sum_k x_{ij}^k \le u_{ij} \quad \forall i, j \tag{7.1d}$$

$$x_{ij}^k \ge 0 \tag{7.1e}$$

where node i and node $j \in N$ and arc $(i, j) \in A$. The decision variables x_{ij}^k define the flow of the commodity k on each arc *(i,j)*. c_{ij}^k denotes the unit cost for the commodity k on arc *(i,j)*. Clearly, (7.1) is a primal block angular problem. The blocks are defined by the constraint set (7.1b) that satisfies the flow conservation of commodity k at each node i and the constraint set (7.1c) that restricts the individual capacity of each arc for commodity k by u_{ij}^k. The constraint set (7.1d) are the complicating constraints since they link the commodities by limiting the total flow on each arc by mutual capacity

u_{ij}. Hence the overall problem has $k \times m$ variables and $k \times (m+n) + m$ constraints.

This research is not examining a specific real-world problem, hence we performed computational experiments on randomly generated problem instances. The problems are generated by random generator *Mnetgen* that can be retrieved from *http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html.* We used C++ service class *Graph* to convert the problems to MPS format.

Table 7.2. Characteristics for Mnetgen Instances

Instances	M*.*.1-2-3	M*.*.4-5-6	M*.*.7-8-9	M*.*.10-11-12
Density	Sparse	Sparse	Dense	Dense
Complicating constraints	40%	80%	40%	80%

Mnetgen generator is one of the well-known multi-commodity network flow problem generators which is developed by Kennington and Ali [53] and revised by Frangioni [54, 55]. The set of problems generated by Mnetgen can be characterized by the number of nodes n where $n \in \{64, 128, 256\}$ and the number of commodities k where $k \in \{4, 8, 16, ..., n\}$. In our context, n is the number of local constraints in each block while k is the number of blocks. 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* with 40% of the arcs have mutual capacity constraints (ie, complicating constraints). The other three problems are *hard* since 80% of the arcs have mutual capacity constraints. Table 7.2 summarizes the problem characteristics of each set of the problems.

Mnetgen generator provides a large set of test instances. Table 7.3 shows dimensions of the Mnetgen instances that we have used in our experiments.

Table 7.3. Dimensions of Mnetgen Instances

(a) M64.4.* instances

Mnetgen Instance	# of complicating constraints	Problem Size
M64.4.1	82	338x720
M64.4.2	63	319x720
M64.4.3	32	288x720
M64.4.4	140	396x720
M64.4.5	136	392x720
M64.4.6	56	312x720
M64.4.7	184	440x1530
M64.4.8	161	417x1482
M64.4.9	69	325x1517
M64.4.10	391	647x1534
M64.4.11	307	563x1529
M64.4.12	149	405x1521

(b) M64.8.* instances

Mnetgen Instance	# of complicating constraints	Problem Size
M64.8.1	81	593x1432
M64.8.2	67	579x1433
M64.8.3	56	568x1434
M64.8.4	147	659x1434
M64.8.5	152	664x1437
M64.8.6	87	599x1436
M64.8.7	226	738x2991
M64.8.8	199	711x2984
M64.8.9	118	630x2970
M64.8.10	428	940x2998
M64.8.11	377	889x2970
M64.8.12	260	772x2973

(c) M128.32.* instances

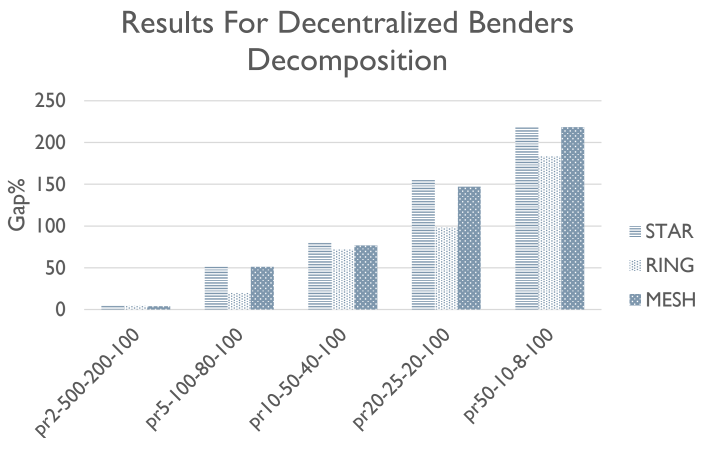
Mnetgen Instance	# of complicating constraints	Problem Size
M128.32.1	394	4488x11538
M128.32.2	398	4492x11538
M128.32.3	403	4497x11521
M128.32.4	413	4507x11527
M128.32.5	412	4508x11527
M128.32.6	413	4507x11525

7.1.2. Results and Discussion

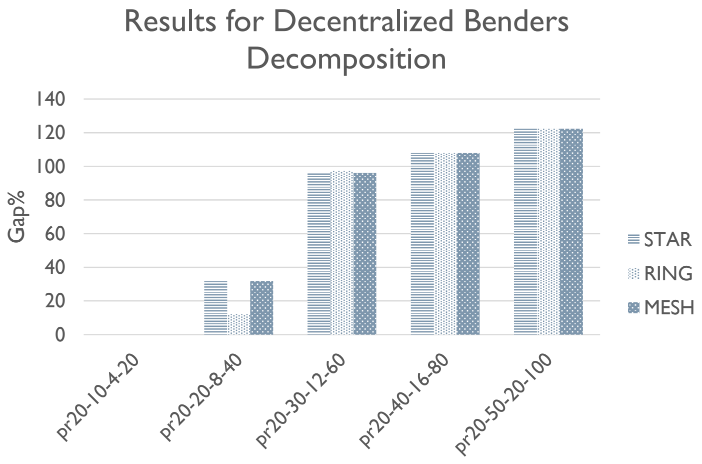
In this section, we present computational results performed on randomly generated generic problems and Multi-commodity network flow problems. We have implemented the Decentralized Benders Decomposition and Decentralized Dantzig-Wolfe Decomposition algorithms with C# utilizing CPLEX 12.5.1.0 running on a Windows7 PC with a 2.6 Turbo GHz CPU and 4 GB RAM. For the problems in Table 7.1, we generate five instances randomly for each problem type and report the average as the result. We use Star, Ring and Mesh topologies in Figure 3.1 as strongly connected communication graph among OAs. Our algorithms designed as if each OA solves BALP

Table 7.4. Results for Decentralized Benders Decomposition on Random Set

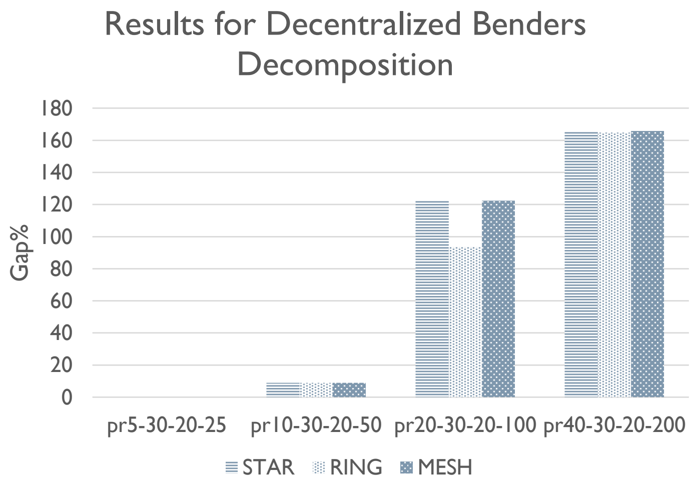
Random Instance	Aggregate Obj. Fn. Value	STAR				RING				MESH			
		LB	UB	GAP %	CPU Time (s)	LB	UB	GAP %	CPU Time (s)	LB	UB	GAP %	CPU Time (s)
pr2-500-200-100	292.73	281.22	296.51	5.22	7200	281.22	294.42	4.50	7200	284.72	296.72	4.03	7200
pr5-100-80-100	776.96	758.73	1156.31	51.17	7200	758.73	913.39	19.90	7200	758.73	1156.31	51.17	7200
pr10-50-40-100	1535.00	1469.35	2696.85	79.96	7200	1469.35	2576.85	72.14	7200	1469.35	2648.42	76.81	7200
pr20-25-20-100	3066.16	2984.32	7736.83	154.99	7200	2984.32	5993.92	98.15	7200	2984.32	7490.82	146.97	7200
pr50-10-8-100	7416.69	7389.58	23587.04	218.39	7200	7389.58	21000.68	183.51	7200	7389.58	23587.04	218.39	7200
pr20-10-4-20	2170.44	2170.44	2170.44	0	1585.85	2170.44	2170.44	0	1040.42	2170.44	2170.44		4699.85
pr20-20-8-40	2257.37	2032.35	2752.59	31.90	7200	2153.29	2427.62	12.15	7200	2032.35	2752.59	31.90	7200
pr20-30-12-60	2406.45	2169.42	4494.05	96.59	7200	2169.42	4511.44	97.32	7200	2169.42	4480.24	96.02	7200
pr20-40-16-80	2510.25	2138.83	4848.40	107.94	7200	2138.83	4848.40	107.94	7200	2138.83	4848.40	107.94	7200
pr20-50-20-100	2498.82	2276.55	5335.55	122.41	7200	2276.55	5335.55	122.41	7200	2276.55	5335.55	122.41	7200
pr5-30-20-25	732.75	732.75	732.75	0	230.25	732.75	732.75	0	196.46	732.75	732.75	0	229.54
pr10-30-20-50	1397.83	1369.66	1493.95	08.89	7200	1369.66	1493.95	8.89	7200	1369.66	1493.95	8.89	7200
pr20-30-20-100	2950.30	2852.13	6463.40	122.40	7200	2852.13	5610.44	93.49	7200	2852.13	6463.40	122.40	7200
pr40-30-20-200	5642.89	5519.44	14880.13	165.88	7200	5519.44	14820.44	164.82	7200	5519.44	14820.44	165.88	7200



(a) Random set 1



(b) Ramdom set 2



(c) Random set 3

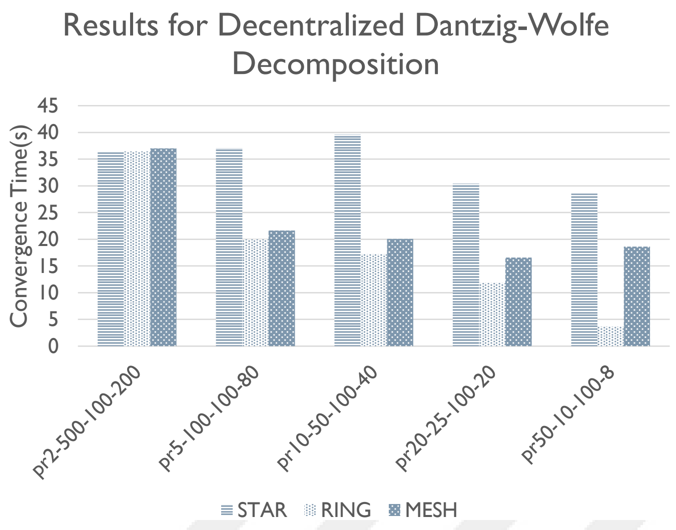
Figure 7.1. Results for Decentralized Benders Decomposition on Random Set

for itself since a central coordination unit is lacking. Hence we allow equal run time for each OA that sum up to two hours for each problem. We find lower and upper bound on the objective function value if the algorithm does not converge to optimal solution within the allowed time and report the percent of the gap.

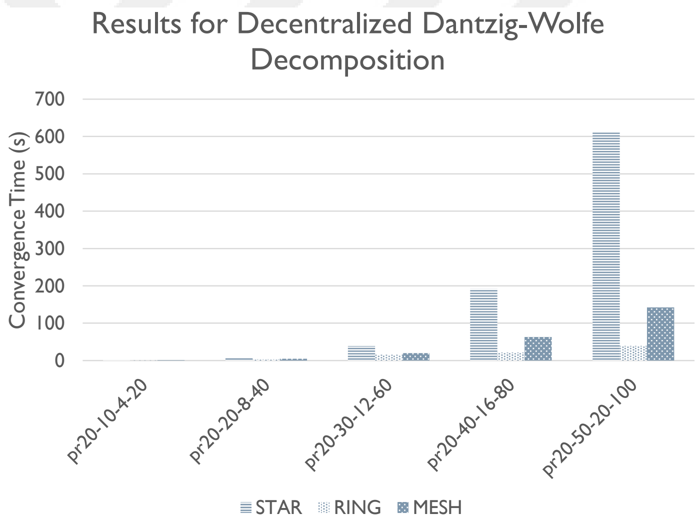
Table 7.4 presents results for first group of problems on Decentralized Benders Decomposition. For the first set of problems, the size of the overall problem is fixed. Thus, subproblem size becomes smaller as the number of OAs increases. However, *Gap%* increases with the number of OAs because of allowing 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 result 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. Figure 7.1 illustrates respective *Gap%* Decentralized Benders Decomposition on random set of problems.

Table 7.5. Results for Decentralized Dantzig-Wolfe Decomposition on Random Set

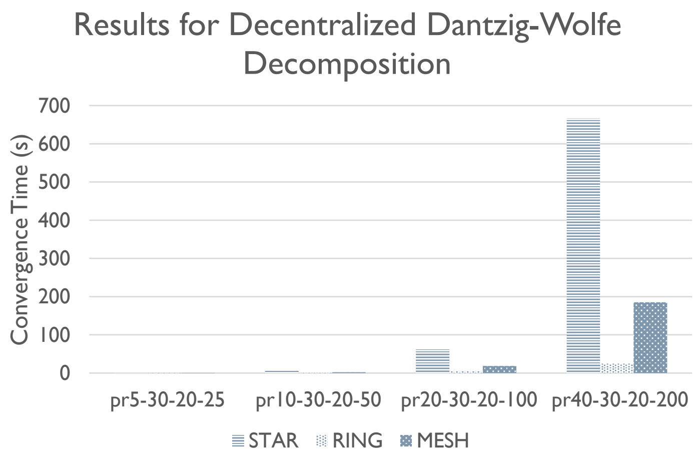
Random Instance	Obj. Fn. Value	CPU Time (s)		
		STAR	RING	MESH
pr2-500-200-100	292.73	36.29	36.47	37.02
pr5-100-80-100	776.96	37.02	19.98	21.65
pr10-50-40-100	1535.00	39.63	17.21	20.06
pr20-25-20-100	3066.16	30.42	11.86	16.58
pr50-10-8-100	7416.69	28.55	3.65	18.63
pr20-10-4-20	2170.44	1.58	0.38	1.06
pr20-20-8-40	2560.39	7.65	2.78	4.80
pr20-30-12-60	2406.45	38.84	15.62	19.67
pr20-40-16-80	2510.25	222.02	40.48	74.78
pr20-50-20-100	2498.82	611.71	38.91	142.01
pr5-30-20-25	732.75	0.49	0.22	0.30
pr10-30-20-50	1397.83	5.69	0.97	2.16
pr20-30-20-100	2950.30	61.75	4.84	18.49
pr40-30-20-200	5642.89	666.40	24.64	185.32



(a) Random set 1



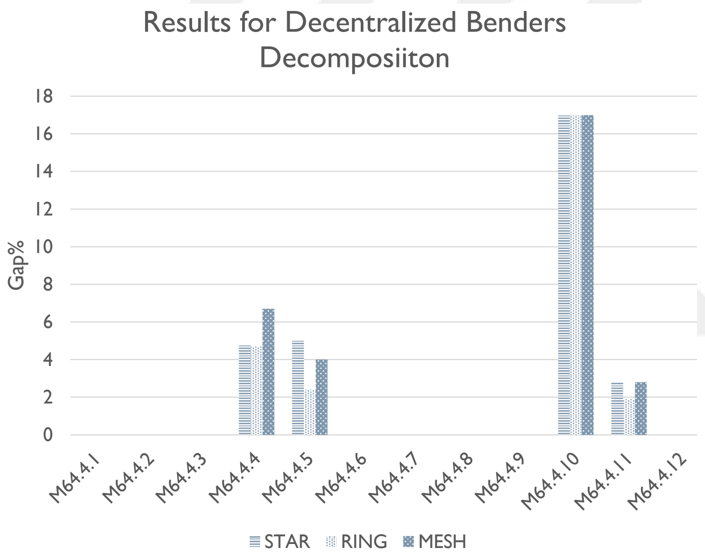
(b) Random set 2



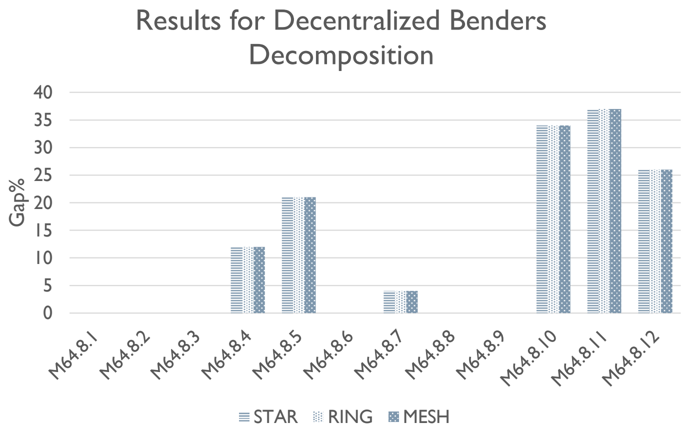
(c) Random set 3

Figure 7.2. Results for Decentralized Dantzig-Wolfe Decomposition on Random set

Table 7.5 and Figure 7.2 present 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 first as the subproblems are getting easier to solve, however more OAs results in higher convergence time.



(a) M64.4.* Instances



(b) M64.8.* Instances

Figure 7.3. Results for Decentralized Benders Decomposition on Mnetgen Instances

Table 7.6. Results for Decentralized Benders Decomposition on M64.4.* Instances

Instance	**Aggregate 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	290806.3	290806.3	290806.3	-	243	290806.3	290806.3	-	219	290806.3	290806.3	-	**208**
M64.4.2	336019.9	336019.9	336019.9	-	60	336019.9	336019.9	-	65	336019.9	336019.9	-	**46**
M64.4.3	348966.6	348966.6	348966.6	-	**0.01**	348966.6	348966.6	-	0.08	348966.6	348966.6	-	0.88
M64.4.4	412475.85	393222.5	413107.3	4.8%	7200	393222.5	412620.5	**4.7%**	7200	395745	423484.6	6.7%	7200
M64.4.5	390578.57	377971.8	397764.6	5%	7200	382524.4	391967.9	**2.4%**	7200	376909.8	392703.9	4%	7200
M64.4.6	506554.45	506554.4	506554.4	-	**7.59**	506554.4	506554.4	-	8.2	506554.4	506554.4	-	9.55
M64.4.7	147862.16	147862.1	147862.1	-	160	147862.1	147862.1	-	**116**	147862.1	147862.1	-	135
M64.4.8	165185.35	165185,3	165185.3	-	203	165185.3	165185.3	-	440	165185.3	165185.3	-	**191**
M64.4.9	192119.41	192119.4	192119.4	-	3.48	192119.4	192119.4	-	**2.06**	192119.4	192119.4	-	2.26
M64.4.10	167479.52	165710	195775	**17%**	7200	165710	195775	**17%**	7200	165710	195775	**17%**	7200
M64.4.11	193238.44	192338.7	197839.7	2.8%	7200	192338.7	196203.2	**1.9%**	7200	192338.7	197863.5	2.8%	7200
M64.4.12	192400.135	192400.1	192400.1	-	351	192400.1	192400.1	-	**312**	192400.1	192400.1	-	320

Table 7.7. Results for Decentralized Benders Decomposition on M64.8.* Instances

Instance	**Aggregate 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	622280.4	622280.4	622280.4	-	1282	622280.4	622280.4	-	**329**	622280.4	622280.4	-	558
M64.8.2	649767	649767	649767	-	979	649767	649767	-	**232**	649767	649767	-	663
M64.8.3	750938	750938	750938	-	53.27	750938	750938	-	**34.54**	750938	750938	-	44.33
M64.8.4	761862.75	730880.7	835831.6	12%	7200	730880.7	835831.6	12%	7200	730880.7	835831.6	12%	7200
M64.8.5	753927.6	728444.4	926353.4	21%	7200	728444.4	926353.4	21%	7200	728444.4	926353.4	21%	7200
M64.8.6	929066.8	929066.8	929066.8	-	7193	929066.8	929066.8	-	**3995**	929066.8	929066.8	-	5447
M64.8.7	304045	293987.6	306481.5	4%	7200	296811.6	306481.5	4%	7200	293705.6	306481.5	4%	7200
M64.8.8	355699.7	355699.7	355699.7	-	5280	355699.7	355699.7	-	4355	355699.7	355699.7	-	**3960**
M64.8.9	357649.1	357649.1	357649.1	-	2123	357649.1	357649.1	-	**700**	357649.1	357649.1	-	1499
M64.8.10	361802	357717.8	542878	34%	7200	357225.7	542878	34%	7200	356868	542878	34%	7200
M64.8.11	418824.03	405916.3	652132.5	37%	7200	407577.5	652132.5	37%	7200	407685.4	652132.5	37%	7200
M64.8.12	394051	381944.8	522477.6	26%	7200	381941.4	522477.6	26%	7200	381944.8	522477.6	26%	7200

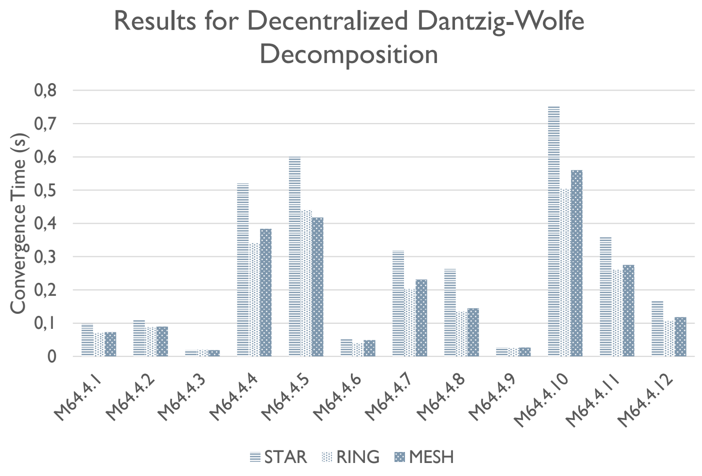
There is an increase in convergence time for the second set and the third set. However, the algorithm reacts more to size of the subproblems than the number of agents for all topology types. While Ring topology outperforms the others, Mesh topology converges faster than Star topology.

Table 7.6 and Table 7.7 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. Figures 7.3a and 7.3b illustrate the *Gap*% for M64.4.* and M64.8.*, respectively. The results shows that there is not a clear dominance of any topology to the others for M64.4.* problems. However, in most of the problems Star topology has longer convergence time than the others. For M64.8.* instances, Ring topology needs less computational time to convergence.

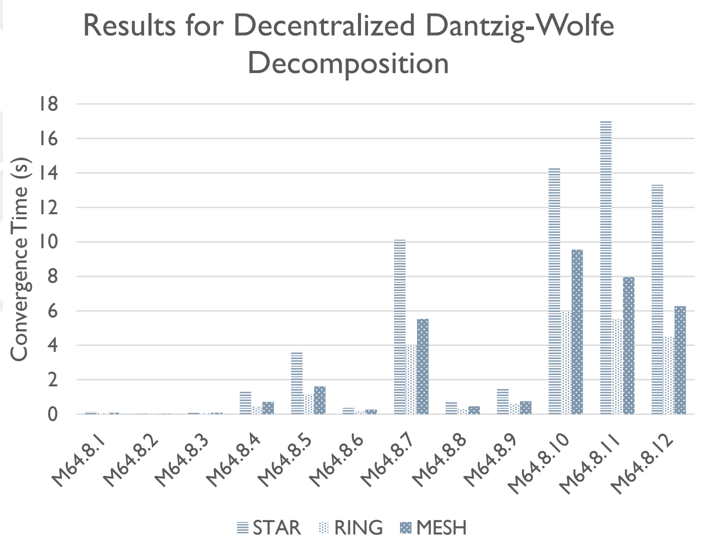
Table 7.8 and Table 7.9 show that Decentralized Dantzig-Wolfe Decomposition method converges to optimal solution under one second for M64.4.* instances (See Figure 7.4a) and within seconds for M64.8.* instances (See Figure 7.4b). While there is not a clear dominance of any topology for M64.4.* instances, Ring topology has less computational time than the others.

Table 7.8. Results for Decentralized Dantzig-Wolfe Decomposition on M64.4.* Instances

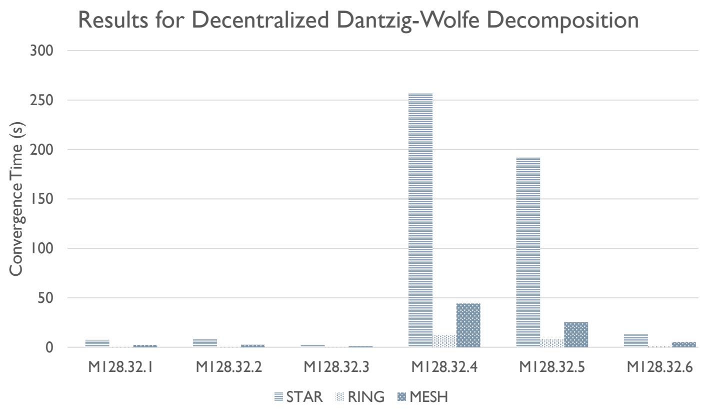
Mnetgen Instance	**Aggregate Obj. Fn. Value**	**STAR**	**RING**	**MESH**
		CPU Time (s)	CPU Time (s)	CPU Time (s)
M64.4.1	290806.3	0.098	**0.071**	0.073
M64.4.2	336019.9	0.111	**0.088**	0.09
M64.4.3	348966.6	**0.019**	0.020	**0.019**
M64.4.4	412475.85	0.52	**0.341**	0.384
M64.4.5	390578.57	0.603	0.44	**0.418**
M64.4.6	506554.45	0.053	**0.04**	0.049
M64.4.7	147862.16	0.319	**0.203**	0.231
M64.4.8	165185.35	0.263	**0.135**	0.145
M64.4.9	192119.41	0.027	**0.025**	0.027
M64.4.10	167479.52	0.752	**0.504**	0.56
M64.4.11	193238.44	0.358	**0.261**	0.275
M64.4.12	192400.135	0.168	**0.107**	0.118



(a) M64.4.* Instances



(b) M64.8.* Instances



(c) M128.32.* Instances

Figure 7.4. Results for Decentralized Dantzig-Wolfe Decomposition on Mnetgen Instances

Table 7.9. Results for Decentralized Dantzig-Wolfe Decomposition on M64.8.* Instances

Mnetgen Instance	**Aggregate Obj. Fn. Value**	**STAR**	**RING**	**MESH**
		CPU Time (s)	CPU Time (s)	CPU Time (s)
M64.8.1	622280.4	0.158	**0.07**	0.09
M64.8.2	649767	**0.027**	0.028	0.029
M64.8.3	750938	0.135	**0.069**	0.091
M64.8.4	761862.75	1.83	**0.84**	1.09
M64.8.5	753927.6	3.66	**1.166**	1.62
M64.8.6	929066.8	0.418	**0.171**	0.275
M64.8.7	304045	10.13	**3.988**	5.54
M64.8.8	355699.7	0.757	**0.304**	0.46
M64.8.9	357649.1	1.506	**0.613**	0.752
M64.8.10	361802	14.28	**5.93**	9.55
M64.8.11	418824.03	17.06	**5.53**	7.96
M64.8.12	394051	13.31	**4.49**	6.27

Table 7.10 and Figure 7.4c present results for a larger test instance of M128.32.* instances. The affect of topology type becomes clear for larger problem sets. Ring topology performs best while Star topology needs more time for convergence. Performance of Mesh topology is in between of this two topologies.

Table 7.10. Results for Decentralized Dantzig-Wolfe Decomposition on M128.32.* Instances

Mnetgen Instance	**Aggregate Obj. Fn. Value**	**STAR**	**RING**	**MESH**
		CPU Time (s)	CPU Time (s)	CPU Time (s)
M128.32.1	11186573.89	465	**29**	164
M128.32.2	118663936.61	492	**32**	170
M128.32.3	12476676.85	201	**17**	80
M128.32.4	12715040.26	-	**756**	2659
M128.32.5	13582810.69	-	**520**	1506
M128.32.6	14617437.18	787	**69**	335

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

Comparison of the Methods: We observe that Decentralized Dantzig-Wolfe Decomposition outperforms Decentralized Benders Decomposition in all instances. One reason for this is the primal block angular structure of the problem that we address.

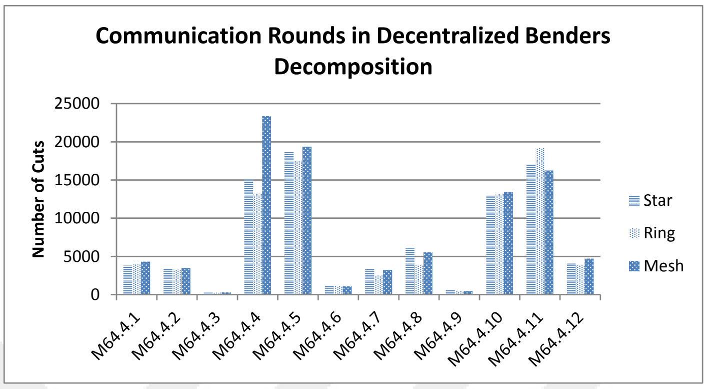
Benders Decomposition can be best applied to the problems in dual block angular structure. Thus, we introduce a variable for each OA for a single common resource to convert the primal BALP problem to a dual one. Hence the complicating constraints in local master problem and local constraint set in each subproblem gets larger hence becomes harder to solve.

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. Hence This results in longer convergence time.

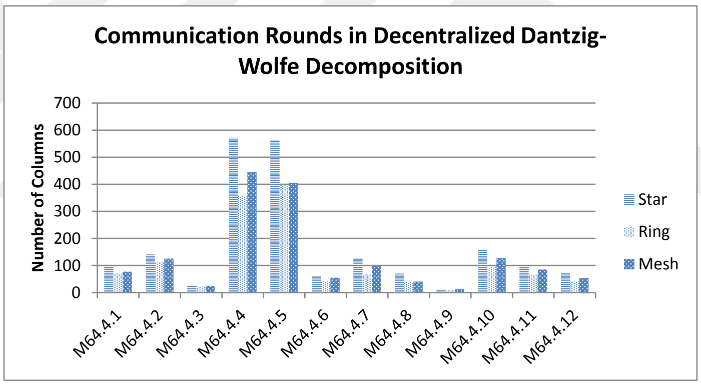
Number of Blocks: We observe that as the number of blocks increase, the time until termination gets longer. This can be explained mainly with the fact that each block is associated with an OA. Both methods solve the overall problem for each OA before termination. Hence as the number of blocks increase, 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 better 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 quick convergence than Star topology almost in all instances. This result holds for Decentralized Benders Decomposition also. Although Mesh topology can outperform Ring topology on small problems, for the larger problems Ring topology converges fast. The reason for this is the cut/column exchange strategy of the proposed methods. While OA i gets cut/column from a neighbour j, both agents i and j adds that cut or column to its local master problem. Thus, while getting a cut/column from the further neighbour, that cut/column is added all the OAs in the path connecting two communicating OAs. In Ring topology, an OA i can reach any other OA j indirectly. Since the length of the path can be largest in Ring topology, any cut/column can be added to the local master of more OAs at a time. This results in fast convergence rate.



(a) Decentralized Benders Decomposition



(b) Decentralized Dantzig-Wolfe Decomposition

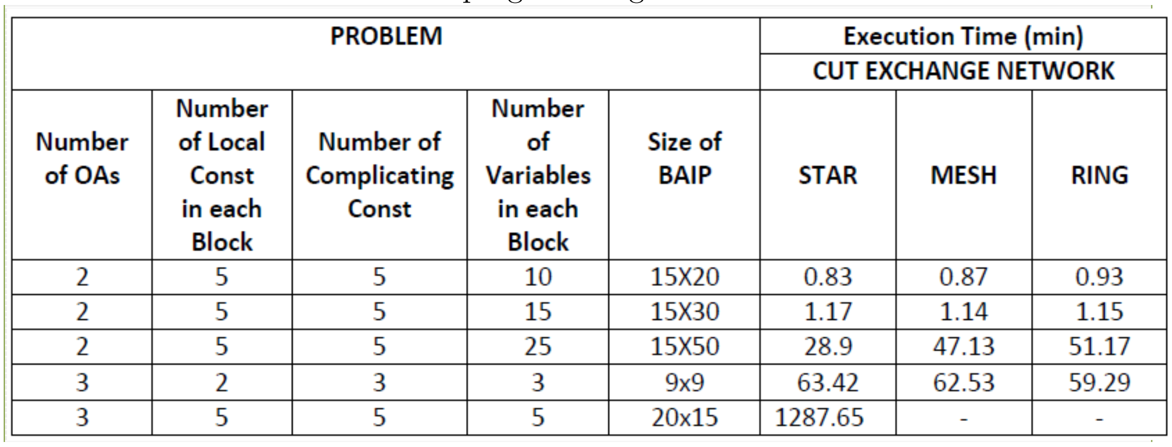
Figure 7.5. Communication Rounds for Mnetgen M64.4.* Instances

Cut/column exchange strategy also affects the number of communication rounds among OAs. To give an example, Figure 7.5 and Figure 7.5b illustrate 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/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/column to the local master of all OAs in the path connecting two communicating OAs. In an iteration a cut/column may be added local master problem of more OAs in Ring topology and this results in faster convergence. In most of the problems, Mesh topology generates less cuts/columns than star topology which confirms the speed of convergence of these methods.

7.2. Numerical Experiments and Preliminary Results for IP Case

We have implemented the Decentralized Integer L-Shaped Method with C# using CPLEX 12.5.1.0 running on a Windows7 PC with a 2.6 Turbo GHz CPU and 4 GB RAM. To get preliminary results, we generated random test instances as we did previously for linear programming case by using the same strategy as authors did in [36]. In integer programming case, we assumed that all variables are integer. We solved problems by using three different cut exchange topologies, Ring, Mesh and Star. Table 7.11 presents the dimension of the test instances and the convergence rate. Convergence of Decentralized Integer L-Shaped Method for integer programming case is very slow. Consider a randomly generated small problem having two complicating constraints and two blocks of local constraints each has two constraint with two variables. Solving such a problem by proposed algorithm assuming star communication network topology requires eighteen iterations where at each iteration, master problem determines common resource allocation for subproblems and subproblems generate a cut with respect to the given allocations.

Table 7.11. Dimensions of Test Instances and Preliminary Results for Integer programming case



PROBLEM					Execution Time (min)		
					CUT EXCHANGE NETWORK		
Number of OAs	Number of Local Const in each Block	Number of Complicating Const	Number of Variables in each Block	Size of BAIP	STAR	MESH	RING
2	5	5	10	15X20	0.83	0.87	0.93
2	5	5	15	15X30	1.17	1.14	1.15
2	5	5	25	15X50	28.9	47.13	51.17
3	2	3	3	9x9	63.42	62.53	59.29
3	5	5	5	20x15	1287.65	-	-

The main reason for this is the cut generation strategy. For IP case, subproblems are integer programs hence linear programming duality is lost. Thus, instead of Benders cuts we generate Laporte and Louveaux cuts. While Benders cuts can exclude more, Laporte and Louveaux cuts excludes only one solution value at each iteration. Searching an optimal solution by eliminating non-optimal ones one by one results in slow convergence.

Another reason for slow convergence in IP case is the use of auxiliary variables. In this case, we represent integer variables as binary variables. Introducing a set of binary variables for each integer variable makes the integer programming problem larger. Solving a larger integer program at each iteration leads to longer convergence time. On the other hand, we can conclude that the number of decision makers and the number of variables also have a significant effect on slow convergence.

8. CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we propose novel decentralized decomposition methods for large-scale linear and integer block angular problems: Decentralized Benders Decomposition and Decentralized Dantzig-Wolfe Decomposition for BALP and Decentralized L-Shaped Method for BAIP. Our methods allow multiple decision makers to cooperate while reaching global optimum without need of a central coordination, but by partial information sharing among one another through a strongly connected communication network. We remark that, our goal is not competing with the computational speed of a centralized algorithm. Instead, we primarily aim to propose decentralized solution approach for decision makers that are unwilling to disclose their local data, but they want to solve the global problem collaboratively for a mutual benefit, in which case centralized approach does not work.

From an organizational point of view, in Decentralized Benders Decomposition, local master problem shares allocation of common resources. In return, the 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, the subproblem generates a column indicating explicit information that disclose optimal way of using common resources in terms of cost, profit or specific proposals.

We prove that the Decentralized Benders Decomposition and Decentralized Dantzig-Wolfe Decomposition can reach same global optimal solution with centralized methods in a finite number of iterations. We confirm theoretical results with computational experiments. We apply proposed methods to block angular linear problem instances that are randomly generated by using a similar approach to that in [36]. Also we evaluate the computational performance of the proposed methods on Multi-commodity Network Flow Problems which are known to show the block angular structure. The test sets are generated by publicly available random generators. We observe that Decentralized Benders Decomposition shows slower convergence rate than Decentralized Dantzig-Wolfe decomposition. However, Decentralized Benders Decomposition gives lower and upper bounds for the optimum solution whenever it is terminated. Nevertheless, a follow-up work will be accelerating the convergence of Decentralized Benders Decomposition. For solving larger realistic models, hybrid methods combining heuristics and decomposition methods may give promising results.

The main result of this research is the following. There are three 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 fast but requires a central coordination unit having full access to managerial information of the 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 decision making is rarely applied to integer programs in literature. In this thesis, we represent an extension of Decentralized Benders Decomposition to integer programming programs in block angular structure. We get preliminary results with randomly generated small test instances. We intend to explore decentralization in integer programs further. On the other hand, decentralized methods for optimization problems are inherently suited to parallel or distributed computing opportunity. Subproblem-local master problem pair for each decision maker can be solved in parallel computers. Thus, for future research direction, parallelization of proposed algorithms will be desirable.

Internet and communication technology hold significant potential for individual decision makers to collaborate. Another interesting research area may be decentralized decision making in an environment where the problem data is continuously evolving on a dynamic internet based communication scheme.

REFERENCES

1. Bradley, S. P., A. C. Hax and T. L. Magnanti, *Applied Mathematical Programming*, 1977.

2. Benders, J., “Partitioning procedures for solving mixed-variables programming problems”, *Numerische Mathematik*, Vol. 4, pp. 238–252, 1962.

3. Dantzig, G. B. and P. Wolfe, “Decomposition Principle for Linear Programs”, *Operations Research*, Vol. 8, No. 1, pp. 101–111, January 1960.

4. Rosen, J. B., “Primal partition programming for block diagonal matrices”, *Numerische Mathematik*, Vol. 6, No. 1, pp. 250–260.

5. Laporte, G. and F. V. Louveaux, “The Integer L-shaped Method for Stochastic Integer Programs with Complete Recourse”, *Oper. Res. Lett.*, Vol. 13, No. 3, pp. 133–142, April 1993.

6. Lübbecke, M. E., “Column generation”, *Wiley Encyclopedia of Operations Research and Management Science*, Vol. 82, October 2010.

7. Mokhtar S. Bazaraa, J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows*, Wiley, 2 edn., 1990.

8. Stoyan J. Stephen Dessouky, M. M. and X. Wang, “Introduction to large scale linear programming and applications”, *Wiley Encyclopedia of Operations Research and Management Science*, 1, 2010.

9. Bertsimas D, J. N., Tsitsiklis, *Introduction to Linear Optimization*, 1997.

10. Chung, W., “Dantzig-Wolfe Decomposition”, *Wiley Encyclopedia of Operations Research and Management Science*, Vol. 22, January 2010.

11. “Generalized Benders Decomposition”, *Journal of Optimization Theory and Applications*, Vol. 10, No. 4, 1972.

12. Infanger, G., *Planning Under Uncertainty: Solving Large-scale Stochastic Linear Programs*, Professional Practices Pamphlet.

13. Taşkın, Z. C., “Benders decomposition”, *Wiley Encyclopedia of Operations Research and Management Science*, Vol. 3, March 2010.

14. Lim, C., “Relationship among Benders, Dantzig-Wolfe, and Lagrangian Optimization”, *Wiley Encyclopedia of Operations Research and Management Science*, Wiley, 2010.

15. “The traveling-salesman problem and minimum spanning trees: Part II”, *Mathematical Programming*, Vol. 1, No. 1, 1971.

16. Fisher, M. L. and J. F. Shapiro, “Constructive Duality in Integer Programming”, *SIAM Journal on Applied Mathematics*, Vol. 27, No. 1, pp. 31–52, 1974.

17. Geoffrion, A. M., *Lagrangean relaxation for integer programming*, Springer, 1974.

18. Shapiro, J. F., “Generalized Lagrange Multipliers in Integer Programming”, *Operations Research*, Vol. 19, No. 1, pp. 68–76, 1971.

19. “Lagrangean relaxation”, *Top*, Vol. 11, No. 2, 2003.

20. Desai, J., “Lagrangian Optimization for LP”, *Wiley Encyclopedia of Operations Research and Management Science*, 1, Wiley Online Library, Hoboken, NJ, USA, Jun 2010.

21. Kornai, J. and T. Lipták, “Two-Level Planning”, *Econometrica*, Vol. 33, No. 1, pp. 141–169, 1965.

22. Kate, A. T., "Decomposition of Linear Programs by Direct Distribution", *Econometrica*, Vol. 40, No. 5, pp. 883–898, 1972.

23. Holmberg, K., *Primal and Dual Decomposition as Organizational Design : Price and / or Resource Directive Decomposition*, Tech. rep., Linköping Institute of Technology, 1996.

24. Obel, B., "A note on mixed procedures for decomposing linear programming problems", *Mathematische Operationsforschung und Statistik. Series Optimization*, Vol. 9, No. 4, pp. 537–544, 1978.

25. "Horizontal mixed decomposition", *European Journal of Operational Research*, Vol. 27, No. 1, pp. 25 – 33, 1986.

26. Roy, T. J., "Cross decomposition for mixed integer programming", *Mathematical Programming*, Vol. 25, No. 1, pp. 46–63, January 1983.

27. Holmberg, K., "Linear mean value cross decomposition: A generalization of the Kornai-Liptak method", *European Journal of Operational Research*, Vol. 62, No. 1, pp. 55–73, 1992.

28. Hong, Y., *Privacy-preserving collaborative optimization*, Ph.D. Thesis, Rutgers, The State University of New Jersey, 10 2013.

29. Weeraddana, P., G. Athanasiou, C. Fischione and J. Baras, "Per-se Privacy Preserving Solution Methods Based on Optimization", *52nd IEEE Conference on Decision and Control*, pp. 206–211, 2013.

30. Li, J. and M. J. Atallah, "Secure and Private Collaborative Linear Programming", *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1–8, Nov 2006.

31. Toft, T., *Financial Cryptography and Data Security: 13th International Confer-*

ence, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers, chap. Solving Linear Programs Using Multiparty Computation, pp. 90–107, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

32. Bednarz, A., *Methods for Two-Party Privacy-Preserving Linear Programming*, Phd thesis, Discipline of Applied Mathematics, School of Mathematical Sciences, The University of Adalaide, 2012.

33. Goldreich, O., *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge University Press, New York, NY, USA, 2004.

34. Du, W., *A Study of Several Specific Secure Two-party Computation Problems*, Ph.D. Thesis, West Lafayette, IN, USA, 2001, aAI3043719.

35. Mangasarian, O. L., "Privacy-preserving linear programming", *Optimization Letters*, Vol. 5, No. 1, pp. 165–172, 2011.

36. Hong, Y., J. Vaidya and H. Lu, "Efficient distributed linear programming with limited disclosure", *Data and Applications Security and Privacy XXV*, pp. 170–185, Springer, 2011.

37. Bednarz, A., N. Bean and M. Roughan, "Hiccups on the Road to Privacy-preserving Linear Programming", *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*, WPES '09, pp. 117–120, 2009.

38. Hong, Y., S. Goel and W. M. Liu, "An efficient and privacy-preserving scheme for P2P energy exchange among smart microgrids", *International Journal of Energy Research*, Vol. 40, No. 3.

39. Vaidya, J., "Privacy-preserving Linear Programming", *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, pp. 2002–2007, 2009.

40. van de Panne, C., "Decentralization for Multidivision Enterprises", *Operations*

Research., Vol. 39, No. 5, pp. 786–797, September 1991.

41. Schleicher, S., “Decentralized optimization of linear economic systems with minimum information exchange of the subsystems”, *Zeitschrift für Nationalökonomie*, Vol. 31, No. 1, pp. 33–44.

42. Jeong, I.-J. and V. J. Leon, “Distributed allocation of the capacity of a single-facility using cooperative interaction via coupling agents”, *International Journal of Production Research*, Vol. 41, No. 1, pp. 15–30, 2003.

43. “A hybrid solution to collaborative decision-making in a decentralized supply-chain”, *Journal of Engineering and Technology Management*, Vol. 29, No. 1, pp. 95 – 111, 2012.

44. Dutta, H. and H. Kargupta, “Distributed Linear Programming and Resource Management for Data Mining in Distributed Environments”, *IEEE International Conference on Data Mining Workshops, 2008. ICDMW ’08.*, pp. 543–552, 2008.

45. Bürger, M., G. Notarstefano and F. Allgöwer, “Locally constrained decision making via two-stage distributed simplex”, *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 5911–5916, Dec 2011.

46. “A distributed simplex algorithm for degenerate linear programs and multi-agent assignments”, *Automatica*, Vol. 48, No. 9, pp. 2298 – 2304, 2012.

47. “Coordinating decentralized linear programs by exchange of primal information”, *European Journal of Operational Research*, Vol. 247, No. 3, pp. 788 – 796, 2015.

48. Poundarikapuram, S. and D. Veeramani, “Distributed Decision-Making in Supply Chains and Private E-Marketplaces”, *Production and Operations Management*, Vol. 13, No. 1.

49. Jeong, I.-J. and V. J. Leon, “A single-machine distributed scheduling methodology

using cooperative interaction via coupling agents", *IIE Transactions*, Vol. 37, No. 2, pp. 137–152, 2005.

50. Martin, R., *Large Scale Linear and Integer Optimization: A Unified Approach.*

51. Kalvelagen, E., "Column generation with GAMS", *GAMS Development Corp., Washington DC*, 2003.

52. Hong, Y., J. Vaidya and H. Lu, "Secure and Efficient Distributed Linear Programming", *Journal of Computer Security*, Vol. 20, No. 5, pp. 583–634, September 2012.

53. Ali, A. and J. Kennington, *Mnetgen program documentation*, Tech. rep., Department of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, TX, 1977.

54. Cappanera, P. and A. Frangioni, "Symmetric and Asymmetric Parallelization of a Cost-Decomposition Algorithm for Multicommodity Flow Problems", *INFORMS Journal on Computing*, Vol. 15, No. 4, pp. 369–384, 2003.

55. Frangioni, A. and G. Gallo, "A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min-Cost Flow Problems", *INFORMS Journal on Computing*, Vol. 11, No. 4, pp. 370–393, 1999.