View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by D-Scholarship@Pitt

Optimization via Benders' Decomposition

by

Hiruni Kamali Pallage

B.Sc., University of Sri Jayewardenepura, 2014

Submitted to the Graduate Faculty of

the Kenneth P. Dietrich School of Arts and Sciences in partial

fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2019

UNIVERSITY OF PITTSBURGH

KENNETH P. DIETRICH SCHOOL OF ARTS AND SCIENCES

This thesis was presented

by

Hiruni Kamali Pallage

It was defended on

July 12, 2019

and approved by

Jeffrey Paul Wheeler, Ph.D., Lecturer II, Mathematics, University of Pittsburgh

Michael A. Trick, Harry B. and James H. Higgins Professor of Operations Research; Dean, Carnegie Mellon University Qatar

G. Bard Ermentrout, Distinguished University Professor, Mathematics, University of Pittsburgh

Jason DeBlois, Associate Professor, Mathematics, University of Pittsburgh

Michael Schneier, Post-Doctoral Associate, Mathematics, University of Pittsburgh

Thesis Advisor: Jeffrey Paul Wheeler, Ph.D., Lecturer II, Mathematics, University of Pittsburgh

Optimization via Benders' Decomposition

Hiruni Kamali Pallage, M.S.

University of Pittsburgh, 2019

In a period when optimization has entered almost every facet of our lives, this thesis is designed to establish an understanding about the rather contemporary optimization technique: Benders' Decomposition. It can be roughly stated as a method that handles problems with complicating variables, which when temporarily fixed, yield a problem much easier to solve. We examine the classical Benders' Decomposition algorithm in greater depth followed by a mathematical defense to verify the correctness, state how the convergence of the algorithm depends on the formulation of the problem, identify its correlation to other well-known decomposition methods for Linear Programming problems, and discuss some real-world examples. We introduce present extensions of the method that allow its application to a wider range of problems. We also present a classification of acceleration strategies which is centered round the key sections of the algorithm. We conclude by illustrating the shortcomings, trends, and potential research directions.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Preface

It is a pleasure to pay tribute to one and all who provided me unflinching encouragement and support in various ways to complete my research work.

First and foremost, I wholeheartedly appreciate my wonderful advisor, Dr. Jeffrey Wheeler, for his patience, motivation, enthusiasm, and immense knowledge. I have spent many hours in productive and fruitful conversation on research and life with my outstanding advisor. Thank you for going above and beyond the role of an advisor and being that person who understood the challenges that graduate school brought into everyday life.

Besides my advisor, I would like to thank Professor Michael A. Trick (Carnegie Mellon University, Qatar) for his intellectual contributions to the work. It was a wonderful experience to work with such an intelligent expert in Operational Research field.

I would also like to thank the rest of my thesis committee: Professor G. Bard Ermentrout (University of Pittsburgh), Professor Jason DeBlois (University of Pittsburgh), and Dr. Michael Schneier (University of Pittsburgh) for their encouragement, valuable time and insightful comments.

Moreover, I owe a great deal to the Department of Mathematics and the University of Pittsburgh for providing me such a great place to broaden my mathematical sphere of knowledge, and to meet so many great and nice friends of my life. I also want to thank all my friends for their help and encouragement in my graduate study and in my life.

Thank you, Mom and Dad for the emotional support, intellectual stimulation and many hours of identity-forming conversation, inspiring me to pursue unconventional dreams in which I truly believe. You are the most supportive parents one could hope for.

Last but not the least, I record my unstinted thanks to my loving and tolerant husband Achna for his faithful support rendered throughout, comforting me in the hardest times.

1.0 Introduction

The objective of any company is to maximize its profits and efficiently use its resources. In attempting to complete a project without a schedule, for example, one may accomplish the goals but not without wasted time and money. As a result, optimization and scheduling has become a tremendous problem for the management of companies and recent years have seen an increased demand for the application of mathematics to develop the perfect schedule to meet the goals.

"Optimization via Benders' Decomposition" (BD) addresses the perennial problem of optimal utilization of finite resources in the accomplishment of an assortment of tasks or objectives. The thesis refers to applications which provide ways to uncover the core of the above real-world challenges, present them in mathematical terms, and devise mathematical solutions for them with the use of BD algorithm.

The main focus of the BD algorithm is to deal with problems where certain variables are temporally fixed yielding a problem considerably easier to solve. The BD method has now developed to be one of the most extensively used exact algorithms since it utilizes the structure of the problem to decentralize the total computational weight. Successful applications are found in many diverse fields, including planning and scheduling.

The remainder of the thesis is organized as follows. After discussing the history related to the BD algorithm, Chapter 3 presents the theory behind the classical BD algorithm followed by simple examples where Chapter 4 offers a mathematical justification to prove the correctness of the algorithm. Then, Chapter 5 focuses on the extensions and generalizations of BD algorithm leading to Chapter 6; which will be about some of the applications of the algorithm where applications are selected from several fields to show the reach of the BD algorithm. Finally, Chapter 7 provides concluding remarks and describes other promising research directions of the algorithm. Further, a primer on Linear Programming and Integer Programming is offered in the Appendix.

We conclude the introduction by illustrating the evolution of the BD algorithm in various fields, giving an informative table provided in *The Benders' Decomposition Algorithm: A*

Literature Review, [27].

	Reference	Application		Reference	Application
1	Behnamian (2014)	Production planning	17	Jiang et al. (2009)	Distribution planning
2	Adulyasak et al. (2015)	Production routing	18	Kim et al. (2015)	Inventory control
3	Boland at al. (2015)	Facility location	19	Laporte et al. (1994)	Traveling salesman
4	Boschetti & Maniezzo (2009)	Project scheduling	20	Luong (2015)	Healthcare planning
5	Botton al. (2013)	Survivable network design	21	Maravelias & Grossmann (2004)	Chemical process design
6	Cai et al. (2001)	Water resource management	22	Moreno-Centeno and Karp (2013)	Implicit hitting sets
7	Canto (2008)	Maintenance scheduling	23	Oliveira et al. (2014)	Investment planning
8	Codato & Fischetti (2006)	Map labeling	24	Osman and Baki (2014)	Transfer line balancing
9	Cordeau et al. (2006)	Logistics network design	25	Pérez- Galarce et al. (2014)	Spanning tree
10	Cordeau al. (2001a)	Loocomotive assignment	26	Pishvaee et al. (2014)	Supply chain network design
11	Cordeau et al. (2001b)	Airline scheduling	27	Rubiales et al. (2013)	Hydrothermal coordination
12	Corréa et al. (2007)	Vehicle routing	28	Saharidis et al. (2011)	Refinery system network planning
13	Côté et al. (2014)	Strip packing	29	Sen et al. (2015)	Segment allocation
14	Fortz and Poss (2009)	Network design	30	Bloom (1983)	13 Capacity expansion
15	Gelareh et al. (2015)	Transportation	31	Wang et al. (2016)	Optimal power flow
16	Jenabi et. Al. (2015)	Power management			

Table 1: Some applications of Benders' Decomposition algorithm from [27]

2.0 History

Jacobus Franciscus (Jacques) Benders (1924 - 2017) was the first professor in the Netherlands in the field of Operations Research and is known for his role in mathematical programming [37]. He obtained his PhD in 1960 with the thesis titled "Partitioning in Mathematical Programming" from Utrecht University. Starting his career as a statistician for the Rubber Foundation in late 1940s, he then moved to Shell Laboratory in Amsterdam in 1955. He researched mathematical programming problems regarding the logistics of the oil refinery and developed the method name after him. Benders was designated Professor of Operations Research at the Eindhoven University of Technology in 1963 and retired in 1989. Furthermore, in 2009, he was bestowed the EURO Gold Medal, the highest distinction in the area of Operations Research in Europe. Although the algorithm was first introduced to solve the Mixed-Integer Linear Programming (MILP) problems, later developments were made to apply the algorithm to a broader range of problems and to increase its efficiency on certain optimization classes.

	Reference	Model		Reference	Model
1	Adulyasak et al. (2015)	Multi-period stochastic problem	11	Jenabi et al. (2015)	Piecewise linear mixed-integer problem
2	Behnamian (2014)	Multi-objective MILP	12	Kim et al. (2015)	Multi-stage stochastic program
3	Cai et al. (2001)	Multi-objective nonconvex nonlinear problem	13	Laporte et al. (1994)	Probabilistic integer formulation
4	Cordeau et al. (2001b)	Pure $0-1$ formulation	14	Li (2013)	Large-scale nonconvex MINLP
5	Corréa et al. (2007)	Binary problem with logical expressions	15	Moreno- Centeno & Karp (2013)	Problem with constraints unknown in advance
6	Gabrel et al. (1999)	Step increasing cost	16	Bloom (1983)	Nonlinear multi-period problem with reliability constraint
7	Côté et al (2014)	MILP with logical constraints	17	Osman and Baki (2014)	Nonlinear integer formulation
8	de Camargo et al (2011)	Mixed-integer nonlinear program (MINLP)	18	Pérez-Galarce et al. (2014)	Minmax regret problem
9	Emami et al. (2016)	Robust optimization problem	19	Pishvaee et al. (2014)	Multi-objective possibilistic programming model
10	Fontaine & Minner (2014)	Bilevel problem with bilinear constraints	20	Raidl et al. (2014)	Integer, bilevel, capacitated problem

Table 2: Some optimization problems solved via Benders' Decomposition from [27]

3.0 Definitions and Examples

We begin by highlighting the significance of Benders' Decomposition (BD) algorithm as an approach for solving certain large-scale optimization problems. When it comes to constructing and solving optimization problems a major concern is that the amount of memory and the computational effort required to solve such problems will grow substantially with the number of variables and constraints. The conventional method of making all decisions simultaneously by solving a massive optimization problem quickly turns out to be intractable with the increase in the number of variables and constraints. To alleviate this difficulty, multistage optimization algorithms such as BD have been developed as an alternative solution methodology. Unlike the traditional methods, these algorithms split the decision-making process into several phases.

In reality, the first step of the BD algorithm is to fix certain variables in the original problem, thus making the resulting subproblem easy to solve. Throughout the thesis we refer to those variables, which make the problem significantly easier to solve when fixed, as **complicating variables.** So it is clear that the core of BD algorithm is to identify the right decomposition for the given model (that is, the right partitioning of the variables). This decision usually demands specific knowledge of the problem at hand including known methods to solve similar problems quickly. It is hence not possible for us within the scope of this thesis to give a complete outline of BD algorithm that works for all problems. However, in Section 3 and 6 we provide some concrete examples where we choose certain variables to be fixed and employ BD algorithm successfully to solve the problem.

In BD, the problem is divided into a master problem (MP) and a subproblem (SP), which are then solved iteratively. The MP which considers a subset of the variables, is solved first. Next, we temporarily fix the variables' values of the MP and solve the SP for the remaining variables. Finally, depending on the solution of the SP, one or more cuts are derived and added to the MP, thus effectively averting the MP from returning to similar areas of the search space. Note that in the classical Benders' Decomposition the SP is a Linear Programming problem, where cuts are generated based on the outcome of its dual

problem (DSP).

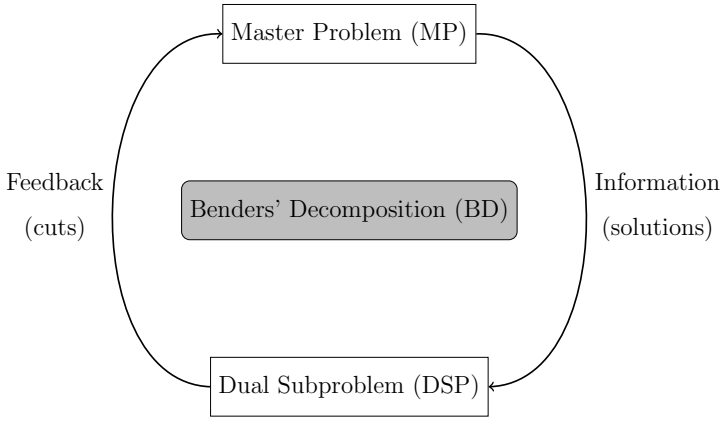

Figure 1: Schematic representation of Benders' Decomposition algorithm from [27]

Regarding cuts, we note that current Integer Programming techniques walk us around corner points of the feasible region in search of corner points that lead to feasible solutions. If it does not arrive at a feasible solution at a specific corner point, the main approach is to cut away that part of the feasible region by introducing a new constraint which throws out that corner point. At the same time, it ensures not to throw off the corner point/s where the optimal value occurs. So, in general **cuts** are additional constraints that cut the feasible region to reduce the solution search space to simply contain feasible solutions.

3.1 Primal and Dual Linear Programs

As mentioned in the previous section, we need to understand concepts of duality in linear programs before we dive into the basic model and solution steps of BD algorithm. We consider a Linear Programming (LP) problem that can be expressed in matrix notation as follows:

$$\begin{aligned} &Maximize && P = \boldsymbol{c}^T \boldsymbol{x} \\ &such\ that && A\boldsymbol{x} \leq \boldsymbol{b} \\ & && \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

where $\boldsymbol{c}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. The linear function $\boldsymbol{c}^T\boldsymbol{x}$ is the **objective function** and the linear inequalities are the **constraints** which generate a **feasible region** to minimize the objective function. The solutions of the primal problem in the feasible region can be written as $\{\boldsymbol{x} \in \mathbb{R}^n | A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$.

We refer to the above original LP as the **primal problem** and any primal problem can be expressed in another LP form which is called the **dual problem**. The corresponding dual problem to the above primal problem can be expressed as follows:

$$\begin{aligned} &\textit{Minimize} && C = \boldsymbol{b}^T\boldsymbol{y} \\ &\textit{such that} && A^T\boldsymbol{y} \geq \boldsymbol{c} \\ & && \boldsymbol{y} \geq \boldsymbol{0}. \end{aligned}$$

These two forms are linked by the following theorem:

Theorem 1 (The Fundamental Principle of Duality from [36])**.**
A minimization problem has a solution if and only if the corresponding dual maximization problem has a solution.

In more specific terms:

Theorem 2 (from [36])**.**
If $\boldsymbol{x}$ satisfies the constraints of a Linear Programming problem in primal form and if $\boldsymbol{y}$ satisfies the constraints of the corresponding dual, then $\boldsymbol{c}^T\boldsymbol{x} \leq \boldsymbol{b}^T\boldsymbol{y}$.

Proof. From the primal form, we have that $\boldsymbol{x} \geq \boldsymbol{0}$ and $A\boldsymbol{x} \leq \boldsymbol{b}$. Similarly, by the dual $\boldsymbol{y} \geq \boldsymbol{0}$ and $A^T\boldsymbol{y} \geq \boldsymbol{c}$ are satisfied. Hence

$$\boldsymbol{c}^T\boldsymbol{x} \leq (A^T\boldsymbol{y})^T\boldsymbol{x} = \boldsymbol{y}^T A\boldsymbol{x} \leq \boldsymbol{y}^T\boldsymbol{b} = \boldsymbol{b}^T\boldsymbol{y}$$

where the last equality holds since both expressions are dot products. □

We know that every LP is either feasible or not feasible and that feasible problems either are unbounded or have a solution. We say that an LP has a **feasible** solution if there exists a set of values for the decision variables that satisfies all the constraints. When it is impossible to find a feasible solution, that is, when we cannot obtain a solution that meets each constraint, LP is said to be **infeasible**. An **unbounded** solution to an LP with the objective of maximizing (minimizing) is when it is possible to construct the solution to be infinitely large (small) while none of the constraints are violated. From the above theorem we can conclude that if primal is unbounded, then dual is infeasible. Likewise, if dual is unbounded, then primal is infeasible. The Duality Theorem allows us to understand more possible relationships among solutions of primal and dual.

Theorem 3 (Duality Theorem for Linear Programs)**.**
For a primal-dual pair of Linear Programming problems, one of these four cases occurs:

1. *Both are infeasible.*
2. *Primal is unbounded and dual is infeasible.*
3. *Dual is unbounded and primal is infeasible.*
4. *Both are feasible and there exist optimal solutions* $\boldsymbol{x}$, $\boldsymbol{y}$ *to primal and dual such that* $\boldsymbol{b}^T\boldsymbol{y} = \boldsymbol{c}^T\boldsymbol{x}$.

Another way to think about this relationship is to create a table of possibilities. We take each of the three rows to denote one of three possibilities of the primal solution. The columns denote the same characteristics of the dual solution. An alternative method to justify the table below is to examine the Simplex Method which solves primal and dual simultaneously.

Primal \ Dual	**Unbounded**	**Has a solution**	**Not feasible**
Unbounded	impossible	impossible	possible
Has a solution	impossible	same value	impossible
Not feasible	possible	impossible	possible

Table 3: The relationship between solutions of primal and dual problems

Any discussion on the duality of LP problems will be incomplete without understanding how to convert a given primal LP to its corresponding dual. Assuming that the primal has M constraints and N variables, now we summarize the relationship between primal and dual:

Primal (or dual)		**Dual (or Primal)**	
Objective	Maximize P	Minimize C	Objective
Variable (N)	≥ 0	$\geq$	Constraints (N)
	≤ 0	$\leq$	
	unbounded	$=$	
Constraints (M)	$\leq$	≥ 0	Variable (M)
	$\geq$	≤ 0	
	$=$	unbounded	
Right side of constraints		Coefficients of variables in objective functions	
Coefficients of variables in objective functions		Right side of constraints	

Table 4: The relationship between primal and dual problems from [30]

Now we consider an easy example to find the dual of the given primal problem.

Primal Problem

$$
\begin{aligned}
\textit{Maximize} \quad & P = 15x_1 + 14x_2 + 16x_3 \\
\textit{such that} \quad & x_1 + 2x_2 \geq 21 \\
& x_1 + x_3 \leq 31 \\
& -5x_1 + 8x_2 + x_3 \leq -51 \\
& x_1 - x_2 + x_3 = 11 \\
& x_1 \geq 0,\ x_2 \leq 0,\ x_3 \ \textit{unbounded}
\end{aligned}
$$

Dual Problem

$$
\begin{aligned}
\textit{Minimize} \quad & C = 21y_1 + 31y_2 - 51y_3 + 11y_4 \\
\textit{such that} \quad & y_1 + y_2 - 5y_3 + y_4 \geq 15 \\
& 2y_1 + 8y_3 - y_4 \leq 14 \\
& y_2 + y_3 + y_4 = 16 \\
& y_1 \leq 0,\ y_2,\ y_3 \geq 0,\ y_4 \ \textit{unbounded}
\end{aligned}
$$

The number of inequalities (variables) in the primal becomes the number of variables (inequalities) in the dual. Thus, the dual may differ in the dimension from the primal and it may be desirable to solve an LP with fewer constraints.

3.2 Basic Model of Benders' Decomposition

We now carry on to describe the basic model of BD algorithm which is critical for understanding all the proceeding discussions in the thesis. Here we consider a Mixed Integer Linear Program (MILP), that is, the original problem (P1) of the following general form:

$$\begin{aligned} Minimize \quad & z = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{y} \\ such\ that \quad & A\boldsymbol{y} \geq \boldsymbol{b} \\ & B\boldsymbol{x} + C\boldsymbol{y} \geq \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{y} \in \mathbb{Y} \subseteq \mathbb{Z}^n \end{aligned}$$

with complicating variables $\boldsymbol{y} \in \mathbb{Z}^n$ which must fit the constraint $A\boldsymbol{y} \geq \boldsymbol{b}$, where $A \in \mathbb{R}^{m\times n}$ is a known matrix and $\boldsymbol{b} \in \mathbb{R}^m$ is a given vector. The variables $\boldsymbol{x} \in \mathbb{R}^s$, as well as $\boldsymbol{y}$ variables, should satisfy the connecting constraint $B\boldsymbol{x} + C\boldsymbol{y} \geq \boldsymbol{d}$, where $B \in \mathbb{R}^{r\times s}$, $C \in \mathbb{R}^{r\times n}$, and $\boldsymbol{d} \in \mathbb{R}^r$. The objective function is to minimize the total cost together with the cost vectors $\boldsymbol{f} \in \mathbb{Z}^n$ and $\boldsymbol{c} \in \mathbb{R}^s$. Note that, $\boldsymbol{x}$ having real components and $\boldsymbol{y}$ having integer components, make the above a MILP.

We suppose that the $\boldsymbol{y}$ variables are complicating variables, so if $\boldsymbol{y}$ variables are fixed, the problem is linear in $\boldsymbol{x}$ and becomes significantly easier to solve. Hence, P1 can be stated as: $\min\limits_{\boldsymbol{y}\in\boldsymbol{R}} \left\{ \boldsymbol{f}^T\boldsymbol{y} \,|\, A\boldsymbol{y} \geq \boldsymbol{b} + \min\{B\boldsymbol{x} \geq \boldsymbol{d} - C\boldsymbol{y}, \boldsymbol{x} \geq \boldsymbol{0}\} \right\}$ with $\boldsymbol{R} = \{\boldsymbol{y} \,|\quad$ there exist $\boldsymbol{x} \geq \boldsymbol{0}$ such that $B\boldsymbol{x} \geq \boldsymbol{d} - C\boldsymbol{y}, \boldsymbol{x} \geq \boldsymbol{0}$, $\boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}$, $\boldsymbol{y} \in \mathbb{Y}\}$. So P1 can be divided into an MP that contains the $\boldsymbol{y}$ variables and an SP that contains the $\boldsymbol{x}$ variables.

Master Problem (MP)

$$\begin{aligned} Minimize \quad & z_{lower} \\ such\ that \quad & z_{lower} \geq \boldsymbol{f}^T\boldsymbol{y} \\ & \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b} \\ & \boldsymbol{y} \in \mathbb{Y} \end{aligned}$$

Primal Subproblem (SP)

$$\begin{aligned} Minimize \quad & \boldsymbol{c}^T\boldsymbol{x} \\ such\ \ that \quad & B\boldsymbol{x} \geq \boldsymbol{d} - C\hat{\boldsymbol{y}} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

where $\hat{\boldsymbol{y}}$ is the solution of the MP.

Also note that taking $\boldsymbol{u}$ to be the dual variable associated with $B\boldsymbol{x} \geq \boldsymbol{d} - C\hat{\boldsymbol{y}}$, the dual subproblem (DSP) of SP can be written as:

$$\begin{aligned} &Maximize && (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T \boldsymbol{u} \\ &such\ that && B^T \boldsymbol{u} \leq \boldsymbol{c} \\ & && \boldsymbol{u} \geq \boldsymbol{0}. \end{aligned}$$

3.3 Solution Steps for the Algorithm

Here we present a basic idea behind the algorithm before formally stating it. After developing the initial MP and SP, the algorithm starts with the MP and alternates between MP and SP till an optimal solution is derived. The objective function of the MP generates a fitting lower bound on the optimal cost; while combining the $\hat{\boldsymbol{y}}$ solution with the objective value of the SP (which is equivalent to fixing $\hat{\boldsymbol{y}}$ in the original problem) provides a valid upper bound on the optimal cost. Then, the optimality gap is computed in each iteration to confirm the convergence of the algorithm. The classical BD approach uses the Branch and Bound algorithm to solve the MP and the Simplex Method is employed to tackle the SP. We will now move our attention to describing BD algorithm in greater depth using mathematical terms.

Algorithm 1 Classical Benders' Decomposition Algorithm from [30]

1. Initialize $\hat{\boldsymbol{y}}$ and set ϵ as necessary.
2. Solve MP1 and find an initial lower bound solution $\hat{z}_{lower}$ at $\hat{\boldsymbol{y}}$.
   - If MP1 is infeasible so is the original problem P1.
   - If MP1 is unbounded, set $\hat{z}_{lower} = \infty$ in MP1 for an arbitrary value of $\hat{\boldsymbol{y}}$ in $\mathbb{Y}$, and proceed to step 3.
3. Solve SP or DSP to get an upper bound solution to the original problem P1.
   Solving DSP: $\hat{z}_{upper} = \boldsymbol{f^T}\hat{\boldsymbol{y}} + (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}}$ is the upper bound solution for optimal dual solution $\hat{\boldsymbol{u}}$. Solving SP: $\hat{z}_{upper} = \boldsymbol{c}^T\hat{\boldsymbol{x}} + \boldsymbol{f}^T\hat{\boldsymbol{y}}$ is the upper bound solution for the original problem P1 for $\hat{\boldsymbol{x}}$.

- If $|\hat{z}_{lower} - \hat{z}_{upper}| \leq \epsilon$ for P1, the process is terminated. Otherwise, add a new constraint (feasibility cut) $z_{lower} \geq \boldsymbol{f^T y} + (\boldsymbol{d} - C\boldsymbol{y})^T \hat{\boldsymbol{u}}$ to MP1 to form MP2 and proceed to step 4.
- If DSP is unbounded (that is, SP is infeasible), then introduce a new cut (infeasibility cut) $(\boldsymbol{d} - C\boldsymbol{y})^T \hat{\boldsymbol{u}} \leq \boldsymbol{0}$ to MP1 to form MP2. Now we use a new SP, feasibility check subproblem below to calculate $\boldsymbol{u}$ in DSP to form the infeasibility cut and then proceed to step 4.

$$\begin{aligned} Minimize \quad & \mathbf{1}^T \boldsymbol{s} \\ such\ that \quad & B\boldsymbol{x} + \mathbf{1}\boldsymbol{s} \geq \boldsymbol{d} - C\hat{\boldsymbol{y}} \qquad \rightarrow \hat{\boldsymbol{u}} \\ & \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{s} \geq \boldsymbol{0} \end{aligned}$$

- If DSP is infeasible (that is SP is unbounded or infeasible), the original problem P1 will have either no feasible solution or an unbounded solution. So the process terminates.

4. Solve the updated master problem MP2 to find a new lower bound solution $\hat{z}_{lower}$ for the original problem P1, with respect to $\hat{\boldsymbol{y}}$. In the following MP2 formulation, either the feasibility cut (second constraint) or the infeasibility cut (third constraint) can be used as discussed in Step 3.

$$\begin{aligned} Minimize \quad & z_{lower} \\ such\ that \quad & A\boldsymbol{y} \geq \boldsymbol{b} \\ & z_{lower} \geq \boldsymbol{f^T y} + (\boldsymbol{d} - C\boldsymbol{y})^T \hat{\boldsymbol{u}} \\ & (\boldsymbol{d} - C\boldsymbol{y})^T \hat{\boldsymbol{u}} \leq \boldsymbol{0} \\ & \boldsymbol{y} \in \mathbb{Y} \end{aligned}$$

Then return to step 3 to solve the subproblem SP or the dual of the subproblem DSP again. If MP2 is unbounded, specify $\hat{z}_{lower} = \infty$ and return to step 3. If MP2 is infeasible, so is the original problem P1. So, the process terminates.

Note that it is common in literature to refer the *feasibility cut* as the *optimality cut* and the *infeasibility cut* as the *feasibility cut*.

Having used the previous pages to explain how classical BD works, we will now provide a flowchart that summarizes the above discussion.

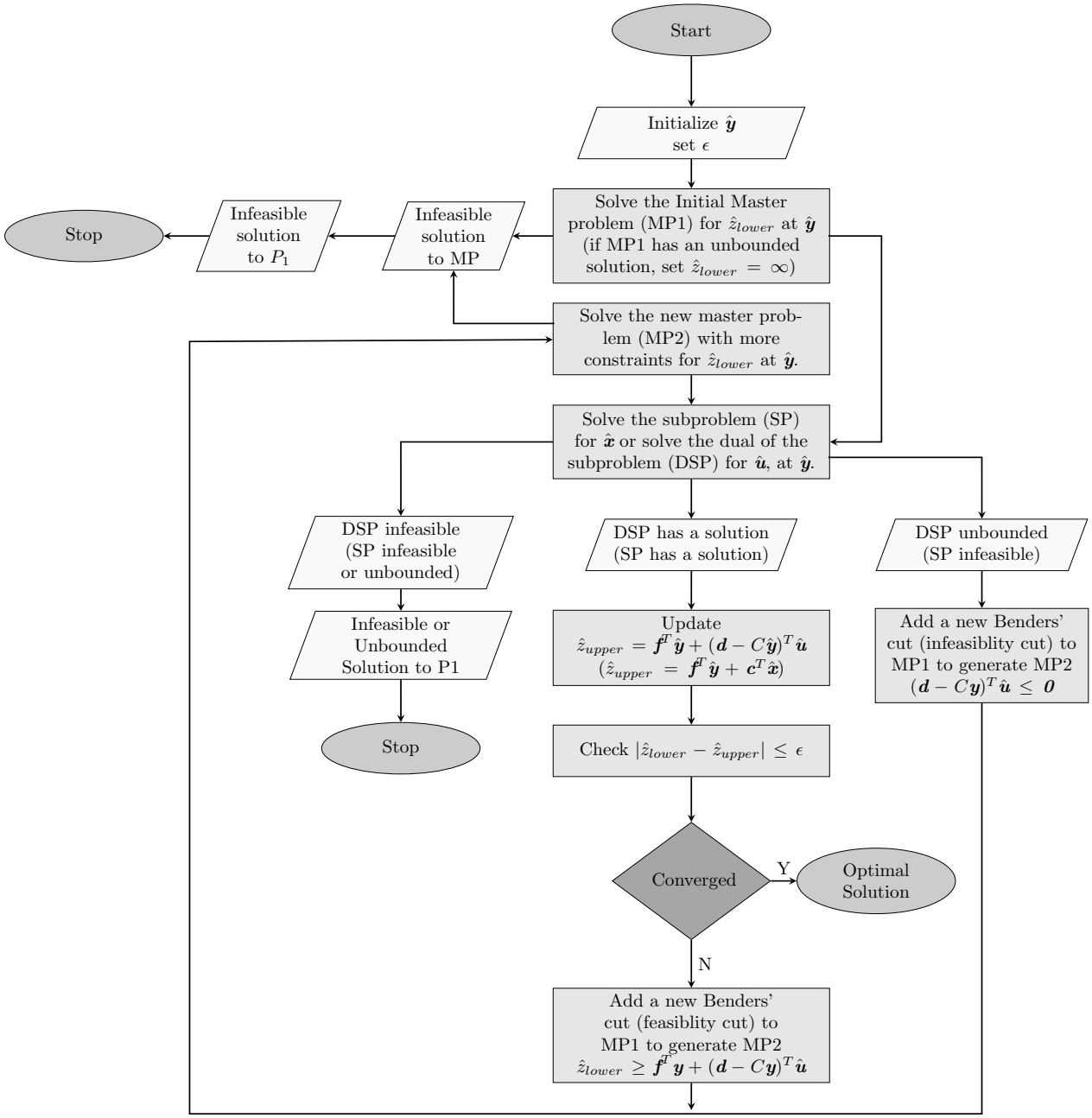

Figure 2: Flowchart of classical Benders' Decomposition algorithm from [30]

We revisit the general idea behind the algorithm to explain some key facts that the careful reader may have already observed.

Given the original problem (P1) our goal is to find values of variables $\boldsymbol{x}$, $\boldsymbol{y}$, and the objective function z. We solve the MP to get a lower bound solution to z (that is z_{lower}) and solve the primal/dual SP to get an upper bound solution to z (that is z_{upper}). We continue the process until the values of z are within the tolerance. In other words, MP and the SP work together to generate a solution to P1.

z_{lower} in the MP is in fact a relaxation of z in P1. Since we do not have value of $\boldsymbol{x}$ variable at the beginning of the process we relax the $\boldsymbol{x}$ variable in z and employ BD algorithm which finds appropriate $\boldsymbol{x}$ values and represent them as ideally a small number of constraints which we then introduce to the MP. That is, even though the variable $\boldsymbol{x}$ does not appear explicitly in the MP, it is represented by the various constraints (feasibility and infeasibility cuts) that are generated by the SP. These cuts are implicitly representing effect of $\boldsymbol{x}$ in $P1$. So once we generate all the constraints related to each feasible $\boldsymbol{x}$ in the primal SP then the z_{lower} in the MP will equal to the z in P1. Thus, MP is implicitly the same problem as P1.

Here we explain what motivates the choice of feasibility and infeasibility cuts to be defined as we stated in step 3 of Algorithm 1. Suppose we figure out a feasible $\hat{\boldsymbol{u}}$ for the DSP for which the DSP is unbounded. It means that this $\hat{\boldsymbol{u}}$ is a direction of unboundedness for which the objective function value of the DSP $(\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\boldsymbol{u} > \boldsymbol{0}$. So as long as we choose any $\hat{\boldsymbol{y}}$ for which $(\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} > \boldsymbol{0}$ then the DSP will be made unbounded. Thus in order to avoid this unbounded ray we got to choose a $\hat{\boldsymbol{y}}$ that does not allow $(\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} > \boldsymbol{0}$. In other words, we must introduce $(\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}} \leq \boldsymbol{0}$ (infeasibility cut) to the MP to restrict the movement in this direction. Feasibility cut works really the same way because at this point we have a feasible $\hat{\boldsymbol{u}}$ which created a finite objective function value for the DSP (as well as a feasible $\hat{\boldsymbol{x}}$ which generated a finite objective function value for the SP). We update the upper bound solution to z using $\hat{z}_{upper} = \boldsymbol{f^T}\hat{\boldsymbol{y}} + (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}}$ or $\hat{z}_{upper} = \boldsymbol{c}^T\hat{\boldsymbol{x}} + \boldsymbol{f}^T\hat{\boldsymbol{y}}$. Now if we are not within the tolerance, we introduce $z_{lower} \geq \boldsymbol{f^T}\boldsymbol{y} + (\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}}$ (feasibility cut) which can be similarly expressed as $z_{lower} \geq \boldsymbol{f^T}\boldsymbol{y} + \boldsymbol{c}^T\hat{\boldsymbol{x}}$ to the MP which says that if we need a better DSP/SP value we better avoid this solution.

In this chapter we give little attention to developing a complete discussion on LP even though we use concepts related to LP extensively. However, we provide more details in the Appendix which includes Linear Programming as well as Integer Programming.

To enhance the understanding of how the BD algorithm works, we now illustrate a simple numerical example.

Example 1.

Solve the following MILP, P1 [30] using the BD algorithm.

$$\begin{aligned} Minimize \quad & x + y \\ such\ that \quad & 2x + y \geq 3 \\ & x \geq 0,\ y \in \{-5, -4, -3, ..., 3, 4\} \end{aligned}$$

Comparing with the basic model for BD: $\boldsymbol{c}^T = [1],\ \boldsymbol{f}^T = [1],\ B = [2],\ C = [1],\ \boldsymbol{d} = [3]$.

Iteration 1:

Master Problem (MP1)	*Primal Subproblem (SP)*	*Dual Subproblem (DSP)*
$Minimize \quad z_{lower}$	$Minimize \quad x$	$Maximize \quad (3 - \hat{y})u$
such that	*such that*	*such that*
$z_{lower} \geq y$	$2x \geq 3 - \hat{y}$	$2u \leq 1$
$y \in \{-5, -4, ..., 3, 4\}$	$x \geq 0$	$u \geq 0$

The lower bound optimal solution to the original problem is: $\hat{z}_{lower} = -5$ *when* $\hat{y} = -5$. *Now we solve the DSP when* $\hat{y} = -5$.

$$\begin{aligned} Maximize \quad & 8u \\ such\ that \quad & 2u \leq 1 \\ & u \geq 0 \end{aligned}$$

The optimal solution to DSP is 4 when $u = 1/2$. *Thus,* $\hat{z}_{upper} = \boldsymbol{f^T}\hat{\boldsymbol{y}} + (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} = \hat{y} + 4 = -5 + 4 = -1$. *Since* $\hat{z}_{upper} = -1 > \hat{z}_{lower} = -5$ *we continue to the next iteration. So, we need to introduce a new cut (feasibility cut),* $z_{lower} \geq \boldsymbol{f^T}\boldsymbol{y} + (\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}}$ *to MP1 to form MP2. The new Benders' cut can be written as:*

$$\begin{aligned} & z_{lower} \geq \boldsymbol{f^T}\boldsymbol{y} + (\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}} \\ & z_{lower} \geq y + (3 - y)\frac{1}{2} \\ & z_{lower} \geq \frac{3}{2} + \frac{1}{2}y. \end{aligned}$$

Iteration 2:

The new master problem MP2:

$Minimize \quad z_{lower}$

such that $\quad z_{lower} \geq y$

$z_{lower} \geq \frac{3}{2} + \frac{1}{2}y$

$y \in \{-5, -4, ..., 3, 4\}.$

The new lower bound optimal solution to the original problem is: $\hat{z}_{lower} = -1$ *when* $\hat{y} = -5$. *Then we solve the DSP when* $\hat{y} = -5$.

$Maximize \quad 8u$

such that $\quad 2u \leq 1$

$u \geq 0$

The optimal solution to DSP is again 4 when $u = 1/2$. *Thus,* $\hat{z}_{upper} = \boldsymbol{f^T}\hat{\boldsymbol{y}} + (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}}$ $= \hat{y} + 4 = -5 + 4 = -1$. *Now since* $\hat{z}_{upper} = -1 = \hat{z}_{lower}$ *the process has converged. The solution for the SP (using the optimal solution to DSP) is* $x = 4$. *Thus,* $x = 4$ *and* $y = -5$ *minimizes the the original problem P1 and* $P1_{min} = \{x + y\}_{min} = -1$.

3.4 Alternative Form of Benders Cuts

Here we switch focus to a slightly different form of representing the Benders' cuts [30]. Recall that the Benders' cuts that were introduced had the form:

$$z_{lower} \geq \boldsymbol{f^T}\boldsymbol{y} + (\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}} \quad feasibility\ cut$$
$$(\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}} \leq \boldsymbol{0} \quad infeasibility\ cut$$

which can also be represented as:

$$z_{lower} \geq \boldsymbol{f^T}\boldsymbol{y} + w(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T\hat{\boldsymbol{u}} \quad feasibility\ cut$$
$$v(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T\hat{\boldsymbol{u}} \leq \boldsymbol{0} \quad infeasibility\ cut$$

where $w(\hat{\boldsymbol{y}})$ is the optimal solution of SP and $v(\hat{\boldsymbol{y}})$ is the optimal solution of the feasibility check subproblem. Here $z_{lower} \geq \boldsymbol{f^T}\boldsymbol{y} + w(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T\hat{\boldsymbol{u}}$ shows that the objective value

of the original problem is decreased by updating $\boldsymbol{y}$ from $\hat{\boldsymbol{y}}$ to a new value. $\hat{\boldsymbol{u}} \in \mathbb{R}^s$ indicates the incremental change in the optimal objective. $v(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T \hat{\boldsymbol{u}} \leq \boldsymbol{0}$ shows that $\hat{\boldsymbol{y}}$ is updated to a new value to eliminate constraint violations in SP based on $\hat{\boldsymbol{y}}$ given in the master problem. $\hat{\boldsymbol{u}} \in \mathbb{R}^t$ indicates the incremental change in the total violation.

Example 2.

Solve the following MILP, P1 [30] using the alternative form of BD algorithm stated above.

$$\begin{aligned} \textit{Minimize} \quad & x_1 + 3x_2 + y_1 + 4y_2 \\ \textit{such that} \quad & -2x_1 - x_2 + y_1 - 2y_2 \geq 1 \\ & 2x_1 + 2x_2 - y_1 + 3y_2 \geq 1 \\ & x_1,\, x_2 \geq 0,\, y_1,\, y_2 \geq 0 \end{aligned}$$

Comparing with the basic model for BD:

$\boldsymbol{c}^T = \begin{bmatrix} 1 & 3 \end{bmatrix}$, $\boldsymbol{f}^T = \begin{bmatrix} 1 & 4 \end{bmatrix}$, $B = \begin{bmatrix} -2 & -1 \\ 2 & 2 \end{bmatrix}$, $C = \begin{bmatrix} 1 & -2 \\ -1 & 3 \end{bmatrix}$, $\boldsymbol{d} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Iteration 1:

Master Problem (MP1)

$$\begin{aligned} \textit{Minimize} \quad & z_{lower} \\ \textit{such that} \quad & \\ & z_{lower} \geq y_1 + 4y_2 \\ & y_1 \geq 0,\, y_2 \geq 0 \end{aligned}$$

Primal Subproblem (SP)

$$\begin{aligned} \textit{Minimize} \quad & x_1 + 3x_2 \\ \textit{such that} \quad & \\ & -2x_1 - x_2 \geq 1 - \hat{y}_1 + 2\hat{y}_2 \\ & 2x_1 + 2x_2 \geq 1 + \hat{y}_1 - 3\hat{y}_2 \\ & x_1,\, x_2 \geq 0 \end{aligned}$$

The lower bound optimal solution to the original problem is: $\hat{z}_{lower} = 0$ when $\hat{y}_1 = \hat{y}_2 = 0$. Now we solve the SP when $\hat{y}_1 = \hat{y}_2 = 0$.

$$\begin{aligned} \textit{Minimize} \quad & x_1 + 3x_2 \\ \textit{such that} \quad & -2x_1 - x_2 \geq 1 \\ & 2x_1 + 2x_2 \geq 1 \\ & x_1,\, x_2 \geq 0 \end{aligned}$$

Applying Simplex Method we can see that the solution to the SP is not feasible. Thus we need to introduce a new cut (infeasibility cut), $v(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T \hat{\boldsymbol{u}} \leq \mathbf{0}$ *to MP1 to form MP2. The feasibility check subproblem below is used to calculate* $\hat{\boldsymbol{u}}$ *in DSP to form the above infeasibility cut. The feasibility check subproblem can be written as:*

$$\begin{aligned} &\textit{Minimize} \quad \mathbf{1}^T \boldsymbol{s} \\ &\textit{such that} \quad B\boldsymbol{x} + \mathbf{1}\boldsymbol{s} \geq \boldsymbol{d} - C\hat{\boldsymbol{y}} \\ &\qquad\qquad \boldsymbol{x} \geq \mathbf{0},\ \boldsymbol{s} \geq \mathbf{0} \end{aligned} \quad \Rightarrow \quad \begin{aligned} &\textit{Minimize} \quad s_1 + s_2 \\ &\textit{such that} \quad -2x_1 - x_2 + s_1 \geq 1 - \hat{y}_1 + 2\hat{y}_2 \\ &\qquad\qquad 2x_1 + 2x_2 + s_2 \geq 1 + \hat{y}_1 - 3\hat{y}_2 \\ &\qquad\qquad x_1,\ x_2 \geq 0,\ s_1,\ s_2 \geq 0. \end{aligned}$$

Applying the Simplex Method to the above feasibility check subproblem we can see that the optimal solution is 1.5 and values of the associated dual variables are $\hat{u}_1 = 1.0$, $\hat{u}_2 = 0.5$. *Now for* $\hat{y}_1 = \hat{y}_2 = 0$ *the Benders' cut can be written as:*

$$\begin{aligned} &v(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T \hat{\boldsymbol{u}} \leq \mathbf{0} \\ &1.5 - 0.5(y_1 - \hat{y}_1) + 0.5(y_2 - \hat{y}_2) \leq 0 \\ &y_1 - y_2 \geq 3. \end{aligned}$$

<u>*Iteration 2*</u>*:*

The new Master Problem MP2:

$$\begin{aligned} &\textit{Minimize} \quad z_{lower} \\ &\textit{such that} \quad z_{lower} \geq y_1 + 4y_2 \\ &\qquad\qquad y_1 - y_2 \geq 3 \\ &\qquad\qquad y_1 \geq 0,\ y_2 \geq 0. \end{aligned}$$

The lower bound optimal solution of the original problem is: $\hat{z}_{lower} = 3$ *for* $\hat{y}_1 = 3$, $\hat{y}_2 = 0$. *We now solve the SP when* $\hat{y}_1 = 3$, $\hat{y}_2 = 0$.

$$\begin{aligned} &\textit{Minimize} \quad x_1 + 3x_2 \\ &\textit{such that} \quad -2x_1 - x_2 \geq -2 \\ &\qquad\qquad 2x_1 + 2x_2 \geq 4 \\ &\qquad\qquad x_1,\ x_2 \geq 0 \end{aligned}$$

Now the primal subproblem SP is feasible with an optimal solution of 6 with primal variables being equal to $\hat{x}_1 = 0$, $\hat{x}_2 = 2$ *and dual variables being equal to* $\hat{u}_1 = 2.0$, $\hat{u}_2 = 2.5$

(values were read from optimal Simplex table for SP). Thus the upper bound solution of the original problem is: $\hat{z}_{upper} = \boldsymbol{c}^T\hat{\boldsymbol{x}} + \boldsymbol{f}^T\hat{\boldsymbol{y}} = 6 + \hat{y}_1 + 4\hat{y}_2 = 6 + 3 = 9$. *Since* $\hat{z}_{upper} = 9 > z_{lower} = 3$ *we continue to the next iteration. As well we need to introduce a new cut (feasibility cut),* $z_{lower} \geq \boldsymbol{f^T y} + w(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T \boldsymbol{u}$ *to MP2 to form MP3. For* $\hat{y}_1 = 3$, $\hat{y}_2 = 0$, *the new Benders' cut can be written as:*

$$
\begin{aligned}
&z_{lower} \geq \boldsymbol{f^T y} + w(\hat{\boldsymbol{y}}) - (\boldsymbol{y} - \hat{\boldsymbol{y}})^T C^T \hat{\boldsymbol{u}} \\
&z_{lower} \geq y_1 + 4y_2 + 6 + 0.5(y_1 - \hat{y}_1) - 3.5(y_2 - \hat{y}_2) \\
&z_{lower} \geq 4.5 + 1.5y_1 + 0.5y_2.
\end{aligned}
$$

Iteration 3:

The new Master Problem MP3:

$$
\begin{aligned}
&\textit{Minimize} \quad z_{lower} \\
&\textit{such that} \quad z_{lower} \geq y_1 + 4y_2 \\
&\qquad\qquad\;\; z_{lower} \geq 4.5 + 1.5y_1 + 0.5y_2 \\
&\qquad\qquad\;\; y_1 - y_2 \geq 3 \\
&\qquad\qquad\;\; y_1 \geq 0,\ y_2 \geq 0.
\end{aligned}
$$

Solving the above, the new lower bound optimal solution of the original problem is: $\hat{z}_{lower} = 9$ *for* $\hat{y}_1 = 3, \hat{y}_2 = 0$. *Note that the SP will remain the same as in iteration 2 since* $\hat{y}_i, i = 1, 2$ *values are the same and thus giving the same optimal solution of 6 and same* $\hat{z}_{upper} = 9$. *Note the process terminates since* $\hat{z}_{upper} = \hat{z}_{lower} = 9$. *Thus,* $x_1 = 0$, $x_2 = 2$, $y_1 = 3$, *and* $y_2 = 0$ *minimizes the the original problem P1 and* $P1_{min} = \{x_1 + 3x_2 + y_1 + 4y_2\}_{min} = 9$.

3.5 The Algorithm with a Relaxed Master Problem

Armed with the standard decomposition strategy of BD algorithm (Section 3.2) we may now study another way of decomposing the original problem where the formulation of the master problem is slightly different.

Here we consider the original problem (P1) of the following general form:

$$\begin{aligned} Minimize \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{y} \\ such\ that \quad & A\boldsymbol{x} + B\boldsymbol{y} \geq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{y} \in \mathbb{Y} \end{aligned}$$

where non-complicating variables $\boldsymbol{x}$, complicating variables $\boldsymbol{y}$, matrices A, B, and vectors $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{f}$ with appropriate dimensions. The relaxed master problem RMP is what we obtain when P1 is written in terms of $\boldsymbol{y}$ variables as follows:

$$\begin{aligned} Minimize \quad & \boldsymbol{f}^T\boldsymbol{y} + q \\ such\ that \quad & \boldsymbol{y} \in \mathbb{Y} \end{aligned}$$

where q is the optimal objective function value of the primal subproblem SP:

$$\begin{aligned} Minimize \quad & \boldsymbol{c}^T\boldsymbol{x} \\ such\ that \quad & A\boldsymbol{x} \geq \boldsymbol{b} - B\hat{\boldsymbol{y}} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

which is an LP for a given value $\hat{\boldsymbol{y}} \in \mathbb{Y}$. As before if the SP is unbounded for some $\hat{\boldsymbol{y}} \in \mathbb{Y}$ that implies that the RMP as well as P1 in unbounded. So assuming the boundedness of the SP, q can be found by solving the DSP:

$$\begin{aligned} Maximize \quad & (\boldsymbol{b} - B\hat{\boldsymbol{y}})^T\boldsymbol{u} \\ such\ that \quad & A^T\boldsymbol{u} \leq \boldsymbol{c} \\ & \boldsymbol{u} \geq \boldsymbol{0} \end{aligned}$$

where $\boldsymbol{u}$ is the dual variable associated with $A\boldsymbol{x} \geq \boldsymbol{b} - B\hat{\boldsymbol{y}}$. A basic remark is that the feasible region of the DSP is independent of $\hat{\boldsymbol{y}}$, which only influences the objective function. Hence, if the DSP is infeasible then the SP is either unbounded (thus making P1 unbounded), or the SP is infeasible (thus making P1 infeasible). Now we will modify the classical BD algorithm to comply with the relaxation of the MP.

Algorithm 2 The Classical Algorithm with a Relaxed Master Problem from [33]

1. Solve RMP1 and find an initial lower bound solution $\hat{q}_{lower}$ and the corresponding $\hat{\boldsymbol{y}}$.
2. Solve DSP to get an upper bound solution $\hat{q}_{upper}$.
   - If $\hat{q}_{lower} = \hat{q}_{upper}$ the process is terminated. Otherwise, add a new constraint (feasibility cut) $q \geq (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}}$ to RMP1 to form RMP2 and proceed to step 3.
   - If DSP is unbounded (that is, SP is infeasible), then introduce a new cut (infeasibility cut) $(\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}} \leq \boldsymbol{0}$ to RMP1 to form RMP2 and proceed to step 3.
   - If DSP is infeasible (that is SP is unbounded or infeasible), the original problem P1 will have either no feasible solution or an unbounded solution. So the process terminates.
3. Solve the updated relaxed master problem RMP2 to find a new lower bound solution $\hat{q}_{lower}$ and corresponding $\hat{\boldsymbol{y}}$. In the following RMP2 formulation, either the feasibility cut (first constraint) or the infeasibility cut (second constraint) can be used as discussed in Step 2.

$$\begin{aligned} &Minimize && \boldsymbol{f}^T\boldsymbol{y} + q \\ &such\ that && q \geq (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}} \\ & && (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}} \leq \boldsymbol{0} \\ & && \boldsymbol{y} \in \mathbb{Y},\ q\ unbounded \end{aligned}$$

   Then return to step 2 to solve the the dual of the subproblem DSP again.

There are further recent studies [11] which have highlighted the fact that the BD algorithm makes the master problem lose all the data related to the non-complicating variables resulting in instability, irregular succession of the bounds, and too many iterations. [12] is one of many examples of a non-standard decomposition strategy where all the variables are kept in the master problem while relaxing the integrality condition to enhance the rate of convergence of the algorithm; even if the difficulty of the master problem is clearly increased.

To illustrate the concepts in this section, we consider the following simple example.

Example 3.

Solve the following MILP, P1 [19] using the BD algorithm discussed above.

$$\begin{aligned} \textit{Minimize} \quad & 2x_1 + 3x_2 + 2y_1 \\ \textit{such that} \quad & x_1 + 2x_2 + y_1 \geq 3 \\ & 2x_1 - x_2 + 3y_1 \geq 4 \\ & x_1,\, x_2 \geq 0,\, y_1 \geq 0 \end{aligned}$$

Comparing with the basic model for BD:

$\boldsymbol{c}^T = \begin{bmatrix} 2 & 3 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.

Iteration 1:

Relaxed Master Problem (RMP1)

$$\begin{aligned} \textit{Minimize} \quad & 2y_1 + q \\ \textit{such that} \quad & \\ & y_1 \geq 0,\, q \geq 0 \end{aligned}$$

Dual Subproblem (DSP)

$$\begin{aligned} \textit{Maximize} \quad & (3 - \hat{y}_1)u_1 + (4 - 3\hat{y}_1)u_2 \\ \textit{such that} \quad & \\ & u_1 + 2u_2 \leq 2 \\ & 2u_1 - u_2 \leq 3 \\ & u_1,\, u_2 \geq 0 \end{aligned}$$

The initial lower bound solution to q is: $\hat{q}_{lower} = 0$ *with* $\hat{y}_1 = 0$. *Now we solve the DSP when* $\hat{y}_1 = 0$.

$$\begin{aligned} \textit{Maximize} \quad & 3u_1 + 4u_2 \\ \textit{such that} \quad & u_1 + 2u_2 \leq 2 \\ & 2u_1 - u_2 \leq 3 \\ & u_1,\, u_2 \geq 0 \end{aligned}$$

Analyzing graphically we can see that there exists four extreme points $(0, 0)$, $(0, 1)$, $(1.6, 0.2)$, *and* $(1.5, 0)$ *in the feasible region of the DSP. The dual optimal solution is 5.6 dual variables being equal to* $u_1 = 1.6, u_2 = 0.2$ *which implies* $\hat{q}_{upper} = 5.6$. *Since* $\hat{q}_{upper} = 5.6 > \hat{q}_{lower} = 0$ *we continue to the next iteration. Thus we need to introduce a new cut (feasibility cut),*

$q \geq (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}}$ *to RMP1 to form RMP2. For* $u_1 = 1.6, u_2 = 0.2$*, the new Benders' cut can be written as:*

$$
\begin{aligned}
& q \geq (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}} \\
& q \geq \left(\begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} y_1 \end{bmatrix} \right)^T \begin{bmatrix} 1.6 \\ 0.2 \end{bmatrix} \\
& q \geq 5.6 - 2.2y_1
\end{aligned}
$$

Iteration 2:

The new Relaxed Master Problem RMP2:

$$
\begin{aligned}
& \textit{Minimize} \quad 2y_1 + q \\
& \textit{such that} \quad q \geq 5.6 - 2.2y_1 \\
& \qquad\qquad y_1 \geq 0,\ q \geq 0
\end{aligned}
$$

Solving the above, we get the objective function value to be 5.091 *at* $y_1 = 2.545$, $q = 0$ *and thus the new lower bound optimal solution to* q *is:* $\hat{q}_{lower} = 0$ *with* $\hat{y}_1 = 2.545$*. Now we solve the DSP when* $\hat{y}_1 = 2.545$*.*

$$
\begin{aligned}
& \textit{Maximize} \quad 0.455u_1 - 3.635u_2 \\
& \textit{such that} \quad u_1 + 2u_2 \leq 2 \\
& \qquad\qquad 2u_1 - u_2 \leq 3 \\
& \qquad\qquad u_1,\ u_2 \geq 0
\end{aligned}
$$

The dual optimal solution is 0.6825 dual variables being equal to $u_1 = 1.5, u_2 = 0$ *which implies* $\hat{q}_{upper} = 0.6825$*. Since* $\hat{q}_{upper} = 0.68253 > \hat{q}_{lower} = 0$ *we continue to the next iteration. Thus we need to introduce a new cut (feasibility cut),* $q \geq (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}}$ *to RMP2 to form RMP3. For* $u_1 = 1.5, u_2 = 0$*, the new Benders' cut can be written as:*

$$
\begin{aligned}
& q \geq (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}} \\
& q \geq \left(\begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} y_1 \end{bmatrix} \right)^T \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} \\
& q \geq 4.5 - 1.5y_1
\end{aligned}
$$

Iteration 3:

The new Relaxed Master Problem RMP3:

$$\begin{aligned} &Minimize && 2y_1 + q \\ &such\ that && q \geq 5.6 - 2.2y_1 \\ & && q \geq 4.5 - 1.5y_1 \\ & && y_1 \geq 0,\ q \geq 0 \end{aligned}$$

Solving the above, we get the objective function value to be 5.286 *at* $y_1 = 1.571$, $q = 2.143$ *and thus the new lower bound optimal solution to* q *is:* $\hat{q}_{lower} = 2.143$ *with* $\hat{y}_1 = 1.571$. *Now we solve the DSP when* $\hat{y}_1 = 1.571$.

$$\begin{aligned} &Maximize && 1.429u_1 - 0.713u_2 \\ &such\ that && u_1 + 2u_2 \leq 2 \\ & && 2u_1 - u_2 \leq 3 \\ & && u_1,\ u_2 \geq 0 \end{aligned}$$

The dual optimal solution is 2.1438 dual variables being equal to $u_1 = 1.6, u_2 = 0.2$ *which implies* $\hat{q}_{upper} = 2.1438$. *Note that the process terminates since* $\hat{q}_{upper} \approx \hat{q}_{lower}$. *Thus,* $x_1 = 0$, $x_2 = 0.714$, *and* $y_1 = 1.571$ *minimizes the the original problem P1 and* $P1_{min} = \{2x_1 + 3x_2 + 2y_1\}_{min} = 5.284$.

In this chapter, we directly stated the BD algorithm and exercised it on several examples to promote the importance of BD algorithm in tackling certain large-scale optimization problems. Equipped with the relevant background machinery, we are now ready to formally state the BD algorithm and provide a mathematical reasoning why this method works.

4.0 The Algorithm and its Justification

Equipped with the machinery from the previous chapter, we are now ready to see what BD algorithm looks like mathematically. This chapter will also provide the mathematical justification to the algorithm, and we may begin by stating some definitions and proving an important theorem and two lemmas that play a major role in the above reasoning. While earlier versions of the algorithm constitute part of J.F. Benders' doctoral dissertation, we refer to [1] which contains a more detailed description of the computational aspects of the method. Consider a mixed variables programming problem of the following form:

$$\max \{\boldsymbol{c}^T\boldsymbol{x} + f(\boldsymbol{y}) \,|\, A\boldsymbol{x} + \boldsymbol{F}(\boldsymbol{y}) \leq \boldsymbol{b},\, \boldsymbol{x} \in \mathbb{R}^p,\, \boldsymbol{y} \in S\} \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^p$, $\boldsymbol{y} \in \mathbb{R}^q$, S is an arbitrary subset of $\mathbb{R}^q$, $A \in \mathbb{R}^{m\times p}$, $f(\boldsymbol{y})$ is a scalar function on S, $\boldsymbol{F}(\boldsymbol{y})$ an m-component vector function on S, and $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^p$ are fixed vectors. Note that any LP problem can be considered as being of type (4.1) after an arbitrary partitioning of the variables into two mutually exclusive subsets which may be easily achieved if the structure of the problem specifies a natural partitioning of the variables. Throughout this chapter $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{z}$ represent vectors in $\mathbb{R}^m$ whereas u_0, x_0 and z_0 denote scalars. We now provide relevant definitions and a basic theorem that are critical for our discussion.

If $A \in \mathbb{R}^{m\times p}$ and $\boldsymbol{c} \in \mathbb{R}^p$ are the matrix and the vector appearing in the formulation of the problem (4.1) we define:

$$C = \{(u_0, \boldsymbol{u}) \,|\, A^T\boldsymbol{u} - \boldsymbol{c}u_0 \geq \boldsymbol{0},\, \boldsymbol{u} \geq \boldsymbol{0},\, u_0 \geq 0\} \subset \mathbb{R}^{m+1}.$$

$$C_0 = \{\boldsymbol{u} \,|\, A^T\boldsymbol{u} \geq \boldsymbol{0},\, \boldsymbol{u} \geq \boldsymbol{0}\} \subset \mathbb{R}^m.$$

$$P = \{\boldsymbol{u} \,|\, A^T\boldsymbol{u} \geq \boldsymbol{c},\, \boldsymbol{u} \geq \boldsymbol{0}\} \subset \mathbb{R}^m.$$

We state an equivalent form to problem (4.1) by introducing a scalar variable x_0.

$$\max\{x_0 \,|\, x_0 - \boldsymbol{c}^T\boldsymbol{x} - f(\boldsymbol{y}) \leq 0,\ A\boldsymbol{x} + \boldsymbol{F}(\boldsymbol{y}) \leq \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{y} \in S\} \tag{4.2}$$

That is $(\bar{x}_0, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is an optimal solution of (4.2) if and only if $\bar{x}_0 = \boldsymbol{c}^T\bar{\boldsymbol{x}} - f(\bar{\boldsymbol{y}})$ and $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is an optimal solution of (4.1). For any point $(u_0, \boldsymbol{u}) \in C$ we link the following region in $\mathbb{R}^{q+1}$: $\{(x_0, \boldsymbol{y}) \,|\, u_0x_0 + \boldsymbol{u}^T\boldsymbol{F}(\boldsymbol{y}) - u_0f(\boldsymbol{y}) \leq \boldsymbol{u}^T\boldsymbol{b},\ \boldsymbol{y} \in S\}$. We take G to represent the intersection of all these regions: $G = \bigcap\limits_{(u_0,\boldsymbol{u})\in C} \{(x_0, \boldsymbol{y}) \,|\, u_0x_0 + \boldsymbol{u}^T\boldsymbol{F}(\boldsymbol{y}) - u_0f(\boldsymbol{y}) \leq \boldsymbol{u}^T\boldsymbol{b},\ \boldsymbol{y} \in S\}$ which is in fact the solution space of (4.1). C being a pointed convex polyhedral cone, it is the convex hull of finitely many extreme half lines which then says that there are H points (u_0^h, u^h), $h = 1, ..., H$ in C so that $G = \bigcup\limits_{h\leq H} \{(x_0, \boldsymbol{y}) \,|\, u_0^hx_0 + (\boldsymbol{u}^h)^T\boldsymbol{F}(\boldsymbol{y}) - u_0^hf(\boldsymbol{y}) \leq (\boldsymbol{u}^h)^T\boldsymbol{b},\ \boldsymbol{y} \in S\}$. We then recall *Farkas' Theorem*: is a solvability theorem for a finite system of linear inequalities.

Theorem 4 (Farkas' Theorem)**.**
Let $B \in \mathbb{R}^{m\times n}$ and $\boldsymbol{d} \in \mathbb{R}^m$. Then exactly one of the following two statements is true:

1. *There exists an $\boldsymbol{x} \in \mathbb{R}^n$ such that $B\boldsymbol{x} = \boldsymbol{d}$ and $\boldsymbol{x} \geq \boldsymbol{0}$.*
2. *There exists a $\boldsymbol{y} \in \mathbb{R}^m$ such that $B^T\boldsymbol{y} \geq \boldsymbol{0}$ and $\boldsymbol{d}^T\boldsymbol{y} < 0$.*

The *Partitioning Theorem for Mixed Variables Programming Problems* which we state next refers to problems (4.3) and (4.4) below:

$$\max\{x_0 \,|\, (x_0, \boldsymbol{y}) \in G\} \tag{4.3}$$

$$\max\{\boldsymbol{c}^T\boldsymbol{x} \,|\, A\boldsymbol{x} \leq \boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}),\ \boldsymbol{x} \geq \boldsymbol{0}\}. \tag{4.4}$$

Theorem 5 (Partitioning Theorem for Mixed Variables Programming Problems from [1])**.**

1. *Problem (4.1) is infeasible if and only if the programming problem (4.3) is infeasible, that is if and only if the set G is empty.*
2. *Problem (4.1) is feasible without having an optimal solution (unbounded), if and only if problem (4.3) is feasible without having an optimal solution.*
3. *If $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.1) and $\bar{x}_0 = \boldsymbol{c}^T\bar{\boldsymbol{x}} - f(\bar{\boldsymbol{y}})$, then $(\bar{x}_0, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.3) and $\bar{\boldsymbol{x}}$ is an optimal solution of the linear programming problem (4.4).*

4. *If $(\bar{x}_0, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.3), then problem (4.4) is feasible and the optimal value of the objective function in this problem is equal to $\bar{x}_0 - f(\bar{\boldsymbol{y}})$. If $\bar{\boldsymbol{x}}$ is an optimal solution of problem (4.4), then $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.1), with optimal value $\bar{x}_0$ for the objective function.*

Proof. Take x_0^* to be an arbitrary number and $\boldsymbol{y}^*$ be an arbitrary point in S. By Farkas' Theorem it follows that the linear system

$$\begin{aligned} A\boldsymbol{x} &\leq \boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y}^*) \\ -\boldsymbol{c}^T\boldsymbol{x} &\leq -x_0^* + f(\boldsymbol{y}^*),\ \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

is feasible if and only if $u_0 x_0^* + \boldsymbol{u}^T\boldsymbol{F}(\boldsymbol{y}^*) - u_0 f(\boldsymbol{y}^*) \leq \boldsymbol{u}^T\boldsymbol{b}$ for any point $(u_0, \boldsymbol{u}) \in C$.

Hence if $(x_0^*, \boldsymbol{x}^*, \boldsymbol{y}^*)$ is a feasible solution of problem (4.2), $(x_0^*, \boldsymbol{y}^*)$ is a feasible solution of problem (4.3). Conversely if $(x_0^*, \boldsymbol{y}^*)$ is a feasible solution of problem (4.3), there exists a vector $\boldsymbol{x}^* \in \mathbb{R}^p$ such that $(x_0^*, \boldsymbol{x}^*, \boldsymbol{y}^*)$ is a feasible solution of problem (4.2). As the problems (4.1) and (4.2) are equivalent, we have proved items (1) and (2) of Theorem 5. Further, it follows that if $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.1) and $\bar{x}_0 = \boldsymbol{c}^T\bar{\boldsymbol{x}} + f(\bar{\boldsymbol{y}})$, then $(\bar{x}_0, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.3). Finally if $(\bar{x}_0, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.3), there is a vector $\bar{\boldsymbol{x}} \in \mathbb{R}^p$, such that $(\bar{x}_0, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.2). Then $\bar{x}_0 = \boldsymbol{c}^T\bar{\boldsymbol{x}} + f(\bar{\boldsymbol{y}})$ and since $\boldsymbol{c}^T\boldsymbol{x} + f(\bar{\boldsymbol{y}}) \leq \bar{x}_0$ for any feasible solution $(\boldsymbol{x}, \bar{\boldsymbol{y}})$ of problem (4.1) ($\bar{\boldsymbol{y}}$ fixed) it follows that $\bar{\boldsymbol{x}}$ is an optimal solution of problem (4.4) which then completes the proof of Theorem 5. □

We see that item 1 and 2 of Theorem 5 address boundary cases and the interesting aspect is item 3 where we have optimality. It is where we actually have solutions (that is the solutions exist and are bounded) and thus item 3 is the heart of BD algorithm.

Theorem 5 does not demand any further requirements of the subset S and of the functions $f(\boldsymbol{y})$ and $\boldsymbol{F}(\boldsymbol{y})$ defined on S. Yet in practice those must have such properties that problem (4.3) can be solved by existing methods; that is, it must be possible to derive whether this problem is infeasible or unbounded, or it should be possible to find an optimal solution if one exists. If these assumptions are met, Theorem 5 declares that problem (4.1) can be solved by a two-step procedure. The first step includes the solution of problem (4.3), heading to

the conclusion that problem (4.1) is infeasible or unbounded, or to the optimal value of the objective function in problem (4.1) and to an optimal vector $\bar{\boldsymbol{y}} \in S$. In the latter case a second step is essential for calculating an optimal vector $\bar{\boldsymbol{x}} \in \mathbb{R}^p$, which is found by solving (4.4).

A direct solution of problem (4.3) would need the calculation of a complete set of constraints, establishing the set G. By looking at the definition of G we see that this could be done by calculating all extreme half-lines of the convex polyhedral cone C which is impractical due to the massive calculating effort associated. Nevertheless, as we are concerned about an optimal solution of problem (4.3) instead of the set G itself, it would be enough to compute only those constraints of G which establish an optimal solution. Next we develop an efficient procedure for computing such constraints.

From now on wards we assume that the set S is closed and bounded, and that $f(\boldsymbol{y})$ and the components of $\boldsymbol{F}(\boldsymbol{y})$ are continuous functions on a subset $\bar{S}$ of $\mathbb{R}^q$ containing S. These assumptions are met in most applications and they exclude difficulties produced by feasible programming problems with no solutions. It can occur that S is not bounded explicitly in the formulation of problem (4.1). In that case we can introduce bounds for the components of $\boldsymbol{y}$ which are large enough that either it is known in advance that there exists an optimal solution agreeing with these bounds or that components of $\boldsymbol{y}$ higher than these bounds have no realistic explanation.

Lemma 1 (from [1])**.**
If problem (4.3) is feasible and S is bounded, then x_0 has no upper bound on G if and only if P is empty.

Proof. By the assumptions it follows that there exists at least one point $(x_0^*, \boldsymbol{y}^*) \in G$. If P is empty, then $u_0 = 0$ for any point $(u_0, \boldsymbol{u}) \in C$. Thus G takes the form $G = \bigcap_{\boldsymbol{u} \in C_0} \{(x_0, \boldsymbol{y}) \,|\, \boldsymbol{u}^T \boldsymbol{F}(\boldsymbol{y}) \leq \boldsymbol{u}^T \boldsymbol{b},\, \boldsymbol{y} \in S\}$, and $(x_0, \boldsymbol{y}^*) \in G$ for any value of x_0. Hence x_0 has no upper bound on G.

If P is not empty, there exists at least one point $(1, \bar{\boldsymbol{u}}) \in C$. Thus for any feasible solution $(x_0, \boldsymbol{y})$ of problem (4.3), by the assumptions imposed on S, $f(\boldsymbol{y})$, and $\boldsymbol{F}(\boldsymbol{y})$ we get that $x_0 \leq \max_{\boldsymbol{y} \in S} \{\bar{\boldsymbol{u}}^T \boldsymbol{b} - \bar{\boldsymbol{u}}^T \boldsymbol{F}(\boldsymbol{y}) + f(\boldsymbol{y})\} < \infty$. Hence x_0 has an upper bound on G. □

Take Q to be a non-empty subset of C and define $G(Q) \subset \mathbb{R}^{q+1}$ by:

$$G(Q) = \bigcap_{(u_0,\boldsymbol{u})\in Q} \{(x_0, \boldsymbol{y}) \,|\, u_0 x_0 + \boldsymbol{u}^T \boldsymbol{F}(\boldsymbol{y}) - u_0 f(\boldsymbol{y}) \leq \boldsymbol{u}^T \boldsymbol{b},\ \boldsymbol{y} \in S\}.$$

We introduce the following programming problem

$$\max \{x_0 \,|\, (x_0, \boldsymbol{y}) \in G(Q)\}. \tag{4.5}$$

If problem (4.5) is infeasible, then so is problem (4.3) since $G \subset G(Q)$. Instead if $(\bar{x}_0, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.5) the question arises whether $(\bar{x}_0, \bar{\boldsymbol{y}})$ is also an optimal solution of problem (4.3) and, if not, how to obtain a better subset Q of C.

Lemma 2 (from [1])**.**
If $(\bar{x}_0, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.5), it is also an optimal solution of problem (4.3) if and only if $\min\{(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T \boldsymbol{u} \,|\, \boldsymbol{u} \in P\} = \bar{x}_0 - f(\bar{\boldsymbol{y}})$.

Proof. Since we assume that the maximum value of x_0 on the set $G(Q)$ is finite, Lemma 1 tells us that Q has at least one point $(u_0, \boldsymbol{u})$ where $u_0 > 0$. Thus P is not empty which in turn implies that the following LP problem is feasible.

$$\min\{(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T \boldsymbol{u} \,|\, \boldsymbol{u} \in P\} \tag{4.6}$$

First we note that an optimal solution $(\bar{x}_0, \bar{\boldsymbol{y}})$ of problem (4.5) is also an optimal solution of problem (4.3) if and only if $(\bar{x}_0, \bar{\boldsymbol{y}}) \in G$. The necessity of this condition follows easily. Further since $Q \subset C$, we have $\max \{x_0 \,|\, (x_0, \boldsymbol{y}) \in G(Q)\} \geq \max \{x_0 \,|\, (x_0, \boldsymbol{y}) \in G\}$ implying that the condition is also sufficient.

The definition of G says that, $(\bar{x}_0, \bar{\boldsymbol{y}}) \in G$ if and only if $(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T \boldsymbol{u} + (-\bar{x}_0 + f(\bar{\boldsymbol{y}}))u_0 \geq 0$ for any point $(u_0, \boldsymbol{u}) \in C$. This occurs if and only if $(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T \boldsymbol{u} \geq 0$ for any $\boldsymbol{u} \in C_0$ and $(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T \boldsymbol{u} \geq \bar{x}_0 - f(\bar{\boldsymbol{y}})$ for any $\boldsymbol{u} \in P$, that is if and only if

$$\min\{(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T \boldsymbol{u} \,|\, \boldsymbol{u} \in P\} \geq \bar{x}_0 - f(\bar{\boldsymbol{y}}).$$

Now the duality theorem for LP problems states that the LP

$$\max\{\boldsymbol{c}^T \boldsymbol{x} \,|\, A\boldsymbol{x} \leq \boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}),\ \boldsymbol{x} \geq \boldsymbol{0}\} \tag{4.7}$$

has a finite optimal solution $\bar{\boldsymbol{x}}$ so that $\boldsymbol{c}^T\bar{\boldsymbol{x}} = \min\{(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T\boldsymbol{u} \,|\, \boldsymbol{u} \in P\}$. Since $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is a feasible solution of problem (4.1), Theorem 5 and the fact $G \subset G(Q)$ provide us that

$$\boldsymbol{c}^T\bar{\boldsymbol{x}} + f(\bar{\boldsymbol{y}}) \leq \max\{x_0 \,|\, (x_0, \boldsymbol{y}) \in G\} \leq \max\{x_0 \,|\, (x_0, \boldsymbol{y}) \in G(Q)\} = \bar{x}_0$$

$$\min\{(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T\boldsymbol{u} \,|\, \boldsymbol{u} \in P\} \leq \bar{x}_0 - f(\bar{\boldsymbol{y}})$$

which completes the proof of Lemma 2. □

If the LP problem (4.6) has a finite optimal solution, at least one of the vertices of the polyhedron P is included in the set of optimal solutions. We are familiar that, in this case, the Simplex Method heads to an optimal vertex $\bar{\boldsymbol{u}}$ of P.

By Lemma 2 we know that if $(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T\bar{\boldsymbol{u}} = \bar{x}_0 - f(\bar{\boldsymbol{y}})$, we have determined an optimal solution $(\bar{x}_0, \bar{\boldsymbol{y}})$ of problem (4.3). Moreover, the Simplex Method offers us, simultaneously, an optimal solution $\bar{\boldsymbol{x}}$ of the dual LP problem (4.7) and Theorem 5 states that $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is an optimal solution of problem (4.1). If

$$(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T\bar{\boldsymbol{u}} \leq \bar{x}_0 - f(\bar{\boldsymbol{y}}) \tag{4.8}$$

the point $(1, \bar{\boldsymbol{u}})$ of C is not in Q. In this case we construct a new subset Q^* of C by including the point $(1, \bar{\boldsymbol{u}})$ in Q.

If the LP problem (4.6) has an unbounded solution, the Simplex Method heads to a vertex $\bar{\boldsymbol{u}}$ of P and to a direction vector $\bar{\boldsymbol{v}}$ of one of the extreme half-lines of C_0 so that the value of the objective function $(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T\boldsymbol{u}$ goes to infinity along the half-line $\{\boldsymbol{u} \,|\, \boldsymbol{u} = \bar{\boldsymbol{u}} + \lambda\bar{\boldsymbol{v}},\ \lambda \geq 0\}$. Besides, we have the inequality

$$(\boldsymbol{b} - \boldsymbol{F}(\bar{\boldsymbol{y}}))^T\bar{\boldsymbol{v}} < 0 \tag{4.9}$$

which indicates that the point $(0, \bar{\boldsymbol{v}})$ of C is not in Q. Here we construct a new subset Q^* of C by including the point $(0, \bar{\boldsymbol{v}})$ in Q.

In either case take $(x_0^*, \boldsymbol{y}^*)$ be an optimal solution of the LP $\max\{x_0 \,|\, (x_0, \boldsymbol{y}) \in G(Q^*)\}$. Then in the first case we have $(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y}^*))^T\bar{\boldsymbol{u}} \geq x_0^* - f(\boldsymbol{y}^*)$, whereas in the second case we have $(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y}^*))^T\bar{\boldsymbol{v}} \geq 0$. Comparing with (4.8) and (4.9) we see that $(x_0^*, \boldsymbol{y}^*) \neq (\bar{x}_0, \bar{\boldsymbol{y}})$. Moreover since $Q^* \supset Q$, we get that $G(Q^*) \subset G(Q)$, thus $x_0^* \leq \bar{x}_0$. In case the LP problem

(4.6) is unbounded, it is possible that the above-mentioned vertex $\bar{\boldsymbol{u}}$ agrees with (4.8). Then, both the point $(1, \bar{\boldsymbol{u}})$, and $(0, \bar{\boldsymbol{v}})$ are not in Q and the new subset Q^* of C can be constructed by including both points in Q. We further note that the constrained set $G(Q^*)$ is gained from $G(Q)$ by introducing the constraint $x_0 + \bar{\boldsymbol{u}}^T \boldsymbol{F}(\boldsymbol{y}) - f(\boldsymbol{y}) \leq \bar{\boldsymbol{u}}^T \boldsymbol{b}$ and/or the constraint $\bar{\boldsymbol{v}}^T \boldsymbol{F}(\boldsymbol{y}) \leq \bar{\boldsymbol{v}}^T \boldsymbol{b}$ to the set of constraints defining this set $G(Q)$.

We are now ready to discuss a finite multi step procedure for solving mixed variables programming problems of the form (4.1). Within a finite number of steps, this algorithm terminates either with the conclusion that problem (4.1) is infeasible, or that this problem is unbounded, or because an optimal solution of problem (4.1) has been found.

Algorithm 3 Multi Step Procedure for Solving Problems of the Form (4.1) from [1]

The procedure starts from a given finite subset $Q^0 \subset C$.

Initial Step.

- If $u_0 > 0$ for at least one point $(u_0, \boldsymbol{u}) \in Q^0$, go to the first part of the iterative step.
- If $u_0 = 0$ for any point $(u_0, \boldsymbol{u}) \in Q^0$, put $x_0^0 = +\infty$, take for $\boldsymbol{y}^0$ an arbitrary point of $G(Q^0)$ and go to the second part of the iterative step.
- If $G(Q^0)$ is empty, the procedure terminates: problem (4.1) is infeasible.

Iterative step, first part.

If the n-th step has to be performed, solve the programming problem

$$\max \{x_0 \,|\, (x_0, \boldsymbol{y}) \in G(Q^n)\}. \tag{4.10}$$

- If problem (4.10) is infeasible, the procedure terminates: problem (4.1) is infeasible.
- If $(x_0^n, \boldsymbol{y}^n)$ is found to be an optimal solution of problem (4.10), go to the second part of the iterative step.

Iterative step, second part.

Solve the LP problem

$$\min\{(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}))^T \boldsymbol{u} \,|\, A^T \boldsymbol{u} \geq \boldsymbol{c},\, \boldsymbol{u} \geq \boldsymbol{0}\}. \tag{4.11}$$

- If problem (4.11) is infeasible, problem (4.1) is either infeasible, or unbounded. (This situation can only be encountered in the first iterative step!)

- If problem (4.11) has a finite optimal solution $\boldsymbol{u^n}$ and

$$(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}))^T \boldsymbol{u^n} = x_0^n - f(\boldsymbol{y^n}), \tag{4.12}$$

the procedure terminates. In this case, if $\boldsymbol{x^n}$ is the optimal solution for the dual problem of problem (4.11), then $(\boldsymbol{x^n}, \boldsymbol{y^n})$ is an optimal solution of problem (4.1) and $\boldsymbol{x_0^n}$ is the optimal value of the objective function in this problem.

- Then if

$$(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}))^T \boldsymbol{u^n} < x_0^n - f(\boldsymbol{y^n}), \tag{4.13}$$

form the set

$$Q^{n+1} = Q^n \cup \{(1, \boldsymbol{u^n})\}, \tag{4.14}$$

replace the step counter n by $n+1$ and repeat the first part of the iterative step.

- If the value of the objective function in problem (4.11) tends to infinity along the half line $\{\boldsymbol{u} \,|\, \boldsymbol{u} = \boldsymbol{u^n} + \lambda \boldsymbol{v^n},\ \lambda \geq 0\}$, $\boldsymbol{u^n}$ being a vertex of P and $\boldsymbol{v^n}$ the direction of an extreme half line of C_0, while

$$(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}))^T \boldsymbol{u^n} \geq x_0^n - f(\boldsymbol{y^n}), \tag{4.15}$$

form the set

$$Q^{n+1} = Q^n \cup \{(0, \boldsymbol{v^n})\}. \tag{4.16}$$

However, if (4.15) is not satisfied, that is if

$$(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}))^T \boldsymbol{u^n} < x_0^n - f(\boldsymbol{y^n}), \tag{4.17}$$

form the set

$$Q^{n+1} = Q^n \cup \{(1, \boldsymbol{u^n}),\ (0, \boldsymbol{v^n})\}. \tag{4.18}$$

In either case replace the step counter n by $n+1$ and repeat the first part of the iterative step.

This algorithm is finite, since at each step where it does not terminate the preceding subset Q^n is expanded by the direction vector of at least one extreme half-line of the polyhedral cone C, which is not already in Q^n. Hence, within a finite number of steps, either the algorithm would terminate or a complete set of constraints establishing the set G would have been gained and Theorem 5 guarantees that the algorithm would stop after the next step. We can justify the termination guidelines as follows:

1. $G(Q^n) \supset G$ together with Theorem 5, item (1): problem (4.1) is infeasible.
2. Lemma 1 and Theorem 5, item (2): problem (4.1) is unbounded.
3. Lemma 2 and Theorem 5, item (4): optimal solution for problem (4.1) is obtained.

The relationship $G(Q^n) \supset G(Q^{n+1}) \supset G$ states that the sequence $\{x_0^n\}$ is non decreasing and $\max\{x_0 \,|\, (x_0, \boldsymbol{y}) \in G\} \leq x_0^n$ for any $n \geq 0$.

If problem (4.11) has an optimal solution $\boldsymbol{u^n}$, then its dual problem $\max\{\boldsymbol{c}^T\boldsymbol{x} \,|\, A\boldsymbol{x} \leq \boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}),\ \boldsymbol{x} \geq \boldsymbol{0}\}$ has an optimal solution $\boldsymbol{x^n}$, with $(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}))^T\boldsymbol{u^n} = \boldsymbol{c}^T\boldsymbol{x^n}$. Since $(\boldsymbol{x^n}, \boldsymbol{y^n})$ is a feasible solution of problem (4.1), by Theorem 5, item (3) we have that $(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}))^T\boldsymbol{u^n} + f(\boldsymbol{y^n}) \leq \max\{x_0 \,|\, (x_0, \boldsymbol{y}) \in G\}$. In other words we obtain upper and lower bounds for the maximum value of x_0 on the set G, or what is the same, for the maximum value of the objective function in problem (4.1) at the end of each step:

$$\max_{k \leq n}\{(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^k}))^T\boldsymbol{u^k} + f(\boldsymbol{y^k})\} \leq \max\{x_0 \,|\, (x_0, \boldsymbol{y}) \in G\} \leq x_0^n.$$

If problem (4.11) in the k-th iterative step is unbounded we have $(\boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^k}))^T\boldsymbol{u^k} = -\infty$; otherwise, it is the optimal value of the objective function in this problem.

The establishment of an initial set Q^0 will highly depend on the actual problem that we are trying to solve. Yet in any case we can start from the set Q^0 comprising only the origin of $\mathbb{R}^{m+1}$, which is always in C. The algorithm then moves to the second part of the iterative step from an arbitrary point $\boldsymbol{y}^0 \in S$, whereas x_0^0 is set to $+\infty$.

The algorithm can essentially generate the entire set Q; that is, all the extreme points of the underlying LP problem and as mentioned before, the algorithm does converge within finite number of steps. In practice we may encounter situations where the first few iterations find some decent extreme points giving us exceptional upper and lower bounds (to the

objective function of the original problem), that get really close fast enough but do not actually converge faster unless we adopt some acceleration strategies. In other words, we really cannot say anything about the rate of convergence of the algorithm even though we know that it does definitely converge.

In practice it may be more convenient to solve the dual problem

$$\max\{\boldsymbol{c}^T\boldsymbol{x} \,|\, A\boldsymbol{x} \leq \boldsymbol{b} - \boldsymbol{F}(\boldsymbol{y^n}),\, \boldsymbol{x} \geq \boldsymbol{0}\} \tag{4.19}$$

of problem (4.11), rather than this problem itself. Even though it is not included in this discussion, in [1] J.F. Benders does justify how the algorithm works with (4.19) instead of problem (4.11) consequently solving the original problem (4.1).

Having the relevant definitions, theorems, formal statement, and mathematical justification of the original algorithm related to the BD algorithm, we now move on to its extensions and applications. Immediately in the next chapter we have described five other versions of the algorithm and next we have collected several different real-world applications to showcase how to apply the BD algorithm and to show some of the many different areas of optimization and scheduling that can use the help of the algorithm.

5.0 Extensions and Generalizations of Benders' Decomposition Algorithm

In the previous sections we saw that the classical BD algorithm addresses certain classes of Mixed Integer Linear Programs for which the integer variables are complicating, and standard duality theory can be applied to the subproblem to generate cuts.

We now carry on describing how the extensions of the method have allowed it to address a broader range of optimization problems including integer subproblems, nonlinear functions, logical expressions, multi-stage programming, and stochastic optimization; where our main source of reference is [27].

The extension of BD that permits the use of Linear Programming duality in place of inference duality in the subproblem is known as **logic-based Benders' Decomposition**.

Likewise, **generalized Benders Decomposition** generalizes the use of nonlinear convex programs as subproblems.

Further in some applications it is efficient to use specialized algorithms to solve the subproblem rather than solving the subproblem explicitly as a Linear Programming problem. For instance, if the subproblem is a linear feasibility problem (that is a Linear Programming problem with no objective function), cuts based on irreducible infeasible subsets of constraints can be derived using a technique referred to as **combinatorial Benders' Decomposition** [33].

In many applications, the situation is that decisions for several groups of second-stage variables are made independently given the first-stage decisions. Thus multiple subproblems are defined and solved separately. For example, in stochastic programming models some decisions need to be taken in a first stage which is followed by the occurrence of a random event that affects the result of the first-stage decision. A resource decision can then be made in a second stage later, once the uncertainty is resolved. In such applications, second-stage recourse problems can be solved disjointedly given the first-stage decisions and hence are agreeable to parallel implementations. Also note that when applied to stochastic problems the BD algorithm is usually referred to as **L-shaped Decomposition**.

Before explaining the different versions of BD in detail, now we summarize the introduc-

tion of the chapter as follows.

	Version	Description
1	classical BD	SP is linear, and cuts are generated from standard dual
2	generalized BD	SP is nonlinear and convex
3	logic-based BD	cuts are generated through inference dual
4	combinatorial BD	SP is a linear feasibility problem
5	L-shaped Decomposition	stochastic programs where multiple SPs solved separately
6	nested BD	applying BD method to a problem more than once

Table 5: Some versions of Benders' Decomposition algorithm

5.1 Generalized Benders' Decomposition

Many of todays real-world optimization problems involve nonlinear functions and constraints. However, if the problem can be easily linearized or the nonlinearity arises only in the domain of the complicating variables, we can still apply classical BD to solve it. Otherwise, we must employ an extended BD method to tackle the problem.

It was A.M. Geoffrion (1972) who proposed the *Generalized Benders' Decomposition* [13] which solves nonlinear problems where the SP is convex. Later in 2005, A.M. Costa [9] showed that it is also specifically appealing for nonconvex nonlinear problems which can be convexified after fixing a subset of variables.

Then in 1991 N. Sahinidis and I.E. Grossmann [29] observed that for MINLP problems the generalized BD method may not achieve a global or even a local optimum. Particularly, if the objective function and some of the constraints are nonconvex or if there exist nonlinear equations, the subproblem may not lead to a unique local optimum and the master problem may remove the global optimum. However, rigorous global optimization approaches can be adopted if the continuous terms are in a special structure like bilinear, linear fractional, concave separable. Thus, the main idea is to use convex envelopes to create lower-bounding

convex MINLPs and combine them with global optimization techniques for continuous variables. This typically has the form of spatial Branch and Bound methods.

Likewise, it is possible that a naive application of the generalized BD method to a convex nonlinear problem converges to a nonstationary point, A. Grothey et al. (1999) [15]. After identifying that the convergence failure occurs due to the way in which the infeasible subproblems are tackled, they developed a procedure for feasibility restoration.

5.2 Logic-based Benders' Decomposition

Even though we can usually transform the optimization problems that comprise logic relations into regular optimization problems, the extra variables and big-M constraints often result a weak formulation. Further, it may have some integer variables and nonlinear functions not yielding a continuous linear subproblem. In these situations, we cannot apply traditional linear duality to generate classical Benders' cuts.

J.N. Hooker and G. Ottosson (2003) [18] and Hooker (2011) [16] proposed the concept of *Logic-based Benders Decomposition* which is similar to classical BD method. Logic-based BD divides a given problem into a master problem and subproblems and then uses constraint-generation techniques to progressively condense the solution space of the relaxed master problem. However, now each subproblem is an inference dual problem that develops the tightest bound for the objective function of the master problem. Then we use this bound to find cuts that are passed back to the master problem. Finally, if the master problem solution agrees with all the bounds produced by the subproblems, process terminates since the convergence has been reached; otherwise, the process continues.

This method can be applied to any form of subproblem including MILP, constraint programming, nonlinear programming or a feasibility-checking problem. Logic-based BD method does not have a standard model for the generation of valid cuts, instead they must be customized to the problem at hand, usually based on knowledge of its structure. Several successful applications of logic-based BD include, planning and scheduling (Hooker, 2007 [17]), transportation network design (Peterson and Trick, 2009 [26]), facility location/fleet

management (Zarandi, 2010 [39]) and radiation therapy (Luong, 2015 [22]).

5.3 Combinatorial Benders' Decomposition

Combinatorial BD, is in fact a particular case of logic-based BD. G. Codato and M. Fischetti (2006) [7], are known for successfully employing this method to MILP problems with a extensive amount of logical and big-M constraints. When the assignment of variable values in the master problem makes the subproblem infeasible, a combinatorial Benders' cut is introduced to the master problem. This cut, which says that at least one of the variables in the master problem must change its value, refines a logical implication from the original model and adds it to the master problem.

However, this method is ineffective for the case of continuous variables. To obtain stronger combinatorial cuts, one must identify small subsets of variables accountable for the infeasibility of the subproblem and express cuts in terms of these variables. The smallest of these subsets are known as minimum infeasible subsets. A successful application of combinatorial BD is related to the Strip Packing Problem by Côté et al. (2014) [10].

5.4 L-shaped Decomposition

Stochastic Linear Programming problems are multi-stage linear programs that contain uncertainty in at least some of the quantities involved in the problem. Note that multi-stage problems are problems in which an optimal initial decision is made, more information becomes available and then further decisions are made.

Stochastic models are generally large and difficult to solve due to the data uncertainty and their combinatorial nature. Yet, they exhibit special structures agreeable to decomposition methods. As a result, efforts were made to build various decomposition-based algorithms for these problems, and L-shaped method is when BD is adopted (Van Slyke and Wets, 1969 [34]).

5.5 Nested Benders' Decomposition

As mentioned above in the table, nested BD method is centered around the notion of employing the BD method to a problem more than once. It is mainly suitable for multi-stage (stochastic) problems (Birge, 1985 [4]) where each pair of adjacent stages can be studied individually.

Here, the scenario tree is considered as a collection of nested two-stage problems and the BD method is utilized repeatedly. Every problem related to an inner node in the tree is both master problem to its children and a subproblem of its parent. However, after solving the problems at a given stage, we must select the sequencing procedure; whether to forward the info related to the primal down to the leaf nodes or send the info related to the dual up to the root node. C. Wolf (2014) [38] addressed this issue and some acceleration strategies.

Further, there are applications (J. Naoum-Sawaya and S. Elhedhli, 2010 [24]) which shows that the nested BD method can be employed to solve well established single-stage problems, specially when we want to simplify the master problem by cutting down the number of integer variables.

6.0 Applications

After discussing theory behind BD algorithm and its extensions, we continue the discussion to show how it can be used on actual everyday problems. We start with an application on classical BD (Algorithm 1), precisely "The Facility Location Problem". We will then move our attention to another important application, "The Intensity Modulated Radiation Therapy Problem" which again employs classical BD but with a relaxed master problem (Algorithm 2). Then, we switch focus to introducing some complicating real-world challenges: "Simultaneous Aircraft Routing and Crew Scheduling", "Hydrothermal Scheduling", "The Concrete Delivery Problem", and "The Lock Scheduling Problem" benefiting from classical BD, generalized BD, logic-based BD, and combinatorial BD respectively.

6.1 The Facility Location Problem

First we will describe the general problem we are trying to solve; next we will present a specific example which appears in Section 10.3 of "Large scale linear and integer optimization: A unified approach" [23]. We also run some iterations of the algorithm to permit the reader to get a feel of how BD works.

The idea behind the facility location problem is to select the best among possible factories, subject to constraints involving the demands of several customers, which must be satisfied by the established factories. This defines the objective of the problem as: selecting factories in order to minimize costs. In fact costs typically include a part which is proportional to the sum of the distances from the customers to the factories, in addition to costs of opening them.

Practically the factories may have limited capacities for servicing, which classifies the problems as a capacited facility location problem. Yet in this thesis we will analyze an uncapacited facility location problem where there is no limit on how much each factory can produce.

6.1.1 General Problem [6]

The facility location problem demonstrates a setting with n factories and m customers where each customer has a demand that has to be fulfilled from one or more of the factories. In our model x_{ij} denotes the fraction of customer j's demand fulfilled from factory i. The cost of fulfilling demand for customer j from factory i is represented by c_{ij}. f_i denotes the cost related to establishing the factory i and the decision variable y_i indicates whether factory i is closed or opened. With these variables the LP problem can be expressed as below:

$$
\begin{aligned}
Minimize \quad & z = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{n} f_i y_i \\
such\ that \quad & \sum_{i=1}^{n} x_{ij} \geq 1,\ j = 1, ..., m \\
& x_{ij} \leq y_i,\ i = 1, ..., n,\ j = 1, ..., m \\
& x_{ij} \geq 0,\ i = 1, ..., n,\ j = 1, ..., m \\
& y_i \in \{0, 1\},\ i = 1, ..., n.
\end{aligned}
$$

The objective function minimizes the sum of factory opening costs and transportation costs. The first constraint states that each customers demand must be satisfied. The second constraint forces factory i to be opened if some customer j is assigned to it which indicates that we can only satisfy the demand from a factory which is opened.

Given we already know which factories are opened the subproblem is intended to derive the transportation scheme. So by fixing the values of $\boldsymbol{y}$ to $\hat{\boldsymbol{y}}$ in the above LP problem we can state the primal subproblem as follows:

$$
\begin{aligned}
Minimize \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\
such\ that \quad & \sum_{i=1}^{n} x_{ij} \geq 1,\ j = 1, ..., m \\
& -x_{ij} + \hat{y}_i \geq 0,\ i = 1, ..., n,\ j = 1, ..., m \\
& x_{ij} \geq 0,\ i = 1, ..., n,\ j = 1, ..., m.
\end{aligned}
$$

We now want to form the dual problem where $i \in O(\hat{\boldsymbol{y}})$ represents the factories which are *open* whereas $i \in C(\hat{\boldsymbol{y}})$ represents the factories which are *closed*. We denote the dual variables related to the demand constraints by v_j and the dual variables related to the setup constraints (that is open/closed duals) by w_{ij}. Then for $j = 1, ..., m$:

$$v_j = \min_{i \in O(\hat{\boldsymbol{y}})} \{c_{ij}\}$$

$$w_{ij} = \begin{cases} 0 & for \quad i \in O(\hat{\boldsymbol{y}}) \\ \max\limits_{i \in C(\hat{\boldsymbol{y}})} \{(v_j - c_{ij}), 0\} & for \quad i \in C(\hat{\boldsymbol{y}}). \end{cases}$$

We will here briefly explain the meaning of the above formulation. $v_j = \min\limits_{i \in O(\hat{\boldsymbol{y}})} \{c_{ij}\}$, $j = 1, ..., m$ implies that for each customer, we will choose the *open* factory that has the least cost. $w_{ij} = 0\ , i \in O(\hat{\boldsymbol{y}})$ says that we will ignore the setup constraint if factory i is open because adding more capacity to an already open factory will not change the cost of the solution. Note that it can never cost besides the fixed costs, f_i to open a facility; hence, w_{ij} is always greater or equal to zero. The most we can gain by opening a factory is the difference between the current cost, v_j and the cost if this i^{th} factory was available, c_{ij}. So $w_{ij} = \max\limits_{i \in C(\hat{\boldsymbol{y}})} \{(v_j - c_{ij}), 0\},\ i \in C(\hat{\boldsymbol{y}})$, tells that if a factory is closed then we could improve our solution by $v_j - c_{ij}$ for customer j by opening factory i, if the value is positive. Now, we refer to the dual variable as $\boldsymbol{u}$ which is in fact $\boldsymbol{u} = [\boldsymbol{v}, \boldsymbol{w}]$, where $\boldsymbol{v}$ denotes the demand duals and $\boldsymbol{w}$ denotes the open/closed factory duals. Then depending on the structure of the problem Benders' feasibility cut can be written as follows:

$$\hat{z}_{lower} \geq \boldsymbol{f^T y} + (\boldsymbol{d} - C\boldsymbol{y})^T \hat{\boldsymbol{u}}$$

$$\hat{z}_{lower} \geq \sum_{i=1}^{m} \hat{v}_j + \sum_{i=1}^{n} \left(f_i - \sum_{j=1}^{m} \hat{w}_{ij} \right) y_i.$$

We will see that it is much convenient to proceed with the specific version of the feasibility cut derived based on the problem structure rather than the general matrix version that we discussed in Chapter 3. Yet, comparing the LP formulation for the facility location problem (Subsection 6.1.1) with the basic model for BD (Section 3.2) and identifying the vectors and matrices associated the problem is the easiest way to justify the above reformation of the feasibility cut.

Its important to note that the facility location problem is not a standard optimization problem that can easily be solved using Excel because of the binary variable $\boldsymbol{y}$ which turns it into an integer program. Although it can be solved using an MIP solver it is much faster to solve it using the BD method. We explain the way that BD works in simple terms as follows.

Once we choose to open certain factories: we either can transport everything with which ever factories we decided to open, or we won't. In the first case there is a particular cost incurred, and BD gives us a feasibility cut implying that if we want it to be cheaper, we cannot just achieve it through the choice we made. In the second case that is, if we cannot transport everything, BD provides an infeasibility cut suggesting that we must open a new factory. Since we consider an uncapacited facility location problem it will always be feasible although it might introduce a huge cost. So here we will only obtain feasibility cuts which we introduce to the MP and then solve the updated MP to identify which factories should be opened to lower the total cost. Now since the dual subproblem has an analytic solution, we just generate numbers instead of solving an LP which in turn makes the BD method a much faster way to tackle this problem.

6.1.2 An Actual Example [6]

We will now demonstrate a simple example problem and run the algorithm on this problem. Our example comprises of 3 possible factories and 5 possible customers. The following table records the c_{ij}'s as well as the f_i's.

	Customers	
Factory	1 2 3 4 5	Fixed costs
1	2 3 4 5 7	2
2	4 3 1 2 6	3
3	5 4 2 1 3	3

Table 6: Basic data corresponding to Example 6.1.2 from [6]

As we described in the general formulation; x_{ij} denotes the fraction of customer j's demand satisfied from factory i, c_{ij} is the cost of satisfying demand for customer j from factory i, f_i is the cost of opening factory i, and y_i is the decision variable that indicates whether factory i is opened or closed. Comparing with the basic model for BD:

$\boldsymbol{c}^T = \Big[2,3,4,5,7,4,3,1,2,6,5,4,2,1,3\Big]_{1\times 15}$,

$\boldsymbol{f}^T = \Big[2,3,3\Big]_{1\times 3}$,

$\boldsymbol{d}^T = \Big[1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\Big]_{1\times 20}$,

$\boldsymbol{y}^T = \Big[y_1,\ y_2,\ y_3\Big]_{1\times 3}$,

$\boldsymbol{x}^T = \Big[x_{1j},\ x_{2j},\ x_{3j}\Big]_{1\times 15}$, j= 1,...,5,

$\boldsymbol{u}^T = \Big[v_1,\ v_2,\ v_3,\ v_4,\ v_5,\ w_{1j},\ w_{2j},\ w_{3j}\Big]_{1\times 20}$, j= 1,...,5,

$$B = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1\\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}_{20\times 15}, C = \begin{bmatrix}
0 & 0 & 0\\
0 & 0 & 0\\
0 & 0 & 0\\
0 & 0 & 0\\
0 & 0 & 0\\
1 & 0 & 0\\
1 & 0 & 0\\
1 & 0 & 0\\
1 & 0 & 0\\
1 & 0 & 0\\
0 & 1 & 0\\
0 & 1 & 0\\
0 & 1 & 0\\
0 & 1 & 0\\
0 & 1 & 0\\
0 & 1 & 0\\
0 & 0 & 1\\
0 & 0 & 1\\
0 & 0 & 1\\
0 & 0 & 1\\
0 & 0 & 1
\end{bmatrix}_{20\times 3}.$$

<u>*Iteration 1:*</u>

Master Problem (MP1)

Minimize z_{lower}

such that

$z_{lower} \geq 2y_1 + 3y_2 + 3y_3$

$y_i \in \{0,1\},\ i = 1,...,3$

Dual Subproblem (DSP)

for $j = 1,...,5$:

$v_j = \min\limits_{i\in O(\hat{\boldsymbol{y}})} \{c_{ij}\}$

$$w_{ij} = \begin{cases} 0 & for \quad i \in O(\hat{\boldsymbol{y}}) \\ \max\limits_{i\in C(\hat{\boldsymbol{y}})} \{(v_j - c_{ij}), 0\} & for \quad i \in C(\hat{\boldsymbol{y}}) \end{cases}$$

To start the iterations, we simply choose only to open factory 1. So, $\hat{y}_1 = 1$, $\hat{y}_2 = 0$, $\hat{y}_3 = 0$ and the lower bound optimal solution to the original problem is: $\hat{z}_{lower} = 2$. Now we solve the DSP when $\hat{\boldsymbol{y}}^T = [1, 0, 0]$.

Since $O(\hat{\boldsymbol{y}})$ only contains factory 1, $v_j = \min_{i \in O(\hat{\boldsymbol{y}})} \{c_{ij}\} = c_{1j}\ for\ j = 1, ..., 5$. This very simply gives us: $\hat{v}_1 = 2,\ \hat{v}_2 = 3,\ \hat{v}_3 = 4,\ \hat{v}_4 = 5,\ \hat{v}_5 = 7$. Next we find w_{ij} using the formulas, $w_{ij} = 0$, $i \in O(\hat{\boldsymbol{y}})$ and $w_{ij} = \max_{i \in C(\hat{\boldsymbol{y}})} \{(v_j - c_{ij}), 0\}$, $i \in C(\hat{\boldsymbol{y}})$. Since factory 1 is open, $\hat{w}_{1j} = 0$ $for\ j = 1, ..., 5$. As factory 2 is a closed facility, $w_{2j} = \max\{(v_j - c_{2j}), 0\}\ for\ j = 1, ..., 5$ which results: $\hat{w}_{21} = 0,\ \hat{w}_{22} = 0,\ \hat{w}_{23} = 3,\ \hat{w}_{24} = 3,\ \hat{w}_{25} = 1$; implying that there is nothing to be gained for customer 1 and 2 if we open factory 2, but there is a gain for customers 3 through 5. Similarly, since factory 3 is a closed facility, $w_{3j} = \max\{(v_j - c_{3j}), 0\}$ $for\ j = 1, ..., 5$ which leads to: $\hat{w}_{31} = 0,\ \hat{w}_{32} = 0,\ \hat{w}_{33} = 2,\ \hat{w}_{34} = 4,\ \hat{w}_{35} = 4$; stating that there is no gain for customers 1 and 2 but there is a gain for 3 through 5 if we open factory 3. Thus, the DSP has a solution, and we can update $\hat{z}_{upper}$ as:

$$\begin{aligned}
\hat{z}_{upper} &= \boldsymbol{f^T}\hat{\boldsymbol{y}} + (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} \\
&= \sum_{i=1}^{5} \hat{v}_j + \sum_{i=1}^{3} \left(f_i - \sum_{j=1}^{5} \hat{w}_{ij} \right) \hat{y}_i \\
&= (2 + 3 + 4 + 5 + 7) + (2 - 0)\hat{y}_1 + (3 - (3 + 3 + 1))\hat{y}_2 + (3 - (2 + 4 + 4))\hat{y}_3 \\
&= 21 + 2\hat{y}_1 - 4\hat{y}_2 - 7\hat{y}_3 \\
&= 23 \qquad\qquad with\ \ \hat{y}_1 = 1,\ \hat{y}_2 = 0,\ \hat{y}_3 = 0.
\end{aligned}$$

Since $\hat{z}_{upper} = 23 > \hat{z}_{lower} = 2$ we continue to the next iteration. Now we need to add a new Benders' cut (feasibility cut) to MP1 to generate MP2, which is:

$$\begin{aligned}
z_{lower} &\geq \boldsymbol{f^T}\boldsymbol{y} + (\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}} \\
z_{lower} &\geq \sum_{i=1}^{5} \hat{v}_j + \sum_{i=1}^{3} \left(f_i - \sum_{j=1}^{5} \hat{w}_{ij} \right) y_i \\
z_{lower} &\geq 21 + 2y_1 - 4y_2 - 7y_3.
\end{aligned}$$

Recall that these iterations must be repeated until the upper and lower bound are the same which will then yield the optimal solution (that is the minimum total cost for satisfying all the demands) and prescribe which factories are to be opened.

Iteration 2:

The new master problem MP2:

$$
\begin{aligned}
&Minimize \quad z_{lower} \\
&such\ that \quad z_{lower} \geq 2y_1 + 3y_2 + 3y_3 \\
&\qquad\qquad z_{lower} \geq 21 + 2y_1 - 4y_2 - 7y_3 \\
&\qquad\qquad y_i \in \{0,1\},\ i = 1, ..., 3.
\end{aligned}
$$

As can easily be seen the optimal solution to this problem is to choose $\hat{y}_1 = 0$, $\hat{y}_2 = 1$, *and* $\hat{y}_3 = 1$, that is keep factory 1 closed and open factories 2 and 3, yielding a solution of 10 (our new lower bound). So we update $\hat{z}_{lower} = 10$ and use this new $\hat{\boldsymbol{y}}$ to do the second iteration. So, we solve the DSP when $\hat{\boldsymbol{y}}^T = [0, 1, 1]$.

Since $O(\hat{\boldsymbol{y}})$ contains factories 2 and 3, $v_j = \min\limits_{i \in O(\hat{\boldsymbol{y}})} \{c_{ij}\} = \min\{c_{2j}, c_{3j}\}\ for\ \ j = 1, ..., 5.$ This results in: $\hat{v}_1 = 4,\ \hat{v}_2 = 3,\ \hat{v}_3 = 1,\ \hat{v}_4 = 1,\ \hat{v}_5 = 3$. Next we find w_{ij} using the formulas, $w_{ij} = 0,\ i \in O(\hat{\boldsymbol{y}})$ and $w_{ij} = \max\limits_{i \in C(\hat{\boldsymbol{y}})} \{(v_j - c_{ij}), 0\},\ i \in C(\hat{\boldsymbol{y}})$. Since factory 2 and 3 are open, $\hat{w}_{2j} = \hat{w}_{3j} = 0\ for\ j = 1, ..., 5$. As factory 1 is closed , $w_{1j} = \max\{(v_j - c_{1j}), 0\}\ for\ j = 1, ..., 5$ which gives us: $\hat{w}_{11} = 2,\ \hat{w}_{12} = 0,\ \hat{w}_{13} = 0,\ \hat{w}_{14} = 0,\ \hat{w}_{15} = 0$. We obtained a feasible solution to the DSP and thus we update $\hat{z}_{upper}$ as:

$$
\begin{aligned}
\hat{z}_{upper} &= \boldsymbol{f^T}\hat{\boldsymbol{y}} + (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} \\
&= \sum_{i=1}^{5} \hat{v}_j + \sum_{i=1}^{3} \left(f_i - \sum_{j=1}^{5} \hat{w}_{ij} \right) \hat{y}_i \\
&= (4 + 3 + 1 + 1 + 3) + (2 - 2)\hat{y}_1 + (3 - 0)\hat{y}_2 + (3 - 0)\hat{y}_3 \\
&= 12 + 3\hat{y}_2 + 3\hat{y}_3 \\
&= 18 \qquad\qquad with\ \ \hat{y}_1 = 0,\ \hat{y}_2 = 1,\ \hat{y}_3 = 1.
\end{aligned}
$$

Since $\hat{z}_{upper} = 18 > \hat{z}_{lower} = 10$ we continue to the next iteration. Again we need to add a new Benders' cut (feasibility cut) to MP2 to generate MP3, which is:

$$
\begin{aligned}
z_{lower} &\geq \boldsymbol{f^T}\boldsymbol{y} + (\boldsymbol{d} - C\boldsymbol{y})^T\hat{\boldsymbol{u}} \\
z_{lower} &\geq \sum_{i=1}^{5} \hat{v}_j + \sum_{i=1}^{3} \left(f_i - \sum_{j=1}^{5} \hat{w}_{ij} \right) y_i \\
z_{lower} &\geq 12 + 3y_2 + 3y_3.
\end{aligned}
$$

Iteration 3:

The new master problem MP3:

$$
\begin{aligned}
&Minimize && z_{lower} \\
&such\ that && z_{lower} \geq 2y_1 + 3y_2 + 3y_3 \\
& && z_{lower} \geq 21 + 2y_1 - 4y_2 - 7y_3 \\
& && z_{lower} \geq 12 + 3y_2 + 3y_3 \\
& && y_i \in \{0,1\},\ i = 1, ..., 3.
\end{aligned}
$$

We can find the optimal solution to this problem as $\hat{y}_1 = 0$, $\hat{y}_2 = 0$, *and* $\hat{y}_3 = 1$, that is keep factories 1 and 2 closed and open factory 3, yielding a solution of 15 (our new lower bound). So we update $\hat{z}_{lower} = 15$ and use this new $\hat{\boldsymbol{y}}$ to do the third iteration. Now we solve the DSP when $\hat{\boldsymbol{y}}^T = [0, 0, 1]$.

Since $O(\hat{\boldsymbol{y}})$ only contains factory 3, $v_j = \min\limits_{i \in O(\hat{\boldsymbol{y}})} \{c_{ij}\} = c_{3j}\ for\ j = 1, ..., 5$. This gives us: $\hat{v}_1 = 5,\ \hat{v}_2 = 4,\ \hat{v}_3 = 2,\ \hat{v}_4 = 1,\ \hat{v}_5 = 3$. Since we have factory 3 to be open, $\hat{w}_{3j} = 0$ $for\ j = 1, ..., 5$. As factory 1 is closed, $w_{1j} = \max\{(v_j - c_{1j}), 0\}\ for\ j = 1, ..., 5$ which leads to: $\hat{w}_{11} = 3,\ \hat{w}_{12} = 1,\ \hat{w}_{13} = 0,\ \hat{w}_{14} = 0,\ \hat{w}_{15} = 0$. Further, factory 2 being closed yields to $w_{2j} = \max\{(v_j - c_{2j}), 0\}\ for\ j = 1, ..., 5$ which gives us: $\hat{w}_{21} = 1,\ \hat{w}_{22} = 1,\ \hat{w}_{23} = 1,\ \hat{w}_{24} = 0$, $\hat{w}_{25} = 0$. With this feasible solution to the DSP we can update $\hat{z}_{upper}$ as:

$$
\begin{aligned}
\hat{z}_{upper} &= \sum_{i=1}^{5} \hat{v}_j + \sum_{i=1}^{3} \left(f_i - \sum_{j=1}^{5} \hat{w}_{ij} \right) \hat{y}_i \\
&= (5 + 4 + 2 + 1 + 3) + (2 - 4)\hat{y}_1 + (3 - 3)\hat{y}_2 + (3 - 0)\hat{y}_3 \\
&= 15 - 2\hat{y}_1 + 3\hat{y}_3 \\
&= 18 \qquad\qquad with\ \ \hat{y}_1 = 0,\ \hat{y}_2 = 0,\ \hat{y}_3 = 1.
\end{aligned}
$$

Since $\hat{z}_{upper} = 18 > \hat{z}_{lower} = 15$ we continue to the next iteration by adding a new Benders' cut (feasibility cut) to MP3 to generate MP4, which is:

$$
\begin{aligned}
z_{lower} &\geq \boldsymbol{f^T y} + (\boldsymbol{d} - C\boldsymbol{y})^T \hat{\boldsymbol{u}} \\
z_{lower} &\geq \sum_{i=1}^{5} \hat{v}_j + \sum_{i=1}^{3} \left(f_i - \sum_{j=1}^{5} \hat{w}_{ij} \right) y_i \\
z_{lower} &\geq 15 - 2y_1 + 3y_3.
\end{aligned}
$$

Iteration 4:

The new master problem MP4:

$$
\begin{aligned}
&Minimize \quad && z_{lower} \\
&such\ that \quad && z_{lower} \geq 2y_1 + 3y_2 + 3y_3 \\
& && z_{lower} \geq 21 + 2y_1 - 4y_2 - 7y_3 \\
& && z_{lower} \geq 12 + 3y_2 + 3y_3 \\
& && z_{lower} \geq 15 - 2y_1 + 3y_3 \\
& && y_i \in \{0,1\},\ i = 1, ..., 3.
\end{aligned}
$$

The optimal solution to this problem is $\hat{y}_1 = 1$, $\hat{y}_2 = 0$, *and* $\hat{y}_3 = 1$, that is keep factory 2 closed and open factories 1 and 3, yielding a solution of 16. So we update $\hat{z}_{lower} = 16$ and use this new $\hat{\boldsymbol{y}}$ to do the fourth iteration. Now we solve the DSP when $\hat{\boldsymbol{y}}^T = [1, 0, 1]$.

Since $O(\hat{\boldsymbol{y}})$ contains factories 1 and 3, $v_j = \min\limits_{i \in O(\hat{\boldsymbol{y}})} \{c_{ij}\} = \min\{c_{1j}, c_{3j}\}\ for\ j = 1, ..., 5.$ This gives us: $\hat{v}_1 = 2,\ \hat{v}_2 = 3,\ \hat{v}_3 = 2,\ \hat{v}_4 = 1,\ \hat{v}_5 = 3$. Since we have factory 1 and 3 to be open, $\hat{w}_{1j} = \hat{w}_{3j} = 0\ for\ j = 1, ..., 5$. As factory 2 is closed, $w_{2j} = \max\{(v_j - c_{2j}), 0\}$ $for\ j = 1, ..., 5$ which leads to: $\hat{w}_{21} = 0,\ \hat{w}_{22} = 0,\ \hat{w}_{23} = 1,\ \hat{w}_{24} = 0,\ \hat{w}_{25} = 0$. With this feasible solution to the DSP we can update $\hat{z}_{upper}$ as:

$$
\begin{aligned}
\hat{z}_{upper} &= \boldsymbol{f^T}\hat{\boldsymbol{y}} + (\boldsymbol{d} - C\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} \\
&= \sum_{i=1}^{5} \hat{v}_j + \sum_{i=1}^{3} \left(f_i - \sum_{j=1}^{5} \hat{w}_{ij} \right) \hat{y}_i \\
&= (2 + 3 + 2 + 1 + 3) + (2 - 0)\hat{y}_1 + (3 - 1)\hat{y}_2 + (3 - 0)\hat{y}_3 \\
&= 11 + 2\hat{y}_1 + 2\hat{y}_2 + 3\hat{y}_3 \\
&= 16 \qquad\qquad with\ \ \hat{y}_1 = 1,\ \hat{y}_2 = 0,\ \hat{y}_3 = 1.
\end{aligned}
$$

Since $\hat{z}_{upper} = 16 = \hat{z}_{lower}$ the process has converged. We have arrived at the optimal solution of opening factories 1 and 3 which incurs a minimum total cost of 16 for satisfying all the demands. That brings us to the end of the discussion where we applied the BD algorithm to a concrete example of the facility location problem realizing how it makes use of the simple sub structure when fixing the open factories. Now we move on to the next section where we illustrate how the BD algorithm works on a problem encountered in Intensity Modulated Radiation Therapy (IMRT) treatment planning.

6.2 The Intensity Modulated Radiation Therapy Problem

Here we consider a matrix segmentation problem appearing in Intensity Modulated Radiation Therapy (IMRT) treatment planning, which is explained in detail in "Mixed-integer programming techniques for decomposing IMRT fluence maps using rectangular apertures" [32].

We are given a matrix of intensity values that are to be delivered to a patient from a specified angle and we are working under the condition that the IMRT device can only deliver radiation through rectangular apertures. An aperture is denoted by a binary matrix where ones occur successively in each row and column, and thus forming a rectangular shape. A **feasible segmentation** is one in which the original given intensity matrix equals the weighted sum of a number of feasible binary matrices, where the weight of each binary matrix is in fact the amount of intensity to be transported through the corresponding aperture. In IMRT context, what we want is to minimize this setup time in treating patients. So, that defines our mathematical goal as: finding a matrix segmentation that utilizes the minimum number of aperture matrices to segment the desired intensity matrix. We first consider a simple example that shows an intensity matrix and a feasible segmentation using three rectangular apertures:

$$\begin{bmatrix} 4 & 11 & 0 \\ 4 & 13 & 2 \\ 0 & 9 & 2 \end{bmatrix} = 4 \times \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 2 \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} + 7 \times \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

6.2.1 General Problem [33]

Let $P_{m \times n}$ be the given intensity matrix where the element (i, j) requires $p_{ij} \in \mathbb{Z}$ units of intensity. Let R be the set of all possible rectangular apertures (that is binary matrices of size $m \times n$ having contiguous rows and columns) that can be utilized in the segmentation of P. For each rectangle $r \in R$, let x_r denote the intensity assigned to rectangle r, and let y_r represent a binary variable that equals 1 if rectangle r is utilized in decomposing P (that is if $x_r > 0$), and equals 0 otherwise. The element (i, j) is said to be **covered** by rectangle r if the (i, j) element of r is 1. We take $C(r)$ to be the set of matrix elements that is covered

by rectangle r. Also, we define $M_r = \min_{(i,j)\in C(r)} \{p_{ij}\}$ as the minimum intensity requirement among the elements of P that are covered by rectangle r. Finally, let $R(i,j)$ denote the set of rectangles that cover element (i,j). With the understanding of above definitions, we now formulate the problem as follows:

$$
\begin{aligned}
Minimize \quad & \sum_{r\in R} y_r \\
such\ that \quad & \sum_{r\in R(i,j)} x_r = p_{ij},\ i = 1,...,m,\ j = 1,...,n \\
& x_r \leq M_r y_r \quad \forall r \in R \\
& x_r \geq 0, \quad y_r \in \{0,1\} \quad \forall r \in R.
\end{aligned}
$$

The objective function minimizes the number of rectangular apertures used in the segmentation. The first constraint ensures that each matrix element receives exactly the required dose, whereas the second constraint introduces the condition that x_r cannot be positive unless $y_r = 1$. Finally, last two constraints represent bounds and logical restrictions on the variables.

Before we discuss about the decomposition approach, we identify the complications that arise in solving the above model, which can be alleviated by utilizing BD. In practical clinical instances the above model which contains two variables and a constraint for each rectangle develops into a large-scaled MIP. Moreover, the M_r terms in second constraint lead to a weak Linear Programming relaxation due to the big-M structure.

The BD approach first selects a subset of the rectangles via the MP, and then employs the SP to check whether the input matrix can be decomposed using only the selected rectangles. We formulate the MP in terms of the $\boldsymbol{y}$ variables as:

$$
\begin{aligned}
Minimize \quad & \sum_{r\in R} y_r \\
& such\ that\ \boldsymbol{y}\ corresponds\ to\ a\ feasible\ segmentation \\
& y_r \in \{0,1\} \quad \forall r \in R.
\end{aligned}
$$

Then for a given vector $\hat{\boldsymbol{y}}$ which denotes a chosen subset of rectangles, we can examine whether the constraints in the MP are met by solving the following SP:

$$
\begin{aligned}
&Minimize \quad 0 \\
&such\ that \quad \sum_{r \in R(i,j)} x_r = p_{ij},\ i = 1, ..., m,\ j = 1, ..., n \\
&\qquad\qquad x_r \leq M_r \hat{y}_r \quad \forall r \in R \\
&\qquad\qquad x_r \geq 0 \quad \forall r \in R.
\end{aligned}
$$

If $\hat{\boldsymbol{y}}$ is related to a feasible segmentation, then SP is feasible and otherwise it is infeasible. Note that the MP is an Integer Programming problem whereas SP is an LP problem. Moreover the second constraint in the SP reduces to a simple upper bound on $\boldsymbol{x}$ variable for a given $\hat{\boldsymbol{y}}$, avoiding the big-M issue linked to the second constraint in the original problem. Given $\hat{\boldsymbol{y}}$, if SP has a feasible solution $\hat{\boldsymbol{x}}$, then ($\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$) establishes a feasible solution to the original problem. Then again, if SP does not produce a feasible solution, then we need to make sure that $\hat{\boldsymbol{y}}$ is removed from the feasible region of the MP. BD employs the theory of LP duality to reach this goal. If we associate variables α_{ij}, and β_r with first and second constraint of the SP respectively, the dual formulation DSP becomes:

$$
\begin{aligned}
&Maximize \quad \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} \alpha_{ij} + \sum_{r \in R} M_r \hat{y}_r \beta_r \\
&such\ that \quad \sum_{(i,j) \in C(r)} \alpha_{ij} + \beta_r \leq 0, \quad \forall r \in R \\
&\qquad\qquad \alpha_{ij}\ \ unbounded,\ i = 1, ..., m,\ j = 1, ..., n \\
&\qquad\qquad \beta_r \leq 0 \quad \forall r \in R.
\end{aligned}
$$

The BD algorithm first solves the MP to optimality, which yields $\hat{\boldsymbol{y}}$. If SP has a feasible solution $\hat{\boldsymbol{x}}$, then ($\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$) relates to an optimal matrix segmentation. Yet, if DSP is unbounded (that is SP is infeasible) we add an infeasibility cut $(\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}} \leq \boldsymbol{0}$ to the MP and re-solve it in the next iteration to gain a new candidate to optimal solution. Depending on the problem structure, the infeasibility cut can we stated as follows:

$$
\sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} \hat{\alpha}_{ij} + \sum_{r \in R} (M_r \hat{\beta}_r) y_r \leq 0
$$

where $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}})$ is an extreme ray of DSP.

Now we consider a minor variation of the matrix segmentation problem, with the new goal of minimizing a weighted combination of the number of matrices used in the segmentation (corresponding to setup time) and the sum of the matrix coefficients (corresponding to *beam-on-time*). In IMRT treatment planning setting, this objective resembles minimizing total treatment time. To include this change in our formulation, we simply change the objective function of the original problem to:

$$Minimize \quad w \sum_{r \in R} y_r + \sum_{r \in R} x_r$$

where the parameter w represents the average setup time per aperture relative to the time needed to deliver a unit of intensity. So the BD algorithm discussed above must be modified accordingly. We first add a variable $t \in \mathbb{R}$ to the new MP, which represents the minimum beam-on-time that can be gained by the set of rectangles selected. Thus, the formulation of the relaxed master problem RMP is:

$$\begin{aligned} &Minimize \quad w \sum_{r \in R} y_r + t \\ &such\ that\ \boldsymbol{y}\ corresponds\ to\ a\ feasible\ segmentation \\ &t \geq minimum\ beam-on-time\ corresponding\ to\ \boldsymbol{y} \\ &t \geq 0, \quad y_r \in \{0,1\}\ \forall r \in R. \end{aligned}$$

Given $\hat{\boldsymbol{y}}$, the minimum beam-on-time for the corresponding segmentation (if one exists) can be found by solving the SP corresponding to the RMP which we denote by (R)SP:

$$\begin{aligned} &Minimize \quad \sum_{r \in R} x_r \\ &such\ that \quad \sum_{r \in R(i,j)} x_r = p_{ij}, \quad \forall i = 1, ..., m,\ j = 1, ..., n \\ &\qquad x_r \leq M_r \hat{y_r} \quad \forall r \in R \\ &\qquad x_r \geq 0,\ \forall r \in R. \end{aligned}$$

Taking α_{ij} and β_r be dual variables related with constraints in (R)SP respectively, the corresponding dual DSP which we represent by (R)DSP can be written as:

$$\begin{aligned} Maximize \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}\alpha_{ij} + \sum_{r\in R} M_r \hat{y}_r \beta_r \\ such\ that \quad & \sum_{(i,j)\in C(r)} \alpha_{ij} + \beta_r \leq 1, \quad \forall r \in R \\ & \alpha_{ij}\ \ unbounded,\ i = 1, ..., m,\ j = 1, ..., n \\ & \beta_r \leq 0 \quad \forall r \in R. \end{aligned}$$

Note that (R)SP is derived by merely altering the objective function of SP, and (R)DSP is derived by altering the right-hand side of first constraint in DSP. If (R)DSP is unbounded, then we introduce an infeasibility cut of as before and solve the new RMP. Otherwise, take the value of t in RMP be $\hat{t}_{lower}$, and the optimal objective function value of (R)DSP be $\hat{t}_{upper}$. If $\hat{t}_{lower} = \hat{t}_{upper} = \hat{t}$, then $(\hat{\boldsymbol{y}}, \hat{t})$ is an optimal solution of RMP, that minimizes the total treatment time. However, if $\hat{t}_{lower} < \hat{t}_{upper}$, then we need to introduce a new constraint. BD, once again adopts LP duality theory to develop such a constraint. Letting α_{ij} and β_r be corresponding dual variables, the following constraint (feasibility cut: $t \geq (\boldsymbol{b} - B\boldsymbol{y})^T \hat{\boldsymbol{u}}$) meets our requirement.

$$t \geq \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}\hat{\alpha}_{ij} + \sum_{r\in R}(M_r\hat{\beta}_r)y_r.$$

6.2.2 An Actual Example [33]

Now we focus on a simple numerical example illustrating the steps of BD algorithm on our matrix segmentation problem. Consider the input matrix $P = \begin{bmatrix} 8 & 3 \\ 5 & 0 \end{bmatrix}$. The set of rectangular apertures that can be used to segment P is:

$$R = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \right\}.$$

Let the average setup time per aperture relative to the time needed to deliver a unit of intensity be $w = 7$. Specifying an x_r and a y_r variable for each rectangle $r = 1, \ldots, 5$, the

goal of minimizing total treatment time can be stated as an MIP as follows (that is the original problem (P1) according to the terminology in previous section):

$$
\begin{aligned}
\textit{Minimize} \quad & 7 \times (y_1 + y_2 + y_3 + y_4 + y_5) + x_1 + x_2 + x_3 + x_4 + x_5 \\
\textit{such that} \quad & x_1 + x_4 + x_5 = 8 \\
& x_2 + x_5 = 3 \\
& x_3 + x_4 = 5 \\
& x_1 \leq 8y_1,\ x_2 \leq 3y_2,\ x_3 \leq 5y_3,\ x_4 \leq 5y_4,\ x_5 \leq 3y_5 \\
& x_r \geq 0,\ \ y_r \in \{0,1\}\ \ \forall r = 1, \ldots, 5.
\end{aligned}
$$

Iteration 1:

The Relaxed Master Problem (RMP1) can be stated as:

$$
\begin{aligned}
\textit{Minimize} \quad & 7 \times (y_1 + y_2 + y_3 + y_4 + y_5) + t \\
\textit{such that} \quad & t \geq 0, \quad y_r \in \{0,1\}\ \forall r = 1, \ldots, 5.
\end{aligned}
$$

For a given $\hat{\boldsymbol{y}}$, the Primal Subproblem (R)SP can be given as:

$$
\begin{aligned}
\textit{Minimize} \quad & x_1 + x_2 + x_3 + x_4 + x_5 \\
\textit{such that} \quad & x_1 + x_4 + x_5 = 8 \\
& x_2 + x_5 = 3 \\
& x_3 + x_4 = 5 \\
& x_1 \leq 8\hat{y}_1,\ x_2 \leq 3\hat{y}_2,\ x_3 \leq 5\hat{y}_3,\ x_4 \leq 5\hat{y}_4,\ x_5 \leq 3\hat{y}_5 \\
& x_r \geq 0,\ \ \forall r = 1, \ldots, 5.
\end{aligned}
$$

Associating dual variables α_{11}, α_{12}, α_{21} with first three constraints of (R)SP and $\beta_1, \ldots, \beta_5$ with last constraint of (R)SP, we get the Dual Subproblem (R)DSP.

$$
\begin{aligned}
\textit{Maximize} \quad & 8\alpha_{11} + 3\alpha_{12} + 5\alpha_{21} + 8\hat{y}_1\beta_1 + 3\hat{y}_2\beta_2 + 5\hat{y}_3\beta_3 + 5\hat{y}_4\beta_4 + 3\hat{y}_5\beta_5 \\
\textit{such that} \quad & \alpha_{11} + \beta_1 \leq 1 \\
& \alpha_{12} + \beta_2 \leq 1 \\
& \alpha_{21} + \beta_3 \leq 1 \\
& \alpha_{11} + \alpha_{21} + \beta_4 \leq 1 \\
& \alpha_{11} + \alpha_{12} + \beta_5 \leq 1 \\
& \alpha_{11},\ \alpha_{12},\ \alpha_{21} \ \textit{unbounded} \\
& \beta_r \leq 0, \ \ \forall r = 1, \ldots, 5.
\end{aligned}
$$

To start the iterations, we solve the RMP1 which has the optimal solution $\hat{y}_1 = \hat{y}_2 = \hat{y}_3 = \hat{y}_4 = \hat{y}_5 = 0, \textit{and } \hat{t} = 0$. So we update $\hat{t}_{lower} = 0$ and solve the (R)DSP when $\hat{\boldsymbol{y}}^T = [0, 0, 0, 0, 0]$.

We set the objective function to: max $8\alpha_{11} + 3\alpha_{12} + 5\alpha_{21}$, and solve (R)DSP. It is unbounded having an extreme ray: $\alpha_{11} = 2$, $\alpha_{12} = -1$, $\alpha_{21} = -1$, $\beta_1 = -2$, $\beta_2 = 0$, $\beta_3 = 0$, $\beta_4 = -1$, $\beta_5 = -1$. So we introduce the following Benders' infeasibility cut to RMP1 to generate RMP2.

$$
\begin{aligned}
\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}\hat{\alpha}_{ij} + \sum_{r\in R} M_r\hat{\beta}_r y_r \leq 0 \\
8 - 16y_1 - 5y_4 - 3y_5 \leq 0
\end{aligned}
$$

<u>*Iteration 2:*</u>

The new relaxed master problem RMP2:

$$
\begin{aligned}
\textit{Minimize} \quad & 7 \times (y_1 + y_2 + y_3 + y_4 + y_5) + t \\
\textit{such that} \quad & 8 - 16y_1 - 5y_4 - 3y_5 \leq 0 \\
& t \geq 0, \quad y_r \in \{0, 1\}\ \forall r = 1, \ldots, 5.
\end{aligned}
$$

An optimal solution to this problem is to choose $\hat{y}_1 = 1$, $\hat{y}_2 = \hat{y}_3 = \hat{y}_4 = \hat{y}_5 = 0$, *and* $\hat{t} = 0$. So still $\hat{t}_{lower} = 0$ and we use this new $\hat{\boldsymbol{y}}$ to do the second iteration. We solve the (R)DSP when $\hat{\boldsymbol{y}}^T = [1, 0, 0, 0, 0]$.

We set the objective function of (R)DSP to: max $8\alpha_{11} + 3\alpha_{12} + 5\alpha_{21} + 8\beta_1$, and solve (R)DSP. It is again unbounded having an extreme ray: $\alpha_{11} = 0$, $\alpha_{12} = 0$, $\alpha_{21} = 1$, $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = -1$, $\beta_4 = -1$, $\beta_5 = 0$. We introduce a Benders' infeasibility cut which can be written as follows to RMP2 to build RMP3.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}\hat{\alpha}_{ij} + \sum_{r\in R} M_r\hat{\beta}_r y_r \leq 0$$

$$5 - 5y_3 - 5y_4 \leq 0$$

Iteration 3:

The new relaxed master problem RMP3:

$$\begin{aligned} \textit{Minimize} \quad & 7 \times (y_1 + y_2 + y_3 + y_4 + y_5) + t \\ \textit{such that} \quad & 8 - 16y_1 - 5y_4 - 3y_5 \leq 0 \\ & 5 - 5y_3 - 5y_4 \leq 0 \\ & t \geq 0, \quad y_r \in \{0, 1\}\ \forall r = 1, \ldots, 5. \end{aligned}$$

An optimal solution to this problem is to choose $\hat{y}_1 = \hat{y}_2 = \hat{y}_3 = 0$, $\hat{y}_4 = \hat{y}_5 = 1$, *and* $\hat{t} = 0$. So still $\hat{t}_{lower} = 0$ and we use this new $\hat{\boldsymbol{y}}$ to do the third iteration. We solve the (R)DSP when $\hat{\boldsymbol{y}}^T = [0, 0, 0, 1, 1]$.

We set the objective function of (R)DSP to: max $8\alpha_{11} + 3\alpha_{12} + 5\alpha_{21} + 5\beta_4 + 3\beta_5$, and solve (R)DSP. It has an optimal solution: $\alpha_{11} = 1$, $\alpha_{12} = 1$, $\alpha_{21} = 1$, $\beta_1 = 0$, $\beta_2 = 0$, $\beta_3 = 0$, $\beta_4 = -1$, $\beta_5 = -1$, and the corresponding objective function value is: $8(1) + 3(1) + 5(1) + 5(-1) + 3(-1) = 8$. Then we update $\hat{t}_{upper} = 8$ and continue to the next iteration since $\hat{t}_{upper} = 8 > \hat{t}_{lower} = 0$. We add the following Benders feasibility cut to RMP3 to form RMP4.

$$t \geq \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}\hat{\alpha}_{ij} + \sum_{r\in R}(M_r\hat{\beta}_r)y_r$$

$$t \geq 16 - 5y_4 - 3y_5$$

Iteration 4:

The new relaxed master problem RMP4:

$$\begin{aligned} Minimize \quad & 7 \times (y_1 + y_2 + y_3 + y_4 + y_5) + t \\ such\ that \quad & 8 - 16y_1 - 5y_4 - 3y_5 \leq 0 \\ & 5 - 5y_3 - 5y_4 \leq 0 \\ & t \geq 16 - 5y_4 - 3y_5 \\ & t \geq 0, \quad y_r \in \{0,1\}\ \forall r = 1, \ldots, 5. \end{aligned}$$

An optimal solution is $\hat{\boldsymbol{y}}^T = [0,0,0,1,1]$; $\hat{t} = 8$. So we update $\hat{t}_{lower} = 8$. Note that $\hat{y}$ is equal to the solution generated in the previous iteration, and therefore $\hat{t}_{upper} = 8$. Since $\hat{t}_{upper} = 8 = \hat{t}_{lower}$, optimality has been reached and we stop. We have arrived at the optimal solution of $\hat{y}_1 = 0$, $\hat{y}_2 = 0$, $\hat{y}_3 = 0$, $\hat{y}_4 = 1$, $\hat{y}_5 = 1\, \hat{x}_1 = 0$, $\hat{x}_2 = 0, \hat{x}_3 = 0, \hat{x}_4 = 5, \hat{x}_5 = 3$, with an optimal solution value of $7 \times (y_1 + y_2 + y_3 + y_4 + y_5) + x_1 + x_2 + x_3 + x_4 + x_5 = 22$. It declares that we should use aperture 4 with an intensity of 5 units and aperture 5 with an intensity of 3 units and the total treatment time will be 22 units.

This marks the conclusion of the discussion where we applied the BD algorithm to an actual example arising in IMRT treatment planning realizing how BD makes use of the relaxation of the MP. The chapter then continues to briefly analyze some complicated real-life applications benefited by BD algorithm.

6.3 Advanced Applications

Here we take into account the four applications: Simultaneous Aircraft Routing and Crew Scheduling, Hydrothermal Scheduling, The Concrete Delivery Problem, and The Lock Scheduling Problem, where they utilize classical BD, generalized BD, logic-based BD, and combinatorial BD respectively.

6.3.1 Simultaneous Aircraft Routing and Crew Scheduling

Suppose we are given a set of flight legs to be flown by a specific type of aircraft. Then simultaneous aircraft routing and crew scheduling problem involves identifying a minimum-cost set of aircraft routes and crew pairings so that we allocate one aircraft and one crew for each flight leg, ensuring side constraints are met. Even though some side constraints such as maximum flight time and maintenance requirements depend only on crews or aircraft, linking constraints enforce minimum connection times for crews that involve aircraft connections. To manage these linking constraints, a solution approach based on BD is proposed in [8]. The proposed method reiterates between a master problem which solves the aircraft routing problem, and a subproblem which solves the crew pairing problem.

6.3.2 Hydrothermal Scheduling

In short-term hydrothermal scheduling, the transmission network is usually formulated with dc power flow techniques. However, such modeling, can be directed to impractical solutions when it is verified with ac power flow.There are other approaches in thermal systems that focus on the ac network modeling but not the optimization of losses. The method presented in [31] focuses on issues such as congestion management and control of service quality that frequently occur in large and weakly meshed networks (the typical pattern of power systems in Latin America). Thus, it represents a new decomposition method that concentrates on the network through ac modeling within the hydrothermal scheduling optimization process including the losses. They adopt generalized Benders' Decomposition and conventional, well-known optimization techniques to solve this problem. The master problem of the suggested model describes the generation levels by considering the inter-temporal constraints. The subproblem is to define both the active and the reactive economical dispatches for every single time gap of the load curve. It satisfies the electrical constraints via a modified ac optimal power flow. They also include accelerating techniques to reduce the number of iterations and CPU time.

6.3.3 The Concrete Delivery Problem

The concrete delivery problem is a complicating optimization problem that we encounter in real life which includes the allocation and distribution of concrete to construction sites. The main scheduling concern in this problem is that consecutive deliveries to a site need to be satisfactorily close in time. In [21] they present an exact logic-based Benders Decomposition for this problem. The master problem which is centered around several characteristics such as the availability of vehicles, geographical orientation of the customers and production centers, and the customers demand for concrete; allocates concrete to customers. Then, the subproblem ensures that the schedule is feasible while satisfying all the routing and scheduling constraints. Infeasibilities in the schedule are transferred back to the master problem through several combinatorial inequalities (Benders' cuts). They adopt a Mixed Integer Programming approach to solve the master problem, and a Constraint Programming model and a dedicated scheduling heuristic to solve the subproblem.

6.3.4 The Lock Scheduling Problem

The lock scheduling problem which is in fact a combinatorial optimization question denotes an actual task faced by countless number of harbors and waterway operators. It involves three deeply interconnected subproblems: scheduling lockages, assigning ships to chambers, and positioning the ships inside the chambers which can be understood respectively as a scheduling, an assignment, and a packing problem. [35] shows that if we merge the first two problems into a master problem and employ the packing problem in the role of the subproblem, we can achieve a decomposition that can be solved efficiently by a combinatorial Benders' method . The MP is solved first, thus ordering the ships into a number of lockages. They subsequently use a packing problem as a subproblem for every lockage which then introduces several combinatorial Benders cuts to the master problem while ensuring the feasibility. This is in fact an exact method to tackle the lock scheduling problem.

7.0 Conclusion

We discussed the classical BD algorithm in greater depth followed by a mathematical verification and examples. Then we introduced the extensions to BD algorithm together with some more complicating real-world applications. Now in the concluding remarks we first discuss about the impact of the problem formulation on its convergence and the relationship to other decomposition methods. We will then move our attention to acceleration strategies, drawbacks, trends, and potential research directions.

7.1 Model Selection for Benders' Decomposition

Note that the different but equivalent formulations of a given problem may not be equivalent from a computational point of view. Geoffrion and Graves [14] identified that the functioning of the BD algorithm is also affected by the way we formulate the problem. However, tighter formulations can often be gained by introducing extra (problem dependent) constraints. Yet, we may end up with a more time-consuming subproblem, which may also have a higher degree of degeneracy. So, there must be a comparison between the reduction in the iterations and the added complexity of the subproblem before arriving at a conclusion.

7.2 Relationship to Other Decomposition Methods

The BD technique is strongly linked with other decomposition methods for LP, such as Dantzig-Wolfe and Lagrangian optimization. The subproblems in BD and Dantzig-Wolfe methods are equivalent and explaining an LP problem by Dantzig-Wolfe optimization method corresponds to applying the BD approach to its dual. Likewise applying BD is the same as employing Cutting Plane algorithm to the Lagrangian dual. However, we cannot identify such simple relationships among the decomposition methods in Integer Programming.

7.3 Shortcomings of Benders' Decomposition

We can identify the key downsides of BD as follows [27]: iterations being time consuming, getting poor feasibility and infeasibility cuts, initial iterations being ineffective, obtaining a zigzagging behavior of the primal solutions and slow convergence at the end of the algorithm, and gaining upper bounds that stay unchanged in succeeding iterations due to the existence of equivalent solutions. As a result, a direct application of the classical BD algorithm may involve enormous amount of computing time and memory.

7.4 Enhancement Strategies of Benders' Decomposition

Now we present a brief overview of ways to increase the convergence of the algorithm. This can be achieved by either advancing the quality of both the generated solutions and the cuts (therefore reducing the number of iterations) or by refining the technique used to optimize the master problem and subproblem in each iteration (therefore reducing the time consumed in each iteration). The decomposition strategy that defines the initial MP and SP determines both the difficulty of the problems and the quality of the solutions and hence is another key element of the algorithm that affects its efficiency. [27] defines a four-dimension classification, that captures all the above aspects.

Decomposition Strategy	**Solution Procedure**	**Solution Generation**	**Cut Generation**
classical	standard/standard	regular MP	classical/classical
modified	standard/advanced	improved MP	classical/improved
	advanced/standard	alternative MP	improved/classical
	advanced/advanced	heuristics	improved/improved
		hybrid	

Table 7: Classification of enhancement strategies from [27]

The **decomposition strategy** states how the problem is subdivided to get the initial master problem and subproblem.

The **solution procedure** concentrates on the algorithms used for the MP and subproblem. The standard techniques are the Simplex Method and Branch and Bound whereas advanced strategies exploit the structure of the master problem and subproblem or are designed to improve the convergence speed. Here standard/advanced denotes that standard methods are adopted to solve the master problem whereas advanced methods are utilized to solve the subproblem. Similarly, they define advanced/advanced, advanced/standard, and standard/standard.

The **solution generation** focuses on the method used to guess initial values for the complicating variables. The classical strategy is to solve the master problem without reformation (denoted as regular master problem). Otherwise, we can use heuristics, an alternative master problem, or an improved master problem to obtain solutions quicker or to derive better solutions. Hybrid approaches can also be specified, for instance; we can obtain an initial value for the master variables via the regular master problem and then upgrade it using heuristics.

The **cut generation** aims at the approach used to derive feasibility and infeasibility cuts. In the classical method this is achieved via the regular subproblem which we gain from the decomposition. Other improved strategies either reformulate the subproblem or answer auxiliary subproblems. Here classical/improved denotes that the classical strategy is used to generate feasibility cuts and the improved strategies are used to generate infeasibility cuts.

We discussed how the BD algorithm which was first intended for Mixed Integer Linear Programming problems with continuous subproblems, was expanded to tackle a broader span of problems including nonlinear, integer, multistage, and constraint programming problems. Then we extended our analysis by presenting four main classes of acceleration strategies that have been developed to enhance the classical algorithm: modifying the decomposition, solving the master problem and subproblem more effectively, generating stronger cuts, and extracting better solutions.

However, the effectiveness of these strategies is problem-dependent, and a combination of them generally leads to the best results. Yet, this does not mean that research into the Benders' Decomposition algorithm is over since there are still many challenges and open questions.

7.5 Promising Research Directions

In this dissertation, we discussed how the BD method was suitable for problems in which temporarily fixing the complicating variables makes the remaining problem significantly easier to handle. Moreover, the BD method can handle problems that experience arithmetical instability due to big-M constraints and the binary variables that switch them on and off. It is also useful to tackle bilevel optimization problems that cannot be transformed into single-level problems via the Karush-Kuhn-Tucker optimality conditions [27]. Besides, it can be applied to optimization problems for which some of the constraints are not known in advance, yet are derived iteratively.

However, it appears that even though the range of problem settings addressed by BD method is expanding, only a few survey articles that focuses on the applications of the BD method is available. So, there is a need for a comprehensive analysis concentrating on different applications of the BD algorithm.

We mentioned above that the acceleration strategies are all problem-dependent, and thus a better understanding their interconnections can lead to improve the convergence rate. Therefore, it would be of great importance to conduct a broad survey on the acceleration methodologies to better understand their limitations and consequences.

An alternative acceleration strategy is to tighten the subproblem thus generating stronger cuts. In [5] they generate Gomory mixed-integer cuts iteratively to tighten the subproblem. Further research in this area will be worthwhile as well.

In fact, the approaches that tighten the master problem generally add valid cuts only once before the initial iterations. Thus, there is also a need for a comprehensive research into the use of more sophisticated Cutting Plane techniques to tighten the master problem further at each iteration.

When it comes to deriving solutions for the complicating variables, it would be interesting to know how to generate better cuts through a careful choice from the multiple optimal solutions of the master problem or how to modify the master problem to obtain solutions with specific characteristics. Even though these notions have been effectively used in the framework of Dantzig-Wolfe decomposition, it appears that an extensive study related to

applying these ideas in BD context is yet to be performed.

In some problems the subproblem may further divide into several independent subproblems that can be optimized simultaneously. The parallel variants of the BD algorithm, is another research area that needs to be addressed thoroughly.

7.6 Commercial Software that Implements Benders' Decomposition

With all this discussion about the algorithm, it is interesting to know whether there is commercial software which has an implementation of BD algorithm built in.

CPLEX V12.7.0 and later provide an automated BD algorithm which saves programming time of those who are familiar with the algorithm while making the algorithm handy to those who know how they want to decompose the problem but are not comfortable enough with the algorithmic details to apply it [20]. In fact, it can utilize annotations that we provide for our model to divide the given formulation into a single master and (perhaps multiple) subproblems. The approach can be used to MILPs and for some forms of problems, this strategy offers substantial performance improvements. CPLEX V12.7.0 has a new annotation characteristic that allows us to specify a decomposition to our model. If we provide annotations to the formulation, it tries to refine the decomposition that we specified and then employs BD algorithm. Yet if we do not annotate our formulation to denote how we want to decompose the problem CPLEX just carries out traditional Branch and Bound.

Suppose that the annotations we provide suggest that two (or more) variables belong to different subproblems whereas problem formulation denotes that these variables share the same constraint implying that these variables are related. Therefore, the subproblems where these variables occur according to our annotations are not disjoint with respect to the partition. In such a situation CPLEX yields an error message. Thus, it is better to verify that we adopt a complete partitioning for our decomposition, that is, the subproblems and the master problem in fact define a decomposition of the original problem into subproblems which are disjoint.

In case that we use CPLEX to apply BD to our problem without providing annotations

that specify the partitioning, all integer variables will be placed in the master problem whereas continuous variables will be placed in subproblems. If we do not have either integer variables or continuous variables in our model, CPLEX will again give an error message saying that it cannot decompose the problem automatically to employ BD algorithm.

Nevertheless, if we need to implement a feature of BD algorithm which is not supported by the automated Benders' feature in a more recent version of CPLEX, we may still implement our own by means of API (application programming interface) calls.

We have seen that BD method is a technique with applications in numerous disciplines of mathematics. What began as a technique in mathematical programming problems concerning the logistics of an oil refinery has become an immensely useful method in optimization and scheduling. This thesis is only an introduction to the scope of the BD algorithm discussing: the basic information to understand and justify the algorithm, a few of the numerous applications, and an overview of its extensions.

Appendix A

Linear Programming

For those who may not be as familiar with Linear Programming (LP), this appendix is provided as a short crash course in basic material needed for our discussion. The first part of the Appendix will go through the formulation of an LP problem, then we will have a short discussion on solution methods of LP problems.

The LP method was first developed by Leonid Kantorovich in 1939, to employ through World War II to manage expenses and earnings to lessen the costs to the army while adding losses to the enemy. This method was not disclosed until 1947 when George B. Dantzig announced the Simplex Method and John von Neumann established the theory of duality as a linear optimization solution and utilized it in the discipline of game theory. In todays global environment, researchers are showing significant attention to LP methods in order to optimize resource schedules which can provide enormous benefits to a service oriented and cost-conscious business.

A.1 Preliminaries

Linear Programming deals with the optimization of a function of variables is known as the objective function, subject to a set of equalities and / or inequalities known as constraints. The adjective *linear* implies that all the constraints and the objective must afford expression as a linear function. First we will identify the basic requirements for a LP problem:

1. There must be a well-defined objective function which is to be optimized (either maximized or minimized) and which can be expressed as a function of decision variables.
2. The amount of limited resources such as money, production, time, personal, technology must be expressed as constraints for the LP problem. The constraints impose restrictions

on the activities (decision variables) in optimizing the objective function.

3. The decision variables should be interrelated and non-negative. The non-negativity shows that LP deals with real-life situations.

The objective function, the set of constraints and the non-negativity constraint together form a Linear Programming problem and we now learn the basic steps in formulating an LP.

Algorithm A.1 Linear Programming Problem Formulation

1. Write down the decision variables of the problem.
2. Formulate the objective function to be optimized as a linear function of the decision variables.
3. Formulate the system constraints, which are also linear relationships of the decision variables reflecting the limited resources of the problem.
4. Add the non-negativity constraint from the consideration so that the negative values of the decision variables do not have any physical interpretation.

If there are n decision variables and m constraints in the problem the mathematical formulation of the LP problem is:

$$
\begin{aligned}
&\textit{Maximize (or minimize)}\ Z = c_1x_1 + c_2x_2 + ... + c_nx_n \\
&\textit{subject to the constraints},\\
&\qquad\qquad a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \quad (\leq, =, \geq) \quad b_1 \\
&\qquad\qquad a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \quad (\leq, =, \geq) \quad b_2 \\
&\qquad\qquad \vdots \\
&\qquad\qquad a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \quad (\leq, =, \geq) \quad b_m \\
&\qquad\qquad x_1,\ x_2,\ ...,\ x_n \geq \quad 0
\end{aligned}
$$

where, x_j is the quantity of the j^{th} decision variable, c_j is a constant representing per unit contribution to the objective function of the j^{th} decision variable, a_{ij} is a constant representing exchange coefficients of the j^{th} decision variable i^{th} constraint, b_j is a constant representing i^{th} constraint requirement, and $x_1, x_2, ..., x_n$ are decision variables.

Now that we have a basic understanding of constructing the mathematical program for a given LP model, we continue with a brief discussion of solution methods of LPs, which comprises of two basic methods: graphical solution method and Simplex Method.

A.2 Graphical Method

An LP problem with only two variables presents a simple case for which the solution can be derived using a graphical method. Before focusing on the steps involved in graphical method, we first take a look at a related theorem which roughly states that the optimal solution to an LP problem occurs at a corner point.

Theorem A.1 (The Fundamental Theorem of Linear Programming from [36])**.**
If the optimal value of the objective function in a Linear Programming problem exists, then that value (known as the optimal solution) must occur at one or more of the corner points of the feasible region.

Due to the scope of this thesis, no attention will be given to actually proving the above theorem and instead we then concentrate on the steps related to graphical method where we utilize the theorem.

Algorithm A.2 Graphical Method

1. Formulate the LP.
2. Represent constraints as equalities on x_1, x_2 coordinate plane and find the convex region formed by them.
3. Plot the objective function
4. Find the vertices of the convex region and also the value of the objective function at each vertex. The vertex that gives the optimal value of the objective function gives the optimal solution to the problem.

It is important to be familiar with the terminology for solutions of the LP models. *Solution* is any specification of values for the decision variables; *feasible solution* is a solution for which all the constraints are satisfied; *infeasible solution* is a solution for which at least

one constraint is violated; *feasible region* is the collection of all feasible solutions; *optimal solution* is a feasible solution that has the most favorable value of the objective function; and *most favorable value* is the largest value is the objective function is to be maximized, whereas it is the smallest value if the objective function is to be minimized. There are in fact two basic methods to find the optimal solution.

1. If the problem is to find the point point in the feasible region, which maximizes the objective function we first draw the objective line when $Z = 0$ which passes through the origin. Then go on drawing lines parallel to this line till the line is farthest away from the origin and passes through only one point of the feasible region. In that case every point on that gives the maximum value of the objective function.
2. Here we determine all the corner points for the feasible region algebraically. Then evaluate the objective function at each of the corner point and identify the optimal value of the objective function.

   We now focus on some simple examples to understand the above concept better.

Example A.1.

Solve the following LP problem using the graphical method.

$$\begin{aligned} &Maximize && Z = 3x_1 + 4x_2 \\ &such\ that && x_1 + x_2 \leq 450 \\ & && 2x_1 + x_2 \leq 600 \\ & && x_1,\, x_2 \geq 0 \end{aligned}$$

Evaluating the objective function at each of the corner points:

$$\begin{aligned} A \equiv (0, 450), &\quad Z_A = 0 + 4 \times 450 = 1800 \\ B \equiv (150, 300), &\quad Z_B = 3 \times 150 + 4 \times 300 = 1650 \\ C \equiv (300, 0), &\quad Z_C = 3 \times 300 + 0 = 900 \\ O \equiv (0, 0), &\quad Z_O = 0 + 0 = 0. \end{aligned}$$

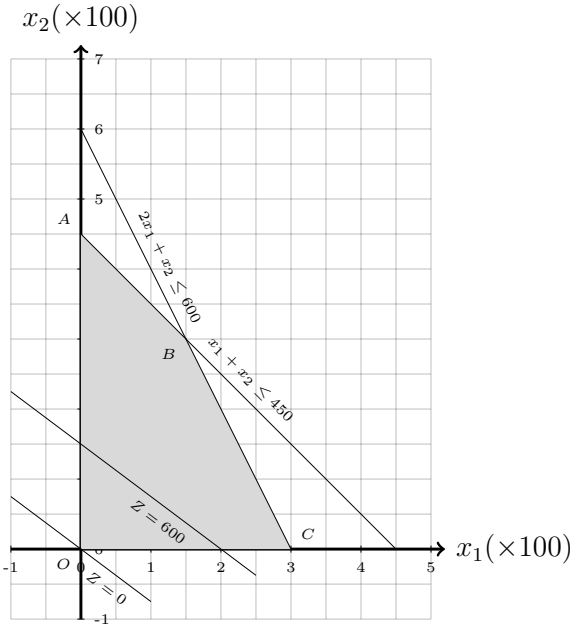

Figure A.1: Feasible region corresponding to Example A.1

Optimal solution occurs at A, where the maximum value is $Z = 1800$ *at* $x_1 = 0$, $x_2 = 450$.

Example A.2.

Solve the following LP problem using the graphical method.

$$\begin{aligned} \textit{Minimize} \quad & Z = 20x_1 + 40x_2 \\ \textit{such that} \quad & 36x_1 + 6x_2 \geq 108 \\ & 3x_1 + 12x_2 \geq 36 \\ & 20x_1 + 10x_2 \geq 100 \\ & x_1,\, x_2 \geq 0 \end{aligned}$$

Evaluating the objective function at each of the corner points:

$$A \equiv (0, 18), Z_A = 0 + 40 \times 18 = 720$$

$$B \equiv (2, 6), \quad Z_B = 20 \times 2 + 40 \times 6 = 280$$

$$C \equiv (4, 2), \quad Z_C = 20 \times 4 + 40 \times 2 = 160$$

$$D \equiv (12, 0), Z_D = 20 \times 12 + 0 = 240.$$

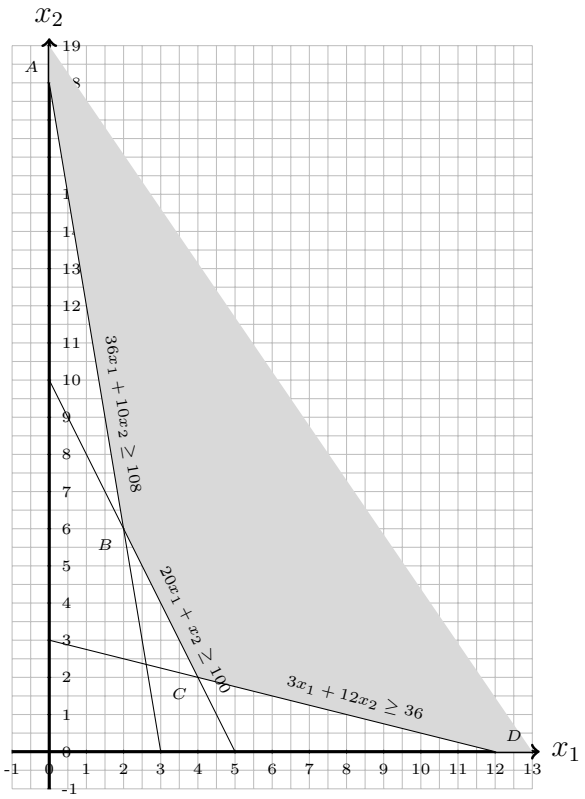

Figure A.2: Feasible region corresponding to Example A.2

Optimal solution occurs at C, where the minimum value is $Z = 160$ *at* $x_1 = 4$, $x_2 = 2$.

However, there are three types of special cases of solutions to LPs. *Multiple optimal solution* is when function falls on more than one optimal point. If there are no points that simultaneously satisfy all constraints in the problem, we call it an *infeasible problem*. In some problems the feasible solution space formed by the constraints is not confined within a closed boundary. In these cases the objective function can sometimes increase indefinitely without ever reaching its maximum limit since it never reaches a constraint boundary. We call such a problem as an *unbounded problem* .

In this section we looked at a geometric method of solving an LP where we had only two unknowns and thus had a feasible region that is a subset of the real plane. As we introduce more variables to the LP, the graph of the feasible region becomes more complicated thus demanding another method to handle the LPs.

A.3 Simplex Method

We can solve LP problems with two or more than two variables by using a systematic procedure known as Simplex Method. The Simplex Method is based on matrix algebra in which a set of simultaneous constraint equations is solved through matrix inverse procedure.

If there are n decision variables and m constraints in the problem the mathematical formulation of the LP problem (which we presented in the previous section) may be put in the following compact form.

$$\begin{aligned} &Maximize\ (or\ minimize) \quad Z = \sum_{j=1}^{n} c_j x_j \\ &subject\ to\ the\ constraints, \quad \sum_{j=1}^{n} a_{ij} x_j \quad (\leq, =, \geq) \quad b_i, \quad i = 1, 2, ..., m \\ &\qquad\qquad\qquad\qquad\qquad\qquad x_j \geq 0, \quad j = 1, 2, ..., n \end{aligned}$$

Then the *canonical form* of LP problem which has the characteristics: all decision variables are non-negative, all constraints of the $\leq$ type, and objective function is of maximization type, can be stated as below.

$$\begin{aligned} &Maximize \qquad\qquad\qquad Z = \sum_{j=1}^{n} c_j x_j \\ &subject\ to\ the\ constraints, \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, ..., m \\ &\qquad\qquad\qquad\qquad\qquad\qquad x_j \geq 0, \quad j = 1, 2, ..., n \end{aligned}$$

Any LP can be put in the canonical form by the use of some elementary transformations:

1. The minimization of a function $f(x)$ is equal to the maximization of the negative expression of this function.
2. Any inequality in one direction can be changed to an inequality in the opposite direction by multiplying both sides of the inequality by -1.
3. An equation may be replaced by two-week inequalities in opposite direction.
4. If a variable is unconstrained in sign, it is expressed as the difference between two non-negative variables.

The *standard form* of an LP problem is when it has the following properties: all the constraints are expressed in the form of equations except the non-negative constraints, the right hand side of each constraint equation is non-negative, the objective function is of the maximization or minimization type, and all the decision variables are non-negative.

Another key fact to know before we dive into the algorithm of Simplex Method is converting inequalities into equations. Constraints with $\leq$ are converted into equations by adding a *slack variable* to each constraint. A slack variable represents the quantity of unused resource. Constraints with $\geq$ are converted into equations by subtracting a *surplus variable* from the left-hand side of the inequality. We now consider the initial Simplex table which takes the following form.

Basic Variables	Coefficients of							
Basis	x_1	x_2	x_3	..	s_1	s_2	..	Solution
Z								

Table A.1: Initial Simplex table

Before we look at the Simplex Method it is useful to be aware of the special cases that the method can encounter. *Degeneracy* (Degenerate Solutions): In any situation, if there is a tie for the smallest ratio, one can be selected arbitrarily. When this tie happens at least one basic variable will be zero in the next iteration and the new solution is said to be degenerate. The zero value of the variable can lead to a rise to a series of solutions that have the same Z value. This is referred to as cycling or looping. *Alternative Optima* (more than one optimal solution): This occurs when the slope of any objective constraint and the objective function line are the same. In the optimal Simplex table, if the coefficient of at least one non-basic variable is equal to zero, the problem has alternative optimal solution. *Unbounded solution*: If all coefficients of the variables to be entered into the basis have zero or negative values at any iteration the problem has an unbounded solution. *Infeasible solution*: If the set of basic variables contains at least one artificial variable in the optimal Simplex table, then the given problem has no feasible solution. We may now proceed to the algorithm.

Algorithm A.3 Simplex Method

1. Convert the model into the standard form.

   Since the slack variables are unused resources, they contribute nothing to the profit Of the objective function. So the objective function coefficient of the slack variable is zero. Also, the Simplex Method requires that any variable that appear in one equation must appear in all the equations. This can be done by proper placement of a zero coefficient.

2. Determine an initial basic feasible solution.

   An initial feasible solution is obtained by setting decision variables to zero. The variables that have positive values are called **basic variables** in the solution. The set of basic variables is called as the **basis** . The remaining variables are called **non-basic variables**.

3. Construct the initial Simplex table.

4. Select an **entering variable** using the optimality condition. Stop if there is no entering variable; last solution is optimal. This is called as the key column **pivot column**. **Optimality condition**: the entering variable in a maximization (minimization) problem is the non-basic variable having the most negative (positive) coefficient in the Z-row. Ties are broken arbitrarily.

   Theorem A.2 (Optimality in Simplex Method)**.**
   For a maximization (minimization) problem the optimum is reached at the iteration where all the z-row coefficients of the non-basic variables are non-negative (non-positive).

5. Select a **leaving variable** using the feasibility condition.

   Feasibility condition: for the maximization and minimization problems, the leaving variable the basic variable associated with the smallest non-negative ratio. Ties a broken arbitrarily. This row is called as the key row or **pivot row** . To calculate ratios, divide each value in the solution column by its corresponding pivot column value. The value at the intersection of the key column and key row is called key element of **pivot element** .

6. Compute the new basic solution by using the appropriate Gauss-Jordan computations. The Gauss Jordan computations to produce the new basic solution include:

   a. new pivot row = current pivot row / pivot element

   b. for all other rows including Z,

      new row= current row- (its pivot column coefficient $\times$ new pivot row)

A.3.1 Artificial Variables Technique

LPs in which all the constraints are $\leq$ with non-negative right hand side values offer a convenient all slack starting basic feasible solution. Models that involve = and / $\geq$ or constraints do not possess this property. In such cases identity Matrix cannot be obtained in the starting Simplex table. Therefore a new type of variable called **artificial variable** is introduced. These variables cannot have any physical meaning. They play the role of slacks at the first iteration and then Simplex procedure may be adopted as usual until the optimal solution is obtained. To solve such LP there are two methods; M-method (big M-method or method of penalties) and two-phase method.

Algorithm A.4 The Big M-method

1. Express the problem in the standard form.
2. Add non-negative artificial variables to the left-hand side of each of the equations corresponding to constraints of the type $\geq$ or =. The coefficients of the slack and surplus variables in the objective functions are zero as they contribute nothing to the objective function. However, if we add zero coefficient to the artificial variable in the objective function it can be end up in the final solution base as a part of the optimal solution. Therefore, to get rid of these variables and would not allow to appear them in the final solution we can assign a very large penalty (-M for maximization and +M for minimization) in the objective function.

   Tie for the key row When determining the key row it is possible to get the smallest positive ratio for two or more variables be identical. Ties between artificial variable and decision/slack variable, select artificial variable. Ties between decision variable and slack variable, select slack variable. Ties between decision variable and decision variable, select any one.

   Tie for the key column When there is a tie to the key column, selection can be done arbitrarily. However, to minimize the number of iterations, the following rules may be used. If there is a tie in between two or more slack (or decision) variables, the choice can be made arbitrarily. If there is a tie in between decision variable and slack variable, the decision variable is chosen.

Example A.3.

A company produces both interior and exterior paints from two raw materials M1 and M2. The following table provides the basic data of the problem.

	Tons of raw material per ton of		Maximum Daily availability (tons)
	Exterior Paint	Interior Paint	
Raw material M1	6	4	24
Raw material M2	1	2	6
Profit per ton ($1000)	5	4	

Table A.2: Basic data corresponding to Example A.3

A market survey indicates that the daily demand for interior paint cannot exceed that of exterior paint by more than one ton. Also, the maximum daily demand of interior paint is 2 tons. Company wants to determine the optimal (best) product mix of interior and exterior paints that maximizes the total daily profit.

Let x_1, x_2 *be the number of tons of exterior paint and interior paint respectively.*

The corresponding LP:

$$\begin{aligned} \textit{Maximize} \quad & Z = 5x_1 + 4x_2 \\ \textit{such that} \quad & 6x_1 + 4x_2 \leq 24 \\ & x_1 + 2x_2 \leq 6 \\ & -x_1 + x_2 \leq 1 \\ & x_2 \leq 2 \\ & x_1,\, x_2 \geq 0. \end{aligned}$$

Introducing slack variables,

$$\begin{aligned} \textit{Maximize} \quad & Z = 5x_1 + 4x_2 \\ \textit{such that} \quad & 6x_1 + 4x_2 + s_1 = 24 \\ & x_1 + 2x_2 + s_2 = 6 \\ & -x_1 + x_2 + s_3 = 1 \\ & x_2 + s_4 = 2 \\ & x_1,\, x_2,\, s_1,\, s_2,\, s_3,\, s_4 \geq 0 \end{aligned}$$

The initial feasible solution when $x_1 = 0$ *and* $x_2 = 0$ *is:* $\max Z = 0$, $s_1 = 24$, $s_2 = 6$, $s_3 = 1$, *and* $s_4 = 2$. *The basic variables are* s_1, s_2, s_3, s_4 *and non-basic variables are* x_1, x_2.

The above information can be shown in the table form as follows.

Basis	x_1	x_2	s_1	s_2	s_3	s_4	Solution	Ratio
Z	-5	-4	0	0	0	0	0	
s_1	**6**	4	1	0	0	0	24	$24/6 = 4$
s_2	1	2	0	1	0	0	6	$6/1 = 6$
s_3	-1	1	0	0	1	0	1	$1/-1 = -1$
s_4	0	1	0	0	0	1	2	$2/0 = \infty$

Table A.3: Initial Simplex table corresponding to Example A.3

We identify column 1 as the pivot column according to item 4 of Algorithm A.3 (that is, the entering variable is x_1). According to item 5 of Algorithm A.3 row 2 can be recognized as the pivot row (that is, the leaving variable is s_1) and pivot element as 6. Then we do the following calculations: $r_2 \to r_2/$pivot element, $r_1 \to r_1 - (-5)r_2$, $r_3 \to r_3 - (1)r_2$, $r_4 \to r_1 - (-1)r_2$, and $r_5 \to r_5 - (0)r_2$ which leads to the second Simplex table corresponding to Example A.3.

	x_1	x_2	s_1	s_2	s_3	s_4	Solution
r_2	6	4	1	0	0	0	24
$r_2 \to r_2/$*pivot element*	1	2/3	1/6	0	0	0	4
r_1	-5	-4	0	0	0	0	0
$r_1 \to r_1 - (-5)r_2$	0	-2/3	5/6	0	0	0	20
r_3	1	2	0	1	0	0	6
$r_3 \to r_3 - (1)r_2$	0	4/3	-1/6	1	0	0	2
r_4	-1	1	0	0	1	0	1
$r_4 \to r_1 - (-1)r_2$	0	5/3	1/6	0	1	0	5
r_5	0	1	0	0	0	1	2
$r_5 \to r_5 - (0)r_2$	0	1	0	0	0	1	2

Table A.4: Calculations leading to the second Simplex table corresponding to Example A.3

Basis	x_1	x_2	s_1	s_2	s_3	s_4	Solution	Ratio
Z	0	-2/3	5/6	0	0	0	20	30
x_1	1	2/3	1/6	0	0	0	4	6
s_2	0	**4/3**	-1/6	1	0	0	2	3/2
s_3	0	5/3	1/6	0	1	0	5	3
s_4	0	1	0	0	0	1	2	2

Table A.5: Second Simplex table corresponding to Example A.3

Again, we identify pivot column (entering variable), pivot row (leaving variable) and pivot element. Then do the necessary calculations to get the following table.

Basis	x_1	x_2	s_1	s_2	s_3	s_4	Solution	Ratio
Z	0	0	3/4	1/2	0	0	21	
x_1	1	0	1/4	-1/2	0	0	3	
x_2	0	1	-1/8	3/4	0	0	3/2	
s_3	0	0	3/8	-5/4	1	0	5/2	
s_4	0	0	-1/8	-3/4	0	1	1/2	

Table A.6: Final optimal Simplex table corresponding to Example A.3

This is the optimal table and the solution is: $Z_{max} = 21$, $x_1 = 3$, *and* $x_2 = 3/2$. *That is we have to produce 3 tons from exterior paint and 1.5 tons from interior paint to get the maximum daily profit $ 21,000.*

There are many applications of Linear Programming that do not accept fractional solutions. If a Linear Programming model necessitates integer solutions it is normal to adopt the techniques learned beforehand to solve the problem over the real numbers, then round the solution. Unfortunately, this does not always produce a feasible solution. So we employ Integer Linear Programming techniques like: Gomory Cut-Planes, and Dankins Branch and Bound which we discuss in the next section of our Appendix.

Appendix B

Integer Programming

An Integer Programming is nothing but a Linear Programming with the added requirement that all variables be integers. After setting up the LP of the problem and taking account all the constraints there are many software like Excel, MATLAB, etc. that can be used to find the optimal solution and the objective value solution.

A special case of the Integer Programming is the Binary Integer Programming. In binary problem, each variable can only take on the value 0 or 1. This may represent the selection or rejection of an option, yes or no questions, an accepted or failed problem, even or odd problems or many other situations.

B.1 Preliminaries

There are many techniques used to solve Integer Programming problems and the two most common techniques used are called Cutting Plane (CP) method and Branch and Bound (BB) method. The Cutting Plane method consists in adding one or more constraints to the Integer Programming problems to help produce an optimal integer solution. There are no theoretical reasons for choosing between the cut algorithm and Branch and Bound algorithm. BB algorithm consists of a systematic enumeration of all candidates to solution by using upper and lower estimated bounds of the quantity being optimized.

B.2 Cutting Plane Algorithm

The first solution technique we discuss is CP method which we state as follows.

Algorithm B.1 Cutting Plane Algorithm

1. Solve the continuous problem as an LP that is ignore integrality.
2. If by chance the optimal basic variables are all integer then the optimal solution has been found. Otherwise:
3. Generate a cut that is a constraint which is satisfied by all integer solutions to the problem but not by the current LP solution.
4. Add this new constraint and go to 1.

The problem is to define cuts that ensure the convergence of the algorithm in a finite number of steps. The first finite algorithm was devised by R.E. Gomory in 1958.

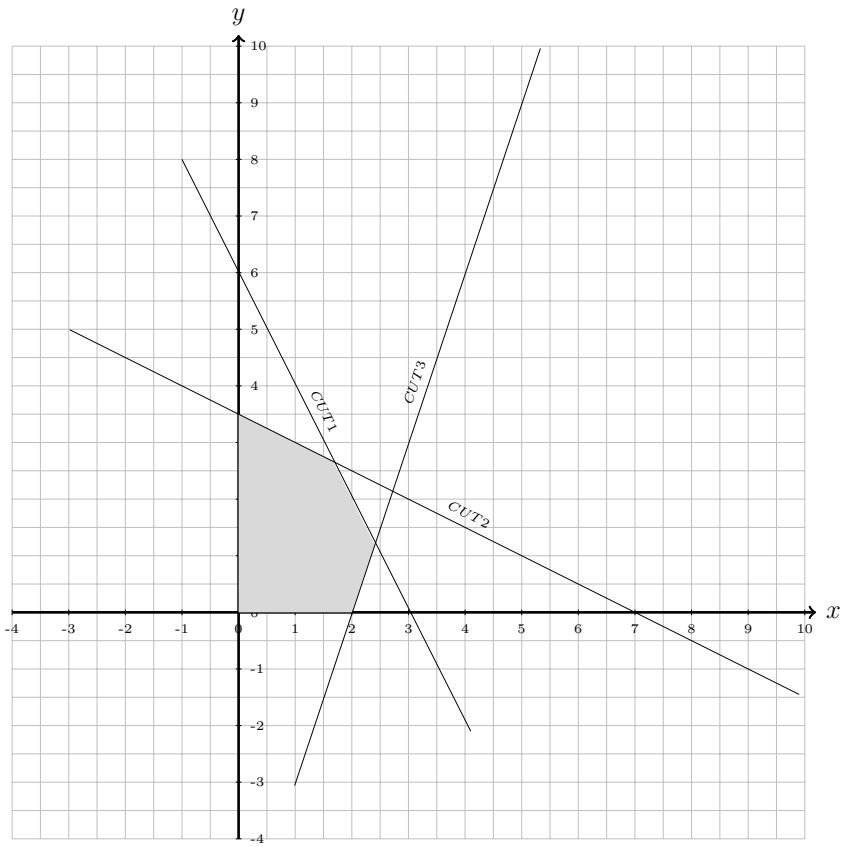

Figure B.1: Gomory cuts

Before introducing the second technique, we consider an example to see how the CP method works.

Example B.1.

Solve the following problem using Cutting Plane Method.

$$\begin{aligned} Minimize \quad & x - y \\ such\ that \quad & 3x + 4y \leq 6 \\ & x - y \leq 1 \\ & x,\, y \in \mathbb{N}_0 \end{aligned}$$

Recall that minimizing $f(x)$ is same as maximizing $-f(x)$.

$$\begin{aligned} Maximize \quad & -(x - y) \\ such\ that \quad & 3x + 4y \leq 6 \\ & x - y \leq 1 \\ & x,\, y \in \mathbb{N}_0 \end{aligned}$$

The feasible region of the current model can be represented as follows.

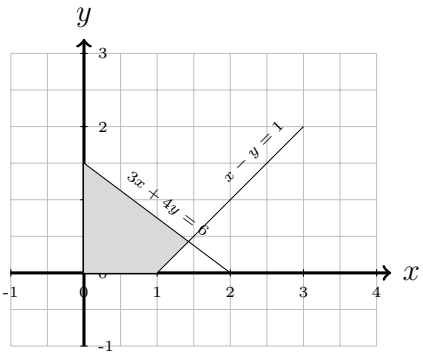

Figure B.2: Initial feasible region corresponding to Example B.1

Introducing slack variables the model can be written as:

$$\begin{aligned} Maximize \quad & P = -x + y \\ such\ that \quad & 3x + 4y + s_1 = 6 \\ & x - y + s_2 = 1 \\ & x,\, y,\, s_1,\, s_2 \in \mathbb{N}_0. \end{aligned}$$

We consider the initial Simplex tableau, with the basic feasible solution: $x = y = 0$ *and* $s_1 = 6$, $s_2 = 1$, $P = 0$.

Basis	x	y	s_1	s_2	Solution
P	1	-1	0	0	0
s_1	3	4	1	0	6
s_2	1	-1	0	1	1

Table B.1: Initial Simplex table corresponding to Example B.1

We can find the final Simplex tableau to be:

Basis	x	y	s_1	s_2	Solution
P	7/4	0	1/4	0	3/2
y	3/4	1	1/4	0	3/2
s_2	7/4	0	1/4	1	5/2

Table B.2: Final optimal Simplex table corresponding to Example B.1

Considering row 2:

$$\frac{3}{4}x + y + \frac{1}{4}s_1 = \frac{3}{2}$$
$$y = \frac{3}{2} - \frac{1}{4}s_1 - \frac{3}{4}x$$
$$y = (1 + \frac{1}{2}) - (0 + \frac{1}{4})s_1 - (0 + \frac{3}{4})x$$
$$y = 1 + (\frac{1}{2} - \frac{1}{4}s_1 - \frac{3}{4}x)$$

Introducing 1st Gomory cut:

$$\frac{1}{2} - \frac{1}{4}s_1 - \frac{3}{4}x \leq 0$$
$$\frac{1}{2} \leq \frac{1}{4}s_1 + \frac{3}{4}x$$

Now by first constraint: $\Rightarrow s_1 = 6 - 3x - 4y$

Thus the new constraint can be written as:

$$\frac{1}{2} \leq \frac{1}{4}(6 - 3x - 4y) + \frac{3}{4}x$$
$$\frac{1}{2} \leq \frac{3}{2} - y$$
$$y \leq 1$$

Thus the LP problem becomes:

$$\begin{aligned} Maximize \quad & P = -x + y \\ such\ that \quad & 3x + 4y \leq 6 \\ & x - y \leq 1 \\ & y \leq 1 \\ & x,\ y,\ s_1,\ s_2 \in \mathbb{N}_0. \end{aligned}$$

The new feasible region of the model can be represented as follows.

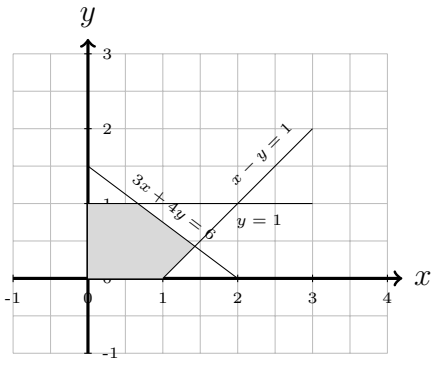

Figure B.3: New feasible region corresponding to Example B.1

Again, introducing slack variables:

$$\begin{aligned} \textit{Maximize} \quad & P = -x + y \\ \textit{such that} \quad & 3x + 4y + s_1 = 6 \\ & x - y + s_2 = 1 \\ & y + s_3 = 1 \\ & x,\, y,\, s_1,\, s_2 \in \mathbb{N}_0. \end{aligned}$$

Then the initial Simplex tableau:

Basis	x	y	s_1	s_2	s_3	Solution
P	1	-1	0	0	0	0
s_1	3	4	1	0	0	6
s_2	1	-1	0	1	0	1
s_3	0	1	0	0	1	1

Table B.3: New initial Simplex table corresponding to Example B.1

We can find the final Simplex tableau to be:

Basis	x	y	s_1	s_2	s_3	Solution
P	1	0	0	0	1	1
s_1	3	0	1	0	-4	2
s_2	1	0	0	1	1	2
y	0	1	0	0	1	1

Table B.4: New final optimal Simplex table corresponding to Example B.1

$\Rightarrow y = 1,\, s_1 = 2,\, s_2 = 2.$

Again considering the first constraint: $x = \frac{6-4y-s_1}{3} = 0.$

Thus $x = 0,\, y = 1$ *maximizes* $P = -x + y$ *and* $P_{max} = 1.$

That is $x = 0,\, y = 1$ *minimizes* $P = x - y$ *and* $P_{min} = -1.$

B.3 Branch and Bound Algorithm

Now we consider our second solution technique which was introduced by R.J. Dankin in 1965. Assume that we are trying to solve the mixed integer problem which we call P_0. The first step is to solve the continuous LP problem obtained by ignoring the integrality constraints. If in the optimal solution, one or more of the integer variables turn out to be non-integer, we choose one such variable and use it to split the given problem P_0 into two sub-problems P_1 and P_2. Suppose the variable chosen is y_j and it takes the non-integral value β_j in the continuous optimum. Then P_1 and P_2 are defined as follows:

$$P_1 = P_0 \quad with \quad the \quad added \quad constraint \quad y_j \leq \lfloor \beta_j \rfloor$$
$$P_2 = P_0 \quad with \quad the \quad added \quad constraint \quad y_j \geq \lfloor \beta_j \rfloor + 1.$$

Now any solution to P_0 is either a solution of P_1 or P_2 and so P_0 can be solved by solving P_1 and P_2. We continue by solving the LP problems associated with P_1 and P_2. Then choose one of the problems and if necessary split it into two subproblems as was done with P_0. The principle of Branch and Bound is illustrated below.

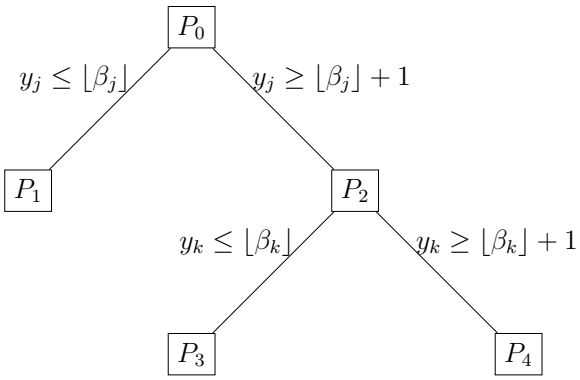

Figure B.4: Branch and Bound algorithm

Before stating the algorithm we now include a remark to our discussion regarding the recent trends in solving Integer Programming problems. Among the methods of solving Integer Programming problems, CP is fast, yet unreliable while BB is reliable but slow. Now

we have a new method called *Branch and Cut* which is in fact a blend of CP method and BB algorithm. This method which mergers the advantages of the two techniques CP and BB to lessen the shortcomings, has recognized to be a very effective method for solving a diversity of Integer Programming problems.

Algorithm B.2 Branch and Bound Algorithm

1. Solve the Linear Programming problem over the real numbers.
2. Identify a solution variable y_j that has a non-integer value β_j.
3. Branch the LP problem into two new problems by introducing into one the constraint $y_j \leq \lfloor \beta_j \rfloor$ and into the other introduce the constraint $y_j \geq \lfloor \beta_j \rfloor + 1$.
4. Solve each branch as an LP problem over the reals. If either has a solution, stop. Else repeat 1 for both branches.

Example B.2.

Solve the following problem by incrementally using BB method.

$$\begin{aligned} Maximize \quad & 4x_1 + 3x_2 + 3x_3 \\ such\ that \quad & 4x_1 + 2x_2 + x_3 \leq 10 \\ & 3x_1 + 4x_2 + 2x_3 \leq 14 \\ & 2x_1 + 1x_2 + 3x_3 \leq 7 \\ & x_1,\, x_2,\, x_3 \in \mathbb{N}_0 \end{aligned}$$

Solving over the reals gives a maximum of $P = 13.8$ *when* $x_1 = 1.2$, $x_2 = 2.2$ *and* $x_3 = 0.8$. *The problem now branches into two LP problems with the first Branch having the additional constraint that* $x_2 \leq 2$ *and the second branch* $x_2 \geq 3$. *Branch 1 yields* $P = 13.6$ *at* $(1.3, 2, 0.8)$ *where Branch 2 has an integer solution* $P = 12$ *at* $(0, 3, 1)$. *Since we may be able to do better than* $P = 12$ *we further split Branch 1 into Branches 3 and 4 by respectively introducing constraints* $x_1 \leq 1$ *and* $x_1 \geq 2$. *Branch 3 yields* $P = 13$ *at* $(1, 2, 1)$ *where Branch 4 gives* $P = 12.2$ *at* $(2, 0.6, 0.8)$. *Branch 3 has an integer solution, but we may be able to do better than its value of 13. Branch 4 further subdivides by introducing the constraints* $x_2 \leq 0$, *in fact* $(x_2 = 0)$ *(Branch 5) and* $x_2 \geq 1$ *(Branch 6). Branch 6 yields another integer*

solution $P = 11$ *at* $(2, 1, 0)$ *which is a smaller value compared to the previous integer value of 13. Branch 5 produces* $P = 11.6$ *at* $(2.3, 0, 0.8)$. *Branch 5 can again be subdivided by introducing the constraint* $x_1 \leq 2$ *(Branch 7) and* $x_1 \geq 3$ *(Branch 8). Branch 7 leads to an integer solution* $P = 11$ *at* $(2, 0, 1)$ *which is again less than the previous integer solution where* $P = 13$. *Branch 8 has an empty feasible region since it clearly violates first constraint. Continuing down any branch would only lead to smaller values for* P, *thus we terminate the process. Taking LP 1 to denote the given problem we can summarize the above as:*

Branch	LP	Solution
	LP 1	$P = 13.8$ at $(1.2, 2.2, 0.8)$
1	LP 1 $x_2 \leq 2$	$P = 13.6$ at $(1.3, 2, 0.8)$
2	LP 1 $x_2 \geq 3$	$P = 12$ at $(0, 3, 1)$
3	LP 1 $x_2 \leq 2,\ x_1 \leq 1$	$P = 13$ at $(1, 2, 1)$
4	LP 1 $x_2 \leq 2,\ x_1 \geq 2$	$P = 12.2$ at $(2, 0.6, 0.8)$
5	LP 1 $x_2 \leq 2,\ x_1 \geq 2,\ x_2 = 0$	$P = 11.6$ at $(2.3, 0, 0.8)$
6	LP 1 $x_2 \leq 2,\ x_1 \geq 2,\ x_2 \geq 1$	$P = 11$ at $(2, 1, 0)$
7	LP 1 $x_2 \leq 2,\ x_1 \geq 2,\ x_2 = 0,\ x_1 \leq 2$	$P = 11$ at $(2, 0, 1)$
8	LP 1 $x_2 \leq 2,\ x_1 \geq 2,\ x_2 = 0,\ x_1 \geq 3$	not feasible

Table B.5: Branches of Branch and Bound method corresponding to Example B.2

Further, the process tree for the above example can be illustrated as below.

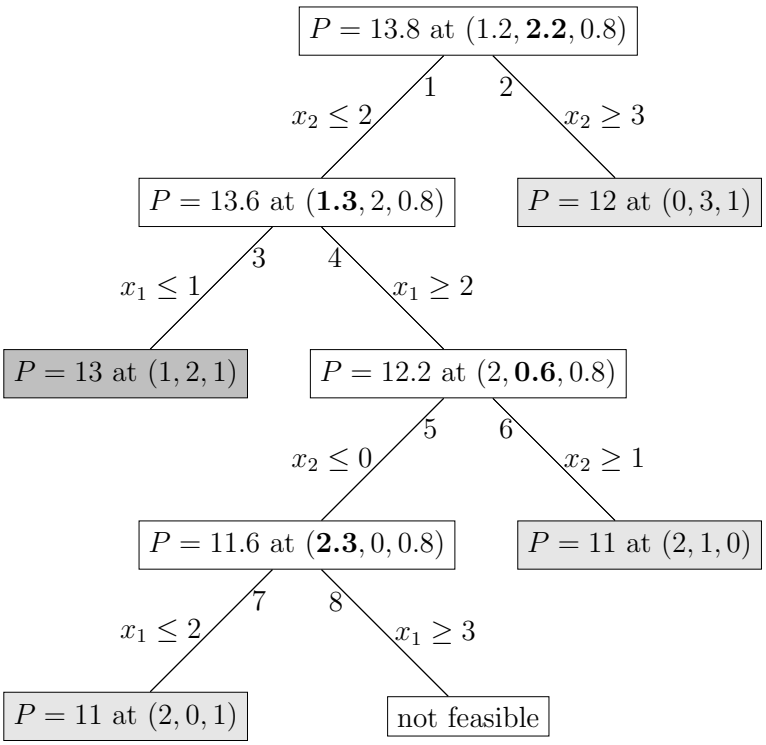

Figure B.5: Process tree corresponding to Example B.2

Thus, $x_1 = 1$, $x_2 = 2$, $x_3 = 1$ maximizes P and $P_{max} = 13$ units.

Bibliography

[1] Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Computational Management Science*, 2(1):3–19, 2005.

[2] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[3] Michael J Best and Klaus Ritter. *Linear programming*. Prentice Hall Upper Saddle River, NJ, 1985.

[4] John R Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations research*, 33(5):989–1007, 1985.

[5] Merve Bodur, Sanjeeb Dash, Oktay Günlük, and James Luedtke. Strengthened benders cuts for stochastic integer programs with continuous recourse. *INFORMS Journal on Computing*, 29(1):77–91, 2016.

[6] Uffe Gram Christensen and Anders Bjerg Pedersen. Lecture note on benders decomposition, 2008.

[7] Gianni Codato and Matteo Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[8] Jean-François Cordeau, Goran Stojković, François Soumis, and Jacques Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation science*, 35(4):375–388, 2001.

[9] Alysson M Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & operations research*, 32(6):1429–1450, 2005.

[10] Jean-François Côté, Mauro Dell'Amico, and Manuel Iori. Combinatorial benders' cuts for the strip packing problem. *Operations Research*, 62(3):643–661, 2014.

[11] Teodor Gabriel Crainic, Mike Hewitt, and Walter Rei. *Partial decomposition strategies for two-stage stochastic integer programs*. CIRRELT, 2014.

[12] Bernard Gendron, Maria Grazia Scutellà, Rosario G Garroppo, Gianfranco Nencioni, and Luca Tavanti. A branch-and-benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research*, 255(1):151–162, 2016.

[13] Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

[14] Arthur M Geoffrion and Glenn W Graves. Multicommodity distribution system design by benders decomposition. *Management science*, 20(5):822–844, 1974.

[15] A Grothey, S Leyffer, and KIM McKinnon. A note on feasibility in benders decomposition. *Numerical Analysis Report NA/188, Dundee University*, 1999.

[16] John Hooker. *Logic-based methods for optimization: combining optimization and constraint satisfaction*, volume 2. John Wiley & Sons, 2011.

[17] John N Hooker. Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55(3):588–602, 2007.

[18] John N Hooker and Greger Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[19] Yuping Huang and Qipeng Phil Zheng. Benders decomposition, 2012.

[20] IBM Corporation. Benders algorithm in cplex v12.7.0. `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/ReleaseNotes/topics/releasenotes127/newBenders.html`, 2016. [Online; accessed 21-July-2019].

[21] Joris Kinable and Michael Trick. A logic based benders approach to the concrete delivery problem. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 176–192. Springer, 2014.

[22] Curtiss Luong. *An Examination of Benders' Decomposition Approaches in Large-scale Healthcare Optimization Problems*. PhD thesis, 2015.

[23] Richard Kipp Martin. *Large scale linear and integer optimization: a unified approach*. Springer Science & Business Media, 2012.

[24] Joe Naoum-Sawaya and Samir Elhedhli. A nested benders decomposition approach for telecommunication network planning. *Naval Research Logistics (NRL)*, 57(6):519–539, 2010.

[25] Manfred Padberg. Classical cuts for mixed-integer programming and branch-and-cut. *Mathematical Methods of Operations Research*, 53(2):173–203, 2001.

[26] Benjamin Peterson and Michael A Trick. A benders approach to a transportation network design problem. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 326–327. Springer, 2009.

[27] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[28] Georgios KD Saharidis and Antonios Fragkogios. Open problems on benders decomposition algorithm. In *Open Problems in Optimization and Data Analysis*, pages 305–317. Springer, 2018.

[29] NV Sahinidis and Ignacio E Grossmann. Convergence properties of generalized benders decomposition. *Computers & Chemical Engineering*, 15(7):481–491, 1991.

[30] M Shahidehopour and Yong Fu. Benders decomposition: applying benders decomposition to power systems. *IEEE Power and Energy Magazine*, 3(2):20–21, 2005.

[31] Wilfredo S Sifuentes and Alberto Vargas. Hydrothermal scheduling using benders decomposition: accelerating techniques. *IEEE Transactions on Power Systems*, 22(3):1351–1359, 2007.

[32] Z Caner Taşkın, J Cole Smith, and H Edwin Romeijn. Mixed-integer programming techniques for decomposing imrt fluence maps using rectangular apertures. *Annals of Operations Research*, 196(1):799–818, 2012.

[33] Zeki Caner Taşkin. Benders decomposition. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[34] Richard M Van Slyke and Roger Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

[35] Jannes Verstichel, Joris Kinable, Patrick De Causmaecker, and G Vanden Berghe. A combinatorial benders decomposition for the lock scheduling problem. *Computers & Operations Research*, 54:117–128, 2015.

[36] Jeffrey Paul Wheeler. *An Introduction to Optimization with Applications in Data Analytics and Machine Learning*. CRC press, 2020.

[37] Wikipedia contributors. Jacques f. benders — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Jacques_F._Benders&oldid=832059510`, 2018. [Online; accessed 13-June-2019].

[38] Christian Wolf. *Advanced acceleration techniques for nested Benders decomposition in stochastic programming*. PhD thesis, Universitätsbibliothek, 2014.

[39] Mohammad Mehdi Fazel Zarandi. *Using decomposition to solve facility location/fleet management problems*. University of Toronto, 2010.

Index