Variants of the simplex algorithm in linear programming problems with a special structure

Carmen Recio Valcarce

Trabajo de fin de grado en Matemáticas
Universidad de Zaragoza

Directora del trabajo: Herminia I. Calvete Fernández
12 de julio de 2017

Prologue

Operations Research is the discipline that deals with the application of advanced analytical methods to help make better decisions.

Mathematical models are used in order to represent real situations abstractly. Among these models we find optimization models, which are those that try to identify the optimal values of the decision variables for an objective function in the set of all the values of the variables that satisfy a set of given conditions.

Based on the functions and variables properties we may classify the optimization problem as a linear, nonlinear, quadratic, integer, etc, optimization problem.

Linear optimization problems satisfy:

• The objective function is a linear function of the decision variables.

• Every restriction can be written as a linear equation or inequation.

The simplex algorithm gives us a systematic method for solving feasible linear optimization problems. This algorithm has become very important due to the several areas of application, its extraordinary efficiency, and because not only does it solve problems numerically, but it also allows us to learn about the use of sensitivity analysis and duality theory.

In this project we will describe some special implementations and variants of the simplex algorithm. These variants try to enhance some aspects such as computational cost or numerical stability. They are also used for solving large linear programming problems and integer optimization problems. Specifically, we will explain the revised simplex method, the case of problems with bounded variables, the decomposition principle and the column generation method. Finally, we will discuss some recent applications of these methods and solve a numerical example of each method. We also include a general review of the applications of these methods nowadays.

To this end, we have studied the books *Linear Programming and Network Flows* by M.S. Bazaraa, J.J. Jarvis and H.D. Sherali, *Introduction to Linear Optimization* by D. Bertsimas and J.N. Tsitsiklis and *Linear and Combinatorial Programming* by K.G. Murty, as well as some research papers.

Summary

El algoritmo simplex es un algoritmo fundamental para resolver problemas de programación lineal. Está basado en recorrer soluciones factibles básicas (puntos extremos) adyacentes hasta que se verifica la condición de optimalidad sobre los costes marginales o se detecta que existe una dirección extrema para la que se cumple la condición de no acotación del problema. En este trabajo se estudian algunas implementaciones especiales y variantes del algoritmo simplex. Estas variantes tratan de mejorar algunos aspectos como el coste computacional o la estabilidad numérica. Se utilizan para resolver problemas de gran dimensión y problemas de optimización entera.

En el primer capítulo se explica el método simplex revisado, una forma sistemática de implementar los pasos del algoritmo simplex que permite ahorrar espacio de almacenamiento al utilizar una tabla de menor tamaño. El algoritmo simplex revisado almacena únicamente la información indispensable para continuar con la siguiente iteración. El algoritmo simplex calcula todos los costes marginales de las variables no básicas para determinar la variable que debe entrar en la base así como todas las columnas asociadas a las mismas; en cambio el algoritmo simplex revisado sólo calcula la columna asociada a la variable no básica que se ha seleccionado para entrar en la base y, una vez determinado el índice de la variable básica que deja la base, esta columna es eliminada y se repite el proceso para la nueva tabla. En el caso del algoritmo simplex, en cada iteración es necesario calcular la inversa de la base; esto puede hacerse de una forma mucho más eficiente si se utilizan operaciones con matrices elementales. Para actualizar la tabla del algoritmo es conveniente utilizar la forma producto de la inversa de una matriz, pues agiliza los cálculos que requiere el pivotaje. Sin embargo en problemas de gran dimensión y problemas sparse (es decir, con pocos coeficientes distintos de cero), es mejor utilizar la factorización LU debido a la mayor precisión y estabilidad numérica de la estrategia de triangulación Gaussiana en la que se basa.

En el segundo capítulo se estudia el método simplex para problemas con variables acotadas. En problemas de este tipo el método más directo para tratar las restricciones sobre la acotación de las variables sería introducir variables de holgura. Sin embargo, esto supondría incrementar en gran medida el número de variables, por lo que no es un método eficiente. El método simplex para variables acotadas trata estas restricciones sin aumentar la dimensión del problema, de forma análoga a la manera en que el método simplex trata las restricciones de no negatividad.

En el tercer capítulo se estudia el principio de descomposición lineal. Éste se refiere a la aplicación de las técnicas de relajación Lagrangiana, de Dantzig-Wolfe o de Benders, que son equivalentes entre sí para problemas de programación lineal. En el caso de problemas de gran dimensión, el objetivo es estructurarlos en uno o más problemas de menor tamaño. Una vez conseguido que el tamaño sea asequible, si algunas de las restricciones poseen una estructura especial, se separa el problema en uno con estructura general y otro con estructura especial de manera que en el segundo se pueda aplicar un método más eficiente. Las restricciones de problemas lineales se dividen en dos tipos: restricciones generales y restricciones con estructura especial. La estrategia del principio de descomposición es trabajar con dos problemas lineales por separado: el problema principal, que tiene las restricciones generales, y el subproblema, que tiene las restricciones con estructura especial. El problema principal transfiere un nuevo conjunto de coeficientes de la función objetivo al subproblema y recibe del subproblema la información sobre la variable que debe entrar en la base en forma de una nueva columna basada en estos coeficientes. Si planteamos el problema en términos de los puntos extremos y las direcciones extremas, como se verá en más detalle en este capítulo, la idea subyacente del principio de descomposición es

encontrar una solución óptima sin necesidad de calcular explícitamente todos los puntos extremos y todas las direcciones extremas.

Una versión especial del principio de descomposición es aquella que se aplica a problemas con una cierta estructura llamada diagonal por bloques o angular. En este tipo de problemas hay un conjunto de restricciones que afectan a todas las variables mientras que otros conjuntos de restricciones sólo afectan a algunos bloques de variables. En el capítulo se explica en detalle este caso. Además, se estudia la descomposición de Benders para problemas enteros mixtos; la estrategia en este caso es descomponer el problema en la parte continua y la parte entera.

En este capítulo también se da una idea general del método de generación de columnas. Este método se suele utilizar en problemas con un número muy grande de variables. Mientras que el método simplex identifica la variable que entra en la base calculando todos los costes marginales, este procedimiento permite la identificación de la variable sin necesidad de calcular todos ellos. Existen muchas variantes de esta técnica general (la descomposición de Dantzig-Wolfe entre ellas). En el capítulo se explica con especial atención una variante que involucra columnas retenidas. En dicha variante del método, el algoritmo almacena en la memoria todas o algunas de las columnas que han sido generadas en el pasado y procede en términos de problemas lineales restringidos que involucran únicamente las columnas almacenadas (retenidas).

El problema de corte de piezas es un ejemplo clásico de la aplicación de la técnica de generación de columnas. En el capítulo se plantea y se desarrolla este problema y en el apéndice 1 se resuelve un ejemplo numérico del mismo.

En el cuarto capítulo se estudian dos artículos recientes en los que se aplican métodos de generación de columnas. El primero está relacionado con la determinación de rutas escolares de autobuses y el segundo con la determinación del lugar óptimo para la construcción de un número fijo de clínicas en África. Ambos problemas son problemas enteros con un número muy grande de variables por lo que se propone el uso del método de generación de columnas para su resolución.

En el apéndice 1 se resuelven ejemplos numéricos de los métodos descritos en los capítulos de este trabajo, con el fin de facilitar la comprensión de los mismos. Estos problemas no están resueltos en la literatura estudiada. El primer ejemplo es un problema de programación lineal de máximo en el que se plantea su resolución mediante el uso de la forma producto de la inversa y el método simplex revisado. El segundo ejemplo es un problema de programación lineal de máximo con variables acotadas en el que se utiliza el método simplex para variables acotadas. En el tercer ejemplo se resuelve un problema lineal de máximo mediante el principio de descomposición para un caso en que las regiones son acotadas. Por último se resuelve un problema de corte de piezas utilizando el método de generación de columnas.

El trabajo acaba con el apéndice 2 en el cual se realiza una revisión más general de las aplicaciones de estos métodos en el mundo real, dando ejemplos y analizando su presencia en los trabajos de investigación actuales.

Contents

Chapter 1

The revised simplex method

1.1 Linear optimization problem statement and basic definitions

Consider the following optimization problem:

$$\begin{array}{ll} \text{Maximize} & f(\mathbf{x}) \\ \text{subject to:} & \\ & g_i(\mathbf{x}) \leq 0, \quad \text{for } i = 1,\ldots,m \\ & h_i(\mathbf{x}) = 0, \quad \text{for } i = 1,\ldots,l \\ & \mathbf{x} \in X \end{array}$$

where $f, g_1, \ldots, g_m, h_1, \ldots, h_l$ are functions defined on $\mathbb{R}^n$, X is a subset of $\mathbb{R}^n$, and $\mathbf{x}$ is a vector of n components $x_1, \ldots, x_n$.

The function f is called the *objective function*. Each of the constraints $g_i(\mathbf{x}) \leq 0$, for $i = 1,\ldots,m$, is called an *inequality constraint*, and each of the constraints $h_i(\mathbf{x}) = 0$, for $i = 1,\ldots,l$, is called an *equality constraint*. The set X might include lower and upper bounds on the variables which can play a useful role in some algorithms. Alternatively, this set might represent some specially structured constraints that can be exploited by the optimization routine, or it might represent complicating constraints that are to be handled separately via a special mechanism.

A vector $\mathbf{x} \in X$ satisfying all the constraints is called a *feasible solution*. The set of all such solutions forms the *feasible region*.

A feasible point $\bar{\mathbf{x}}$ such that $f(\mathbf{x}) \leq f(\bar{\mathbf{x}})$, $\forall$ feasible point $\mathbf{x}$, is called *optimal solution*.

In the special case when the objective function is linear and when all the constraints, including the set X, can be represented by linear inequalities and/or linear equations, the above problem is called a *linear program*.

1.1.1 Standard format

Any given linear program can be put in different equivalent forms by suitable manipulations. A linear program is said to be in *standard format* if all restrictions are equalities and all variables are nonnegative. The simplex method, which is a well-known algorithm for solving linear problems, is designed to be applied only after the problem is put in standard form. How to convert a linear program to standard form is shown in [2].

Therefore, any linear program can be written as:

$$\begin{array}{ll} \text{Maximize} & \sum_{j=1}^{n} c_j x_j \\ \text{subject to:} & \\ & \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad \text{for } i = 1,\ldots,m \\ & x_j \geq 0, \qquad\quad \text{for } j = 1,\ldots,n \end{array}$$

The coefficients $c_1,\ldots,c_n$ are called the *cost coefficients* and $x_1,\ldots,x_n$ are the *decision variables*. The coefficients a_{ij} for $i=1,\ldots,m$ $j=1,\ldots,n$ are called the *technological coefficients* and they form the *constraint matrix* **A**.

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

We assume that $rank(\boldsymbol{A})=m$. The column vector whose ith component is b_i is referred to as the *right-hand-side vector* and we denote it by **b**. Let $\mathbf{x}=(x_1,\ldots,x_n)^t$ and $\mathbf{c}=(c_1,\ldots,c_n)$.

Using matrix notation, the linear programming problem in standard form can be written as:

$$\begin{array}{ll} \text{Maximize} & \mathbf{cx} \\ \text{subject to:} & \\ & \mathbf{Ax}=\mathbf{b} \\ & \mathbf{x}\geq 0 \end{array} \tag{1.1}$$

1.1.2 Duality in linear programming

When solving a linear program, there is another associated linear program that we happen to be simultaneously solving. The original linear program is called the *primal* linear programming problem, and the related linear program is called the *dual* linear programming problem. Actually, the terms primal and dual for this related pair of linear programs are relative since the dual of the dual is the primal itself. The dual problem may be used to obtain the solution to the primal problem and its variables provide information about the original linear program.

Given the primal linear program (1.1), the dual linear program is defined as:

$$\begin{array}{ll} \text{Minimize} & \mathbf{Wb} \\ \text{subject to:} & \\ & \mathbf{WA}\geq\mathbf{c} \\ & \mathbf{W} \text{ unrestricted} \end{array}$$

where **W** is a $1\times m$ vector of dual variables.

1.1.3 Basic feasible solution

Definition. Consider the system $\mathbf{Ax}=\mathbf{b}$ and $\mathbf{x}\geq 0$, where **A** is an $m\times n$ matrix and **b** is an m vector. Suppose that $rank(\mathbf{A},\mathbf{b})=rank(\mathbf{A})=m$. After possibly rearranging the columns of **A**, let **A=[B,N]** where **B** is an $m\times m$ matrix with $rank(\boldsymbol{B})=m$, and **N** is an $m\times(n-m)$ matrix. The solution $\mathbf{x}=\begin{pmatrix}\mathbf{x_B}\\ \mathbf{x_N}\end{pmatrix}$ to the equations $\mathbf{Ax}=\mathbf{b}$, where $\mathbf{x_B}=\mathbf{B}^{-1}\mathbf{b}$ and $\mathbf{x_N}=\mathbf{0}$ is called a *basic solution* of the system. If $\mathbf{x_B}\geq 0$, then **x** is called a *basic feasible solution* (bfs) of the system. The components of $\mathbf{x_B}$ are called *basic variables* and the components of $\mathbf{x_N}$ are called *nonbasic variables*

Definition. Two bfs are said to be *adjacent* if their sets of basic variables have $m-1$ common variables. A bfs is said to be *non-degenerate* if it has exactly m positive components.

We may also write $\mathbf{c}=\begin{pmatrix}\mathbf{c_B} & \mathbf{c_N}\end{pmatrix}$ where $\mathbf{c_B}$ is an $1\times m$ vector whose components are the coefficients of the objective function that correspond to the basic variables and $\mathbf{c_N}$ is an $1\times(n-m)$ vector whose components are the coefficients of the objective function that correspond to the nonbasic variables.

1.1.4 Optimality conditions for linear programming problems

Theorem 1. Let the linear programming problem in standard format (1.1). We assume that the feasible region is nonempty. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ be the extreme points and $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_h$ be the extreme directions of the feasible region. Then the problem has an optimal solution if and only if

$$\mathbf{c}\mathbf{d}_i \leq 0,\ i = 1, \ldots, h.$$

Moreover, if the problem has an optimal solution, then there exists an extreme point which is an optimal solution to the problem.

Further details on the caracterization of extreme points and extreme directions of the feasible region of a linear program can be found in [1, 2].

1.1.5 The simplex algorithm

The simplex algorithm was developed by G.B. Dantzig in 1947. Using Theorem 1, the algorithm moves from one extreme point of the feasible region to an adjacent one until either identifying the optimal or concluding the problem has no optimal solution.

Given the standard formulation of the linear optimization problem (1.1), the steps of the algorithm are as follows:

Steps of the simplex algorithm

1. Let $\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_\mathbf{B} \\ \hat{\mathbf{x}}_\mathbf{N} \end{pmatrix}$ be a bfs with basis $\mathbf{B}$, and basis inverse $\mathbf{B}^{-1}$.

2. The value of the objective function in $\hat{\mathbf{x}}$ is

$$\mathbf{c}\hat{\mathbf{x}} = \begin{pmatrix} \mathbf{c}_\mathbf{B} & \mathbf{c}_\mathbf{N} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_\mathbf{B} \\ \hat{\mathbf{x}}_\mathbf{N} \end{pmatrix} = \begin{pmatrix} \mathbf{c}_\mathbf{B} & \mathbf{c}_\mathbf{N} \end{pmatrix} \begin{pmatrix} \mathbf{B}^{-1}\mathbf{b} \\ \mathbf{0} \end{pmatrix} = \mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{b}$$

3. Is $\hat{x}$ an optimal solution, i.e., $\mathbf{c}\mathbf{x} \leq \mathbf{c}\hat{\mathbf{x}}\ \forall\, \mathbf{x} \in S = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$?

   Let $\mathbf{x} \in S$. Since $\mathbf{x}$ is a feasible solution $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.

   If we write **A=[B,N]** then

$$\mathbf{b} = \mathbf{A}\mathbf{x} = [\mathbf{B}, \mathbf{N}] \begin{pmatrix} \mathbf{x}_\mathbf{B} \\ \mathbf{x}_\mathbf{N} \end{pmatrix} = \mathbf{B}\mathbf{x}_\mathbf{B} + \mathbf{N}\mathbf{x}_\mathbf{N} \Rightarrow \mathbf{x}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_\mathbf{N}$$

   The value of the objective function in $\mathbf{x}$ is

$$\mathbf{c}\mathbf{x} = \begin{pmatrix} \mathbf{c}_\mathbf{B} & \mathbf{c}_\mathbf{N} \end{pmatrix} \begin{pmatrix} \mathbf{x}_\mathbf{B} \\ \mathbf{x}_\mathbf{N} \end{pmatrix} = \mathbf{c}_\mathbf{B}\mathbf{x}_\mathbf{B} + \mathbf{c}_\mathbf{N}\mathbf{x}_\mathbf{N} = \mathbf{c}_\mathbf{B}(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_\mathbf{N}) + \mathbf{c}_\mathbf{N}\mathbf{x}_\mathbf{N} =$$
$$\mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_\mathbf{N} - \mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{N})\mathbf{x}_\mathbf{N} = \mathbf{c}\hat{\mathbf{x}} + (\mathbf{c}_\mathbf{N} - \mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{N})\mathbf{x}_\mathbf{N}$$

   If the vector $\mathbf{c}_\mathbf{N} - \mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{N} \leq \mathbf{0}$ then $(\mathbf{c}_\mathbf{N} - \mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{N})\mathbf{x}_\mathbf{N} \leq \mathbf{0}$. Therefore $\mathbf{c}\mathbf{x} \leq \mathbf{c}\hat{\mathbf{x}}\ \forall\, \mathbf{x} \in S$, that is, $\hat{\mathbf{x}}$ is an optimal solution and we stop.

4. If the vector $\mathbf{c}_\mathbf{N} - \mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{N} \not\leq \mathbf{0}$, then the aim of the algorithm is either to identify if the optimal solution value is unbounded or to find a better (or at least not worse) adjacent bfs.

   Suppose that $c_k - \mathbf{c}_\mathbf{B}\mathbf{B}^{-1}\mathbf{A}_k > 0$. Let $\mathbf{Y}_k = \begin{pmatrix} y_{1k} \\ \ldots \\ y_{rk} \\ \ldots \\ y_{mk} \end{pmatrix} = \mathbf{B}^{-1}\mathbf{A}_k$

- CASE 1: $\mathbf{Y}_k \leq \mathbf{0}$
  Using the characterization of extreme directions [2], we have that the nonzero vector $\mathbf{d} = \begin{pmatrix} \mathbf{d_B} \\ \mathbf{d_N} \end{pmatrix} = \begin{pmatrix} -\mathbf{B}^{-1}\mathbf{A}_k \\ \mathbf{e}_k \end{pmatrix}$ where $\mathbf{e}_k$ is an $(n-m)\times 1$ vector containing a 1 in the kth position and zeros in the rest of them, is an extreme direction of the feasible region S.
  Moreover, $\mathbf{cd} = \begin{pmatrix} \mathbf{c_B} & \mathbf{c_N} \end{pmatrix} \begin{pmatrix} \mathbf{d_B} \\ \mathbf{d_N} \end{pmatrix} = \mathbf{c_B}\mathbf{d_B} + \mathbf{c_N}\mathbf{d_N} = \mathbf{c_B}(-\mathbf{B}^{-1}\mathbf{A}_k) + c_k = c_k - \mathbf{c_B}\mathbf{B}^{-1}\mathbf{A}_k > 0$. Using Theorem 1, we conclude that the problem is unbounded. The algorithm stops.
- CASE 2: $\mathbf{Y}_k \not\leq \mathbf{0}$
  We denote $\bar{\mathbf{b}} = \mathbf{B}^{-1}\mathbf{b} = \begin{pmatrix} \bar{b}_1 \\ \cdots \\ \bar{b}_r \\ \cdots \\ \bar{b}_m \end{pmatrix}$. Let $\dfrac{\bar{b}_r}{y_{rk}} = \min\limits_{1\leq i\leq m}\left\{\dfrac{\bar{b}_i}{y_{ik}} : y_{ik} > 0\right\}$ and consider the point:
  $\tilde{\mathbf{x}} = \hat{\mathbf{x}} + \dfrac{\bar{b}_r}{y_{rk}} \begin{pmatrix} -\mathbf{B}^{-1}\mathbf{A}_k \\ \mathbf{e}_k \end{pmatrix}$
  It is easy to check that $\tilde{\mathbf{x}} \geq \mathbf{0}$ and $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b}$. Therefore $\tilde{\mathbf{x}}$ is a feasible solution. Moreover, $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$ are adjacent feasible solutions. The value of the objective function in the new solution $\tilde{\mathbf{x}}$ is
  $$\mathbf{c}\tilde{\mathbf{x}} = \mathbf{c}\hat{\mathbf{x}} + \frac{\bar{b}_r}{y_{rk}} \begin{pmatrix} \mathbf{c_B} & \mathbf{c_N} \end{pmatrix} \begin{pmatrix} -\mathbf{B}^{-1}\mathbf{A}_k \\ \mathbf{e}_k \end{pmatrix} = \mathbf{c}\hat{\mathbf{x}} + \frac{\bar{b}_r}{y_{rk}} \left(c_k - \mathbf{c_B}\mathbf{B}^{-1}\mathbf{A}_k\right) \geq \mathbf{c}\hat{\mathbf{x}}$$
  Hence, $\tilde{\mathbf{x}}$ is a bfs with a better (or at least not worse) value of the objective function. Repeat the process with $\tilde{\mathbf{x}}$.

Since the number of extreme points is finite [1, 2], in the absence of degeneracy either the optimal solution is reached, or else we conclude that the problem is unbounded. In case of degeneracy, some rules for selecting the entering and the leaving variables have been developed in order to guarantee convergence. (See [2, pag 175-187]).

1.2 Revised simplex method

The revised simplex method is a systematic procedure for implementing the steps of the simplex method that enables us to save storage space by using a smaller array. The revised simplex method does not store all the tableau information but merely the inverse of the matrix basis.

If we examine the steps of the simplex algorithm, it is easy to see that for each iteration we need to update the basic variables and compute the matrix $\mathbf{B}^{-1}$. This can be done in a more efficient way by using the elementary operations.

Suppose that we have a bfs with a known $\mathbf{B}^{-1}$. We construct the array shown in Table 1.1, where $\mathbf{w} = \mathbf{c_B}\mathbf{B}^{-1}$ and $\bar{\mathbf{b}} = \mathbf{B}^{-1}\mathbf{b}$. This tableau is called the *revised simplex tableau*. We may assume that we start with $\mathbf{B} = \mathbf{I}$. The idea is to imagine that we are performing the simplex algorithm with the whole simplex tableau, but we are only keeping the information presented in this array.

Assume that we want to analyse the maximization problem, the first thing we need to calculate are the reduced costs, $c_j - z_j$ for each nonbasic variable, where $z_j = \mathbf{c_B}\mathbf{B}^{-1}\mathbf{A}_j$. Since $\mathbf{w}$ is known $c_j - z_j = c_j - \mathbf{w}\mathbf{A}_j$. If $c_j - z_j \leq 0\ \forall j$ then the current bfs is optimal. Otherwise, let k be the index such that $c_k - z_k = \max\limits_j\{c_j - z_j\} > 0$. We have to examine the updated column of x_k. Compute $\mathbf{Y}_k = \mathbf{B}^{-1}\mathbf{A}_k$. If $\mathbf{Y}_k \leq \mathbf{0}$, then the optimal solution value is unbounded and the algorithm stops. If $\mathbf{Y}_k \not\leq \mathbf{0}$ then we append the updated column of x_k to the revised simplex tableau (see Table 1.1).

The r index can now be calculated with the usual minimum ratio test $\dfrac{\bar{b}_r}{y_{rk}} = \min\limits_{1\leq i\leq m}\left\{\dfrac{\bar{b}_i}{y_{ik}} : y_{ik} > 0\right\}$.

BASIS INVERSE	RHS		x_k
$\mathbf{w}$	$\mathbf{c_B}\bar{\mathbf{b}}$		$c_k - z_k$
$\mathbf{B}^{-1}$	$\bar{b}_1$ $\bar{b}_2$ $\vdots$ $\bar{b}_r$ $\vdots$ $\bar{b}_m$		y_{1k} y_{2k} $\vdots$ y_{rk} $\vdots$ y_{mk}

Table 1.1: Revised simplex tableau and column associated with the variable entering the basis

Now, pivoting at y_{rk} in the usual way gives the new values of $\mathbf{w}$, $\mathbf{B}^{-1}$, $\bar{\mathbf{b}}$ and $\mathbf{c_B}\bar{\mathbf{b}}$, and the process is repeated.

In the case of a minimization problem, we need to change the sign of $c_j - z_j$. If $c_k - z_k = \min_j\{c_j - z_j\} > 0$, stop; the current bfs is optimal. Otherwise, we select the variable x_k to enter the basis.

1.3 Comparison of the simplex and the revised simplex methods

Now we are going to analyse the differences between the simplex and the revised simplex methods.

For the revised simplex method an $(m+1)\times(m+1)$ array is needed while for the simplex method we need an $(m+1)\times(n+1)$ array. If $n \gg m$, this would result in a substantial saving in computer core storage.

The following table shows the number of multiplications and additions per iteration of both methods:

METHOD		PIVOTING	$c_j - z_j$'s	TOTAL
Simplex	Multiplications	$(m+1)(n-m+1)$		$m(n-m)+n+1$
	Additions	$m(n-m+1)$		$m(n-m+1)$
Revised Simplex	Multiplications	$(m+1)^2$	$m(n-m)$	$m(n-m)+(m+1)^2$
	Additions	$m(m+1)$	$m(n-m)$	$m(n+1)$

The number of operations required per iteration in the simplex method is slightly less than those required for the revised simplex method. Most practical problems are sparse, that is, the density d of nonzero elements (number of nonzero elements divided by total number of elements) in the constraint matrix is usually small. We can take advantage of this while calculating $c_j - z_j$. Note that $z_j = \mathbf{w}\mathbf{A}_j$ and we can skip zero elements of $\mathbf{A}_j$ when doing the calculation $\mathbf{w}\mathbf{A}_j = \sum_{i=1}^{m} w_i a_{ij}$. Therefore, in the revised simplex method the number of operations for calculating the jth reduced cost is $dm(n-m)$ multiplications and $dm(n-m)$ additions. When pivoting no operations are skipped for any of the methods.

Empirically, it has been observed that the simplex method requires on the order of m to $3m$ iterations. If we examine the table, since the number of operations per iteration is of order $O(mn)$, the average empirical complexity of the simplex method is $O(m^2n)$ (i.e. it is bounded from above by some constant times m^2n).

To sum up, if $n \gg m$ and the density d is small, the computational cost of the revised simplex method is significantly smaller than that of the simplex method. Also the use of the original data for calculating the reduced costs and the updated column Y_k reduces the cumulative round-off error.

1.4 Product form of the inverse of a matrix

We will now examine a way of implementing the revised simplex method where the inverse of the basis is stored as the product of elementary matrices. Remember that an *elementary matrix* is a square matrix that differs from the identity in only one row or one column. For sparse problems, this results in fewer storage and computational burdens, and it provides greater numerical stability by reducing accumulated round-off errors.

Consider a basis $\mathbf{B}$ composed of the columns $\mathbf{A}_{\mathbf{B}_1}$,$\mathbf{A}_{\mathbf{B}_2}$,... $\mathbf{A}_{\mathbf{B}_m}$ and suppose that $\mathbf{B}^{-1}$ is known. Now suppose that the nonbasic column $\mathbf{A}_k$ replaces $\mathbf{A}_{\mathbf{B}_r}$, resulting in a new basis $\hat{\mathbf{B}}$. We wish to find $\hat{\mathbf{B}}^{-1}$ in terms of $\mathbf{B}^{-1}$. Since $\mathbf{A}_k = \mathbf{B}\mathbf{Y}_k$ and $\mathbf{A}_{\mathbf{B}_i} = \mathbf{B}\mathbf{e}_i$ where $\mathbf{e}_i$ is a vector of zeros except for 1 at the ith position, we have

$$\hat{\mathbf{B}} = \left(\mathbf{A}_{\mathbf{B}_1}, \mathbf{A}_{\mathbf{B}_2}, ..., \mathbf{A}_{\mathbf{B}_{r-1}}, \mathbf{A}_k, \mathbf{A}_{\mathbf{B}_{r+1}}, ..., \mathbf{A}_{\mathbf{B}_m}\right) = (\mathbf{B}\mathbf{e}_1, \mathbf{B}\mathbf{e}_2, ..., \mathbf{B}\mathbf{e}_{r-1}, \mathbf{B}\mathbf{Y}_k, \mathbf{B}\mathbf{e}_{r+1}, ..., \mathbf{B}\mathbf{e}_m) = \mathbf{B}\mathbf{T}$$

where $\mathbf{T}$ is the identity with the rth column replaced by Y_k. The inverse of $\mathbf{T}$, which we will denote by $\mathbf{E}$, is given by:

$$\mathbf{E} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{c} \qquad\qquad r\text{th column} \\ \qquad\qquad\quad \downarrow \end{array} \\ \begin{bmatrix} 1 & \cdots & 0 & \frac{-y_{1k}}{y_{rk}} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & \frac{-y_{(r-1)k}}{y_{rk}} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{1}{y_{rk}} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{-y_{(r+1)k}}{y_{rk}} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \frac{-y_{mk}}{y_{rk}} & 0 & \cdots & 1 \end{bmatrix} \end{array} \begin{array}{l} \\ \\ \\ \longleftarrow r\text{th row} \\ \\ \\ \\ \end{array}$$

Therefore $\hat{\mathbf{B}}^{-1} = \mathbf{T}^{-1}\mathbf{B}^{-1} = \mathbf{E}\mathbf{B}^{-1}$ where the elementary matrix $\mathbf{E}$ is known. Only the nonidentity column and its position r need to be stored to specify $\mathbf{E}$.

If the basis $\mathbf{B}_1$ at the first iteration is the identity $\mathbf{I}$. Then the basis inverse $\mathbf{B}_2^{-1}$ at iteration 2 is $\mathbf{B}_2^{-1} = \mathbf{E}_1\mathbf{B}_1^{-1} = \mathbf{E}_1\mathbf{I} = \mathbf{E}_1$ where $\mathbf{E}_1$ is the elementary matrix corresponding to the first iteration. Repeating the same argument we have that $\mathbf{B}_3^{-1} = \mathbf{E}_2\mathbf{B}_2^{-1} = \mathbf{E}_2\mathbf{E}_1$.

In general $\mathbf{B}_t^{-1} = \mathbf{E}_{t-1}\mathbf{E}_{t-2}...\mathbf{E}_2\mathbf{E}_1$. This equation is called *the product form of the inverse*. Using this form, all the steps of the simplex method can be performed without pivoting.

The product inverse method is conceptually important and useful but it is computationally obsolete because it is based on a complete Gauss-Jordan elimination technique for solving systems of equations.

Nowadays most modern computer packages use the *LU factorization method* which is based on the more efficient Gaussian triangularization strategy. This method is most useful for large-scale and sparse problems, and it is accurate and numerically stable (round-off errors are controlled and do not tend to accumulate). The LU factorization of a matrix can be found in [6].

Chapter 2

The simplex algorithm for bounded variables

2.1 Problem statement and basic definitions

In most practical problems the variables are bounded. A variable is said to be *bounded* if $l_j \leq x_j \leq u_j$ where $l_j < u_j$. Let's consider the following linear programming problem with bounded variables:

$$\begin{array}{ll} \text{Minimize} & \mathbf{cx} \\ \text{subject to:} & \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{array} \tag{2.1}$$

where $\mathbf{l}$ and $\mathbf{u}$ are the lower and upper bound vectors, respectively. We assume that $\mathbf{A}$ is a $m \times n$ matrix of rank m.

We can also assume without loss of generality that l_j is finite $\forall j$. If **l=0** then we have the usual nonnegativity restrictions. Observe that any lower bound vector can be transformed into the zero vector by using the change of variables $\mathbf{x}' = \mathbf{x} - \mathbf{l}$, this way we obtain the following problem:

$$\begin{array}{ll} \text{Minimize} & \mathbf{c}(\mathbf{x}' + \mathbf{l}) \\ \text{subject to:} & \\ & \mathbf{Ax}' = \mathbf{b} - \mathbf{Al} \\ & \mathbf{0} \leq \mathbf{x}' \leq \mathbf{u} - \mathbf{l} \end{array}$$

The most straightforward method of handling the constraints $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$ is to introduce the slack vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, leading to the constraints $\mathbf{x} + \mathbf{x}_1 = \mathbf{u}$ and $\mathbf{x} - \mathbf{x}_2 = \mathbf{l}$. Nevertheless, this method is the least efficient since it increases the number of equality constraints from m to $m+2n$ and the number of variables from n to $3n$. Even if **l=0** or it is transformed into $\mathbf{0}$, the slack vector $\mathbf{x}_1$ is needed, which increases both the constraints and variables by n.

The simplex method with bounded variables handles bound constraints without increasing the problem size, that is, in an implicit way, analogously to the way the simplex method deals with the nonnegativity constraints.

Consider the feasible region of problem (2.1). Assume that $\mathbf{A}$ can be partitioned into $[\mathbf{B}, \mathbf{N}_1, \mathbf{N}_2]$ where $\mathbf{B}$ is a square matrix of rank m. A point $\bar{\mathbf{x}}$ is a basic solution if, assuming it is partitioned accordingly as $\bar{\mathbf{x}} = (\bar{\mathbf{x}}_\mathbf{B}, \bar{\mathbf{x}}_{\mathbf{N}_1}, \bar{\mathbf{x}}_{\mathbf{N}_2})$, $\bar{\mathbf{x}}_{\mathbf{N}_1} = \mathbf{l}_{\mathbf{N}_1}$, $\bar{\mathbf{x}}_{\mathbf{N}_2} = \mathbf{u}_{\mathbf{N}_2}$, and $\bar{\mathbf{x}}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}_1\mathbf{l}_{\mathbf{N}_1} - \mathbf{B}^{-1}\mathbf{N}_2\mathbf{u}_{\mathbf{N}_2}$.

Furthermore, if $\mathbf{l}_\mathbf{B} \leq \bar{\mathbf{x}}_\mathbf{B} \leq \mathbf{u}_\mathbf{B}$, then $\bar{\mathbf{x}}$ is said to be a *basic feasible solution* (bfs). The matrix $\mathbf{B}$ is called the *basis*, $\mathbf{x}_\mathbf{B}$ are the *basic variables*, and $\mathbf{x}_{\mathbf{N}_1}$, and $\mathbf{x}_{\mathbf{N}_2}$ are the *nonbasic variables* at their lower and upper limits, respectively.

If, in addition, $\mathbf{l}_\mathbf{B} < \bar{\mathbf{x}}_\mathbf{B} < \mathbf{u}_\mathbf{B}$, then $\bar{\mathbf{x}}$ is called a *nondegenerate basic feasible solution*; otherwise it is called a *degenerate basic feasible solution*.

Definition. The partition $[\mathbf{B},\mathbf{N}_1,\mathbf{N}_2]$ corresponding to a basic (feasible) solution is said to be a *basic (feasible) partition*. Two basic (feasible) partitions are said to be *adjacent* if all but one nonbasic variable coincides in the two partitions being fixed at the same bounds.

2.2 Steps of the simplex algorithm for bounded variables

Suppose that $\mathbf{A} = [\mathbf{B},\mathbf{N}_1,\mathbf{N}_2]$. Accordingly, $\mathbf{x} = [\mathbf{x_B},\mathbf{x}_{\mathbf{N}_1},\mathbf{x}_{\mathbf{N}_2}]$ and $\mathbf{c} = [\mathbf{c_B},\mathbf{c}_{\mathbf{N}_1},\mathbf{c}_{\mathbf{N}_2}]$. Both the basic variables and the objective function can be represented in terms of the independent (that is, nonbasic) vectors $\mathbf{x}_{\mathbf{N}_1}$ and $\mathbf{x}_{\mathbf{N}_2}$ as follows:

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}_1\mathbf{x}_{\mathbf{N}_1} - \mathbf{B}^{-1}\mathbf{N}_2\mathbf{x}_{\mathbf{N}_2} \tag{2.2a}$$

$$\begin{aligned} z = {} & \mathbf{c_B}\mathbf{x_B} + \mathbf{c}_{\mathbf{N}_1}\mathbf{x}_{\mathbf{N}_1} + \mathbf{c}_{\mathbf{N}_2}\mathbf{x}_{\mathbf{N}_2} = \\ & \mathbf{c_B}(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}_1\mathbf{x}_{\mathbf{N}_1} - \mathbf{B}^{-1}\mathbf{N}_2\mathbf{x}_{\mathbf{N}_2}) + \mathbf{c}_{\mathbf{N}_1}\mathbf{x}_{\mathbf{N}_1} + \mathbf{c}_{\mathbf{N}_2}\mathbf{x}_{\mathbf{N}_2} = \\ & \mathbf{c_B}\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_{\mathbf{N}_1} - \mathbf{c_B}\mathbf{B}^{-1}\mathbf{N}_1)\mathbf{x}_{\mathbf{N}_1} + (\mathbf{c}_{\mathbf{N}_2} - \mathbf{c_B}\mathbf{B}^{-1}\mathbf{N}_2)\mathbf{x}_{\mathbf{N}_2} \end{aligned} \tag{2.2b}$$

Suppose that we have a current bfs where $\mathbf{x}_{\mathbf{N}_1} = \mathbf{l}_{\mathbf{N}_1}$, $\mathbf{x}_{\mathbf{N}_2} = \mathbf{u}_{\mathbf{N}_2}$ and $\mathbf{l_B} \le \mathbf{x_B} \le \mathbf{u_B}$. This solution is represented by the following tableau.

	z	$\mathbf{x_B}$	$\mathbf{x}_{\mathbf{N}_1}$	$\mathbf{x}_{\mathbf{N}_2}$	RHS
z	1	$\mathbf{0}$	$\mathbf{c_B}\mathbf{B}^{-1}\mathbf{N}_1 - \mathbf{c}_{\mathbf{N}_1}$	$\mathbf{c_B}\mathbf{B}^{-1}\mathbf{N}_2 - \mathbf{c}_{\mathbf{N}_2}$	$\hat{z}$
$\mathbf{x_B}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{B}^{-1}\mathbf{N}_1$	$\mathbf{B}^{-1}\mathbf{N}_2$	$\hat{b}$

The right-hand-side column gives the *true values* of z and $\mathbf{x_B}$ (denoted by $\hat{z}$ and $\hat{b}$). We can obtain these values from the equations:

$$\begin{aligned} \hat{\mathbf{b}} &= \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}_1\mathbf{l}_{\mathbf{N}_1} - \mathbf{B}^{-1}\mathbf{N}_2\mathbf{u}_{\mathbf{N}_2} \\ \hat{z} &= \mathbf{c_B}\hat{\mathbf{b}} + \mathbf{c}_{\mathbf{N}_1}\mathbf{l}_{\mathbf{N}_1} + \mathbf{c}_{\mathbf{N}_2}\mathbf{u}_{\mathbf{N}_2} \end{aligned}$$

Now we try to find a bfs with a better (or at least not worse) value of the objective function.

From Equation (2.2b) and noting that $\mathbf{c}_{\mathbf{N}_1} - \mathbf{c_B}\mathbf{B}^{-1}\mathbf{N}_1$ and $\mathbf{c}_{\mathbf{N}_2} - \mathbf{c_B}\mathbf{B}^{-1}\mathbf{N}_2$ give the reduced costs $c_j - z_j$ of the lower and upper bounded nonbasic variables, respectively, we get:

$$z = \mathbf{c_B}\mathbf{B}^{-1}\mathbf{b} - \sum_{j\in R_1}(z_j - c_j)x_j - \sum_{j\in R_2}(z_j - c_j)x_j \tag{2.3}$$

where R_1 is the set of indices of nonbasic variables at their lower bounds and R_2 is the set of indices of nonbasic variables at their upper bounds. For $j \in R_1$, if $z_j - c_j > 0$, it would be beneficial to increase x_j from its current value of l_j. Similarly, for $j \in R_2$, if $z_j - c_j < 0$, it would be convenient to decrease x_j from its current value of u_j. As in the simplex method, we should modify the value of only one nonbasic variable x_k.

The index k is determined as follows. First examine

$$\max\left(\max_{j\in R_1}(z_j - c_j), \max_{j\in R_2}(z_j - c_j)\right)$$

If this maximum is positive, then let k be the index for which the maximum is achieved. If it corresponds to R_1, then x_k is increased from its current level of l_k, and if it corresponds to R_2, then x_k is decreased from its current level of u_k.

If this maximum is less than or equal to 0, then $z_j - c_j \le 0$, $\forall j \in R_1$, and $z_j - c_j \ge 0$, $\forall j \in R_2$. Examining Equation (2.3) this implies that the current solution is optimal.

Now we are going to discuss the two cases: when x_k is increased from its current level l_k and when x_k is decreased from its current level u_k. In both cases, we want to determine how much we can either increase or decrease the variable x_k while maintaining the solution feasible.

Case 1: Increasing x_k from its current level l_k

Let $x_k = l_k + \triangle_k$, where $\triangle_k$ is the increase in x_k. Noting that all other nonbasic variables are fixed and that the current value of $\mathbf{x_B}$ and z are respectively $\hat{\mathbf{b}}$ and $\hat{z}$, substituting $x_k = l_k + \triangle_k$ in Equations (2.2a) and (2.3), we get

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}_1\mathbf{l}_{\mathbf{N}_1} - \mathbf{B}^{-1}\mathbf{N}_2\mathbf{u}_{\mathbf{N}_2} - \mathbf{B}^{-1}\mathbf{A}_k \triangle_k = \hat{\mathbf{b}} - \mathbf{Y}_k \triangle_k \tag{2.4}$$

$$z = \mathbf{c_B}\mathbf{B}^{-1}\mathbf{b} - \sum_{j\in R_1}(z_j - c_j)l_j - \sum_{j\in R_2}(z_j - c_j)u_j - (z_k - c_k)\triangle_k = \hat{z} - (z_k - c_k)\triangle_k \tag{2.5}$$

Since $z_k - c_k > 0$, then from Equation (2.5) it is clear that our aim is to increase $\triangle_k$ as much as possible.

If x_k is increased then $\triangle_k$ cannot be greater than $u_k - l_k$ in order to maintain feasibility.

On the other hand, the basic variables are modified according to Equation (2.4), that is, $\mathbf{x_B} = \hat{\mathbf{b}} - \mathbf{Y}_k \triangle_k$. The new values of the basic variables must hold their bounds, therefore the $\triangle_k$ value is also limited by this. Let's discuss the possible cases:

1. x_i **is a basic variable such that** $y_{ik} > 0$

   Since $\triangle_k \geq 0$, the upper-bound constraint is trivially verified. In order to guarantee the lower-bound constraint, the value of $\triangle_k$ must be at most equal to:

   $$\gamma_1 = \min_{1\leq i\leq m}\left\{\frac{\hat{b}_i - l_{B_i}}{y_{ik}} : y_{ik} > 0\right\} = \frac{\hat{b}_r - l_{B_r}}{y_{rk}} \tag{2.6}$$

   If there is not any basic variable x_i such that $y_{ik} > 0$, then $\triangle_k$ could take any nonnegative value without losing feasibility, that is, γ_1 can be made arbitrarily large, $\gamma_1 = \infty$.

2. x_i **is a basic variable such that** $y_{ik} < 0$

   Again, since $\triangle_k \geq 0$, the lower-bound constraint is trivially verified. In order to guarantee the upper-bound constraint, the value of $\triangle_k$ must be at most equal to:

   $$\gamma_2 = \min_{1\leq i\leq m}\left\{\frac{u_{B_i} - \hat{b}_i}{-y_{ik}} : y_{ik} < 0\right\} = \frac{u_{B_r} - \hat{b}_r}{-y_{rk}} \tag{2.7}$$

   If there is not any basic variable x_i such that $y_{ik} < 0$, then $\triangle_k$ could take any nonnegative value without losing feasibility, $\gamma_2 = \infty$.

We have denoted the value of $\triangle_k$ at which a basic variable drops to its lower bound by γ_1 and the value of $\triangle_k$ at which a basic variable reaches its upper bound by γ_2. Observe that the value of $\triangle_k$ at which x_k reaches its upper bound u_k is obviously $u_k - l_k$. These three cases give the maximum increase in $\triangle_k$ before being blocked by a variable or by x_k itself. Then $\triangle_k$ is given by

$$\triangle_k = \min(\gamma_1, \gamma_2, u_k - l_k)$$

If $\triangle_k = \infty$, then the increase in x_k is unblocked and by Equation (2.5) the problem is unbounded. Otherwise if $\triangle_k < \infty$, a new bfs is obtained where $x_k = l_k + \triangle_k$ and the basic variables are modified according to Equation (2.4).

The current tableau must be updated to reflect the new bfs.

If $\triangle_k = u_k - l_k$, then no change of the working basis is made and x_k is still nonbasic, except this time it is nonbasic at its upper bound. Only the RHS column is changed. According to Equations (2.4) and (2.5), $\hat{z}$ is replaced by $\hat{z} - (z_k - c_k)\triangle_k$ and $\hat{\mathbf{b}}$ is replaced by $\hat{\mathbf{b}} - \mathbf{Y}_k \triangle_k$.

If $\triangle_k$ is given by γ_1 or γ_2, then x_k enters the basis and $x_{\mathbf{B}_r}$ leaves the basis, where the index r is determined according to Equation (2.6) if $\triangle_k = \gamma_1$ or according to Equation (2.7) if $\triangle_k = \gamma_2$. The tableau, except the RHS column, is updated by pivoting at $\mathbf{Y}_{rk}$. The right-hand-side column is updated according to Equations (2.4) and (2.5) except that the rth component of the new vector $\hat{\mathbf{b}}$ is replaced by $l_k + \triangle_k$.

Case 2: Decreasing x_k from its current level u_k

In this case $z_k - c_k < 0$ and $x_k = u_k - \triangle_k$, where $\triangle_k$ denotes the decrease in x_k. Noting Equations (2.2a) and (2.3) we get:

$$\mathbf{x_B} = \hat{\mathbf{b}} + \mathbf{Y}_k \triangle_k \tag{2.8}$$

$$z = \hat{z} + (z_k - c_k) \triangle_k \tag{2.9}$$

The maximum increase in $\triangle_k$ is given by

$$\triangle_k = \min(\gamma_1, \gamma_2, u_k - l_k)$$

where γ_1 and γ_2 are specified by:

$$\gamma_1 = \begin{cases} \min\limits_{1 \le i \le m} \left\{ \dfrac{\hat{b}_i - l_{B_i}}{-y_{ik}} : y_{ik} < 0 \right\} = \dfrac{\hat{b}_r - l_{B_r}}{-y_{rk}} & \text{if } \mathbf{Y}_k \not\geq 0 \\ \infty & \text{if } \mathbf{Y}_k \geq 0 \end{cases} \tag{2.10}$$

$$\gamma_2 = \begin{cases} \min\limits_{1 \le i \le m} \left\{ \dfrac{u_{B_i} - \hat{b}_i}{y_{ik}} : y_{ik} > 0 \right\} = \dfrac{u_{B_r} - \hat{b}_r}{y_{rk}} & \text{if } \mathbf{Y}_k \not\leq 0 \\ \infty & \text{if } \mathbf{Y}_k \leq 0 \end{cases} \tag{2.11}$$

If $\triangle_k = \infty$, then the decrease of x_k is unblocked and by Equation (2.9) the problem is unbounded.

If $\triangle_k < \infty$, then a new bfs is obtained where $x_k = u_k - \triangle_k$ and the basic variables are modified according to Equation (2.8).

- If $\triangle_k = u_k - l_k$, then x_k is still nonbasic but at its lower bound. The tableau is unchanged, except for the RHS column, which is updated according to Equations (2.8) and (2.9).
- If $\triangle_k$ is given by γ_1 or γ_2, then x_k enters the basis and $x_{\mathbf{B}_r}$ leaves the basis, where the index r is determined according to Equation (2.10) if $\triangle_k = \gamma_1$ or according to Equation (2.11) if $\triangle_k = \gamma_2$. The tableau except the RHS column is updated by pivoting at $\mathbf{Y}_{rk}$. The right-hand-side column is updated according to Equations (2.8) and (2.9) except that the rth component of the new vector $\hat{\mathbf{b}}$ is replaced by $u_k - \triangle_k$.

2.3 Constructing an initial basic feasible solution

The method discussed in Section 2.2 does not specify how to find an initial bfs. In those cases where it is not possible to easily find a bfs by examining the problem, then we may start by adding artificial variables and employ the *two-phase method* or the *big-M method* [2] to drive the artificial variables out of the basis.

2.4 Finite convergence: Degeneracy and cycling

In the abscence of degeneracy, at any iteration, the procedure described previously either moves from one bfs to an improved bfs, or declares unboundedness. Hence it must terminate finitely, either with an optimal solution or an indication of unboundedness, because the number of bfs is finite. However, in the presence of degeneracy, it is possible to perform a *degenerate pivot* in which γ_1 or γ_2 is zero. Hence the working basis changes, but we still remain at the same extreme point solution. Consequently, it is possible to *cycle*, that is, go through a sequence of consecutive degenerate pivots, and loop through the same set of working bases. For avoiding cycling, the *lexicographic rule* [2] can be adapted to problems with bounds.

Chapter 3

The decomposition principle and the column generation technique

3.1 Introduction

The *decomposition principle* refers to the application of the Dantzig-Wolfe or the Benders' or the Lagrangian relaxation techniques, which are all equivalent for linear programming problems. These techniques are used for dealing with large-scale and specially structured linear programming problems. In the case of large problems, the aim is to convert them into one or more smaller problems of manageable size. If some constraints possess a certain special structure, it may be useful to separate the linear program into one with general structure and one with special structure where a more efficient method can be applied.

On the one hand, the general *column generation* technique can be used to obtain the optimal solution of a linear program when it has a very large number of variables. The idea is to generate a column only if that column is eligible to enter the basis. A method of this type requires an efficient subroutine for identifying such a variable.

Consider the minimization standard form problem. Suppose that the number of columns (variables) is so large that it is impossible to generate and store the entire matrix of technological coefficients $\mathbf{A}$ in memory. Experience with large problems indicates that, usually, most of the columns never enter the basis, therefore we do not need to generate these unused columns. This reminds us of the revised simplex method which, at any given iteration, only requires the current basic columns and the column which is to enter the basis. There is only one problem that remains: discovering variables x_i with negative reduced costs $c_i - z_i$, without having to generate all columns. Sometimes, this can be accomplished by solving an optimization problem with a special structure: a smallest reduced cost $c_i - z_i$ can be found efficiently without computing every $c_i - z_i$.

On the other hand, in some linear programs, the constraints can be divided into two sets: general constraints (or *complicating* constraints) and constraints with special structure. The strategy of the decomposition procedure is to operate in two separate linear programs: one over the set of general constraints which is called the *master problem*, and one over the set of special constraints which is called the *subproblem*. The master problem passes down a new set of cost coefficients to the subproblem and receives from the subproblem a new column (entering variable) based on these cost coefficients. For this reason, the procedure is known as a *colum generation* technique.

Therefore, the key point of these methods is the ability to formulate and solve the optimization problem associated with determining a smallest reduced cost or solving the subproblem efficiently. There are a lot of variants of this general technique. Lübbecke and Desrosiers [9] give an insight into different selected topics on this matter.

3.2 The decomposition principle (bounded region case)

Consider the following linear program:

$$\begin{array}{ll} \text{Minimize} & \mathbf{cx} \\ \text{subject to:} & \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \in X \end{array}$$

where X is a polyhedral set representing constraints of special structure, $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{b}$ is an m vector.

Assume that X is bounded. Then, according to the representation theorem, any point $\mathbf{x} \in X$ can be represented as a convex combination of the finite number of extreme points of X [1]. Denoting these points by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$, any $\mathbf{x} \in X$ can be represented as:

$$\begin{gathered} \mathbf{x} = \sum_{j=1}^{t} \lambda_j \mathbf{x}_j \\ \sum_{j=1}^{t} \lambda_j = 1 \\ \lambda_j \geq 0,\ j = 1,2,\ldots,t \end{gathered}$$

Substituting for $\mathbf{x}$, the optimization problem can be transfomed into the following *master problem* in the variables $\lambda_1, \lambda_2, \ldots, \lambda_t$.

$$\text{Minimize} \sum_{j=1}^{t} (\mathbf{cx}_j)\lambda_j \tag{3.1a}$$

subject to:

$$\sum_{j=1}^{t} (\mathbf{Ax}_j)\lambda_j = \mathbf{b} \tag{3.1b}$$

$$\sum_{j=1}^{t} \lambda_j = 1 \tag{3.1c}$$

$$\lambda_j \geq 0 \ \ j = 1,2,\ldots,t \tag{3.1d}$$

Note that this problem might seem more difficult than the original one since we need to compute a priori every extreme point $\mathbf{x}_i$, $i = 1,\ldots,t$. The underlying idea of the decomposition principle is to find an optimal solution without explicitly computing all the extreme points.

3.2.1 Steps of the Dantzig-Wolfe decomposition technique

We may use the revised simplex method for solving problem (3.1). Suppose that we have a bfs $\lambda = (\lambda_{\mathbf{B}}, \lambda_{\mathbf{N}})$, and that the associated $(m+1) \times (m+1)$ basis inverse $\mathbf{B}^{-1}$ is known. Denoting the dual variables corresponding to Equations (3.1b) and (3.1c) by $\mathbf{w}$ and α, we get $(\mathbf{w}, \alpha) = \hat{\mathbf{c}}_{\mathbf{B}}\mathbf{B}^{-1}$, where $\hat{\mathbf{c}}_{\mathbf{B}}$ is the cost of the basic variables with $\hat{c}_j = \mathbf{c}x_j$ for each basic variable λ_j. The basis inverse, the dual variables, the values of the basic variables, and the objective function are displayed in Table 3.1 where, $\bar{\mathbf{b}} = \mathbf{B}^{-1} \begin{pmatrix} \mathbf{b} \\ 1 \end{pmatrix}$.

The simplex revised method proceeds by concluding that the current solution is optimal or else by deciding to increase a nonbasic variable λ_k. This is done by first calculating

$$z_k - \hat{c}_k = \max_{1 \leq j \leq t} z_j - \hat{c}_j = \max_{1 \leq j \leq t} (\mathbf{w}, \alpha) \begin{bmatrix} \mathbf{Ax}_j \\ 1 \end{bmatrix} - \mathbf{cx}_j = \max_{1 \leq j \leq t} \mathbf{wAx}_j + \alpha - \mathbf{cx}_j \tag{3.2}$$

BASIS INVERSE	RHS
$(\mathbf{w}, \alpha)$	$\hat{\mathbf{c}}_\mathbf{B}\bar{\mathbf{b}}$
$\mathbf{B}^{-1}$	$\bar{\mathbf{b}}$

Table 3.1: Master array

Since $z_j - \hat{c}_j = 0$ for the basic variables, then the foregoing maximum is greater than or equal to 0. Thus if $z_k - \hat{c}_k = 0$, then $z_j - \hat{c}_j \leq 0$ for all nonbasic variables and the current solution is optimal. On the other hand, if $z_k - \hat{c}_k > 0$, then the nonbasic variable λ_k may be increased.

Determining the index k using the Equation (3.2) directly is computationally cumbersome because t is very large. Since X is bounded, the maximum of any linear objective is achieved at one of its extreme points. Therefore,

$$\max_{1\leq j\leq t} (\mathbf{wA} - \mathbf{c})\mathbf{x}_j + \alpha = \max_{\mathbf{x}\in X} (\mathbf{wA} - \mathbf{c})\mathbf{x} + \alpha$$

To sum up we have to solve the following linear *subproblem* which is easy due to the special structure of X.

$$\begin{aligned} \text{Maximize} \quad & (\mathbf{wA} - \mathbf{c})\mathbf{x} + \alpha \\ \text{subject to:} \quad & \\ & \mathbf{x} \in X \end{aligned} \tag{3.3}$$

Since the objective function contains a constant, we must initialize the RHS value for z to α instead of 0. Let $\mathbf{x}_k$ be an optimal solution to the foregoing subproblem with objective value $z_k - \hat{c}_k$. If $z_k - \hat{c}_k = 0$, then the bfs $(\lambda_\mathbf{B}, \lambda_\mathbf{N})$ is optimal. Otherwise if $z_k - \hat{c}_k > 0$, then the variable λ_k enters the basis. The corresponding column is updated giving $Y_k = \mathbf{B}^{-1} \begin{pmatrix} \mathbf{A}x_j \\ 1 \end{pmatrix}$. Note that $Y_k \nleq 0$ since X was assumed bounded. The updated column $\begin{pmatrix} z_k - \hat{c}_k \\ Y_k \end{pmatrix}$ is adjoined to the revised simplex tableau. The variable $\lambda_{\mathbf{B}_r}$ leaving the basis is determined by the usual minimum ratio test. The basis inverse, dual variables, and RHS are updated by pivoting at y_{rk}. After updating, the process is repeated.

Note that the decomposition algorithm is a direct implementation of the revised simplex method except that the calculation of reduced costs $z_k - \hat{c}_k$ is performed by solving a subproblem. Therefore the algorithm converges in a finite number of iterations provided that a cycling prevention rule is used in both the master step and the subproblem.

3.2.2 Summary of the decomposition algorithm

INITIALIZATION STEP

Find an initial bfs of the system defined by Equations (3.1b), (3.1c) and (3.1d). Let the basis be $\mathbf{B}$ and form the *master array* displayed in Table 3.1.

MAIN STEP

1. Solve the *subproblem* (3.3). Let $\mathbf{x}_k$ be an optimal bfs with objective function value $z_k - \hat{c}_k$. If $z_k - \hat{c}_k = 0$ stop; the bfs of the last master step is an optimal solution to the overall problem. Otherwise go to step 2.
2. Let $Y_k = \mathbf{B}^{-1} \begin{pmatrix} \mathbf{A}\mathbf{x}_j \\ 1 \end{pmatrix}$ and adjoin the updated column $\begin{pmatrix} z_k - \hat{c}_k \\ Y_k \end{pmatrix}$ to the master array. Pivot at y_{rk} where the index r is determined as follows:

$$\frac{\bar{b}_r}{y_{rk}} = \min_{1\le i\le m+1}\left\{\frac{\bar{b}_i}{y_{ik}} : y_{ik} > 0\right\}.$$

This updates the dual variables, the basis inverse, and the right-hand-side. After pivoting, the column of λ_k is deleted and step 1 is repeated.

3.2.3 Calculation and use of lower bounds

The decomposition algorithm stops when $\max_j\{z_j - \hat{c}_j\} = 0$. Because of the large number of variables $\lambda_1, \lambda_2, \ldots, \lambda_t$, when we are dealing with very large problems, it is not convenient to perform all the computations.

Instead, we will adopt the following strategy: develop a lower bound on the objective function of any feasible solution of the overall problem, and hence a lower bound on the optimal objective. The decomposition algorithm generates feasible points with better (or at least not worse) objective values via the master problem, therefore, we have a sequence of nonincreasing upper bounds. Hence we may stop when the difference between the objective of the current feasible point and the lower bounds is within an acceptable tolerance. This may not give the true optimal point, but will guarantee good feasible solutions.

Consider subproblem (3.3) where $\mathbf{w}$ is the dual vector passed from the master step. Let the optimal objective value of the foregoing subproblem be $z_k - \hat{c}_k$. Now let $\mathbf{x}$ be any feasible solution of the overall problem, that is, $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{x} \in X$. By definition of $z_k - \hat{c}_k$ and since $\mathbf{x} \in X$, we have

$$(\mathbf{wA} - \mathbf{c})\mathbf{x} + \alpha \le z_k - \hat{c}_k$$

Since $\mathbf{Ax} = \mathbf{b}$, this inequality implies that

$$\mathbf{cx} \ge \mathbf{wAx} - (z_k - \hat{c}_k) + \alpha = \mathbf{wb} + \alpha - (z_k - \hat{c}_k) = \hat{\mathbf{c}}_\mathbf{B}\bar{\mathbf{b}} - (z_k - \hat{c}_k)$$

Since this is true for each $\mathbf{x} \in X$ with $\mathbf{Ax} = \mathbf{b}$, then

$$\min_{\{\mathbf{Ax}=\mathbf{b}, \mathbf{x}\in X\}} \mathbf{cx} \ge \hat{\mathbf{c}}_\mathbf{B}\bar{\mathbf{b}} - (z_k - \hat{c}_k)$$

In other words, $\hat{\mathbf{c}}_\mathbf{B}\bar{\mathbf{b}} - (z_k - \hat{c}_k)$ is a lower bound on the optimal objective value of the overall problem.

3.2.4 Obtaining a starting basic feasible solution

We distinguish two cases depending on the form of constraints (3.1b).

Case 1. Inequality constraints

Consider the problem (3.1), where constraints (3.1b) are inequalities.

If we find $\mathbf{x}_1 \in X$ such that $\mathbf{Ax}_1 \le \mathbf{b}$, then

$$\mathbf{B} = \begin{pmatrix} \mathbf{I} & \mathbf{Ax}_1 \\ \mathbf{0} & 1 \end{pmatrix} \qquad \mathbf{B}^{-1} = \begin{pmatrix} \mathbf{I} & -\mathbf{Ax}_1 \\ \mathbf{0} & 1 \end{pmatrix}$$

where the identity corresponds to the slack vector $\mathbf{s} \ge \mathbf{0}$.

The initial array is given by the following tableau

	BASIS	INVERSE	RHS
z	$\mathbf{0}$	$\mathbf{cx}_1$	$\mathbf{cx}_1$
s	$\mathbf{I}$	$-\mathbf{Ax}_1$	$\mathbf{b} - \mathbf{Ax}_1$
λ_1	$\mathbf{0}$	1	1

If there is no obvious $\mathbf{x} \in X$ with $\mathbf{Ax} \leq \mathbf{b}$, we can convert the master problem to equality form by adding appropriate slack variables, where the constraints are manipulated so that the RHS values are nonnegative. Then the artificial variables are added, as needed, to create an identity matrix. This identity matrix is the initial basis $\mathbf{B}$. The two-phase or big-M methods can be used to drive the artificial variables out of the basis. If at termination there is a positive artificial variable, then the overall problem has no feasible solution.

Case 2. Equality constraints

If constraints (3.1b) are equalities, $m+1$ artificial variables can be introduced to form the initial basis. The artificial variables are eliminated by the two-phase or by the big M-method.

3.3 The decomposition principle (unbounded region case)

Let X be an unbounded set, then $\mathbf{x} \in X$ if and only if

$$\begin{gathered}\mathbf{x} = \sum_{j=1}^{t} \lambda_j \mathbf{x}_j + \sum_{j=1}^{l} \mu_j \mathbf{d}_j \\ \sum_{j=1}^{t} \lambda_j = 1 \\ \lambda_j \geq 0,\ j = 1,2,\ldots,t \\ \mu_j \geq 0,\ j = 1,2,\ldots,l\end{gathered}$$

where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$ are the extreme points of X and $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_l$ are the extreme directions of X. The primal problem can be transformed into a master problem with the variables $\lambda_1, \lambda_2, \ldots, \lambda_t$ and $\mu_1, \mu_2, \ldots, \mu_l$ as follows:

$$\text{Minimize } \sum_{j=1}^{t} (\mathbf{cx}_j)\lambda_j + \sum_{j=1}^{l} (\mathbf{cd}_j)\mu_j \tag{3.4a}$$

subject to:

$$\sum_{j=1}^{t} (\mathbf{Ax}_j)\lambda_j + \sum_{j=1}^{l} (\mathbf{Ad}_j)\mu_j = \mathbf{b} \tag{3.4b}$$

$$\sum_{j=1}^{t} \lambda_j = 1 \tag{3.4c}$$

$$\lambda_j \geq 0 \ j = 1,2,\ldots,t$$

$$\mu_j \geq 0 \ j = 1,2,\ldots,l$$

Suppose that we have a bfs of the foregoing system with basis $\mathbf{B}$, and let $\mathbf{w}$ and α be the dual variables corresponding to Equations (3.4b) and (3.4c) respectively. Furthermore suppose that $\mathbf{B}^{-1}$, $(\mathbf{w}, \alpha) = \hat{\mathbf{c}}_\mathbf{B}\mathbf{B}^{-1}$ ($\hat{\mathbf{c}}_\mathbf{B}$ is the cost of the basic variables), and $\bar{b} = \mathbf{B}^{-1}\begin{pmatrix} \mathbf{b} \\ 1 \end{pmatrix}$ are known and displayed. Consider a master array like the one shown in Table 3.1. Recall that the current solution is optimal to the overall problem if $z_j - \hat{c}_j \leq 0$ for each variable. In particular the following conditions must hold at optimality

$$\lambda_j \text{nonbasic} \Rightarrow 0 \geq z_j - \hat{c}_j = (\mathbf{w}, \alpha)\begin{pmatrix} \mathbf{Ax}_j \\ 1 \end{pmatrix} - \mathbf{cx}_j = \mathbf{wAx}_j + \alpha - \mathbf{cx}_j \tag{3.5}$$

$$\mu_j \text{nonbasic} \Rightarrow 0 \geq z_j - \hat{c}_j = (\mathbf{w}, \alpha)\begin{pmatrix} \mathbf{Ad}_j \\ 0 \end{pmatrix} - \mathbf{cd}_j = \mathbf{wAd}_j - \mathbf{cd}_j \tag{3.6}$$

Since the number of nonbasic variables is very large, checking conditions (3.5) and (3.6) by generating the corresponding extreme points and directions is computationally infeasible. However, we may

determine whether or not these conditions hold by solving *subproblem* (3.3). If the optimal solution of the subproblem is unbounded, that is, if there exists an extreme direction $\mathbf{d}_k$ such that $(\mathbf{wA}-\mathbf{c})\mathbf{d}_k > 0$, then condition (3.6) is violated. Moreover $z_k - \hat{c}_k = (\mathbf{wA}-\mathbf{c})\mathbf{d}_k > 0$ and μ_k is eligible to enter the basis. In this case $\begin{pmatrix} \mathbf{Ad}_k \\ 0 \end{pmatrix}$ is updated by premultiplying by $\mathbf{B}^{-1}$ and the resulting column $\begin{pmatrix} z_k - \hat{c}_k \\ Y_k \end{pmatrix}$ is inserted in the array of Table 3.1 and the revised simplex method is continued.

Otherwise, if the solution of the subproblem is bounded, that is, if $(\mathbf{wA}-\mathbf{c})\mathbf{d}_j \leq 0$ for all extreme directions then condition (3.6) holds. Now we check whether (3.5) holds. Let $\mathbf{x}_k$ be an optimal extreme point and consider the optimal objective, $z_k - \hat{c}_k$ to the subproblem. If $z_k - \hat{c}_k \leq 0$, then by optimality of $\mathbf{x}_k$, for each exteme point $\mathbf{x}_j$ we have

$$(\mathbf{wA}-\mathbf{c})\mathbf{x}_j + \alpha \leq (\mathbf{wA}-\mathbf{c})\mathbf{x}_k + \alpha = z_k - \hat{c}_k \leq 0$$

and hence condition (3.5) holds and we stop with an optimal solution of the overall problem. If, on the other hand, $z_k - \hat{c}_k > 0$, then λ_k is introduced in the basis. This is done by inserting the column $\begin{pmatrix} z_k - \hat{c}_k \\ Y_k \end{pmatrix}$ into the array shown in Table 3.1 and pivoting, where $Y_k = \mathbf{B}^{-1}\begin{pmatrix} \mathbf{Ax}_k \\ 1 \end{pmatrix}$. Note that if the master problem includes slack or other explicitly present variables then the $z_j - \hat{c}_j$ values for these variables must be checked before deducing optimality. Also, the bounded subproblems yield lower bounds as in the bounded case.

3.4 Block diagonal or angular structure

A special version of the decomposition principle has been successfully applied to problems that have a certain structure known as *block diagonal* or *angular structure*. In this kind of problems there is a set of constraints that affects to all the variables whereas other sets of constraints only affect to some blocks of variables. Let X be a set that has a block diagonal structure. Then X can be decomposed into several sets $X_1, X_2, \ldots, X_T$, each involving a subset of the variables, which do not appear in any other set. If we decompose the $\mathbf{x}$ vector accordingly into the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$, the vector $\mathbf{c}$ into $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_T$, and the matrix $\mathbf{A}$ of the master constraints $\mathbf{Ax} = \mathbf{b}$ into matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_T$, we get the following problem:

$$\begin{aligned}
&\text{Minimize } \mathbf{c}_1\mathbf{x}_1 + \mathbf{c}_2\mathbf{x}_2 + \ldots + \mathbf{c}_T\mathbf{x}_T \\
&\text{subject to:} \\
&\qquad \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 + \ldots + \mathbf{A}_T\mathbf{x}_T = \mathbf{b} \\
&\qquad \mathbf{B}_1\mathbf{x}_1 \leq \mathbf{b}_1 \\
&\qquad \mathbf{B}_2\mathbf{x}_2 \leq \mathbf{b}_2 \\
&\qquad \vdots \\
&\qquad \mathbf{B}_T\mathbf{x}_T \leq \mathbf{b}_T \\
&\qquad \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T \geq 0
\end{aligned}$$

where $X_i = \{\mathbf{B}_i\mathbf{x}_i \leq \mathbf{b}_i, \mathbf{x}_i \geq 0\}$ for $i = 1, 2, \ldots, T$. Problems with this structure can be solved by the decomposition algorithm and the block diagonal or angular structure of X can be utilized further.

For subproblem i, $\mathbf{x}_i \in X_i$ if and only if:

$$\begin{aligned}
\mathbf{x}_i &= \sum_{j=1}^{t_i} \lambda_{ij}\mathbf{x}_{ij} + \sum_{j=1}^{l_i} \mu_{ij}\mathbf{d}_{ij} \\
\sum_{j=1}^{t_i} \lambda_{ij} &= 1 \\
\lambda_{ij} &\geq 0,\ j = 1, 2, \ldots, t_i \\
\mu_{ij} &\geq 0,\ j = 1, 2, \ldots, l_i
\end{aligned}$$

where the $\mathbf{x}_{ij}$'s and the $\mathbf{d}_{ij}$'s are the extreme points and the extreme directions (if any) of X_i. Replacing each $\mathbf{x}_i$ by the foregoing representation, the original problem can be reformulated as the following *master problem* with $m+T$ constraints:

$$\text{Minimize } \sum_{i=1}^{T}\sum_{j=1}^{t_i}(\mathbf{c}_i\mathbf{x}_{ij})\lambda_{ij}+\sum_{i=1}^{T}\sum_{j=1}^{l_i}(\mathbf{c}_i\mathbf{d}_{ij})\mu_{ij} \tag{3.8a}$$

subject to:

$$\sum_{i=1}^{T}\sum_{j=1}^{t_i}(\mathbf{A}_i\mathbf{x}_{ij})\lambda_{ij}+\sum_{i=1}^{T}\sum_{j=1}^{l_i}(\mathbf{A}_i\mathbf{d}_{ij})\mu_{ij}=\mathbf{b} \tag{3.8b}$$

$$\sum_{j=1}^{t_i}\lambda_{ij}=1,\ i=1,\ldots,T \tag{3.8c}$$

$$\lambda_{ij}\geq 0,\ j=1,2,\ldots,t_i\ i=1,\ldots,T$$

$$\mu_{ij}\geq 0,\ j=1,2,\ldots,l_i\ i=1,\ldots,T$$

Suppose that we have a bfs of the foregoing system with an $(m+T)\times(m+T)$ basis $\mathbf{B}$. Note that each basis must contain at least one variable λ_{ij} from each block i. Further suppose that $\mathbf{B}^{-1}$, $\bar{\mathbf{b}}=\mathbf{B}^{-1}\begin{pmatrix}\mathbf{b}\\1\end{pmatrix}$, $(\mathbf{w},\alpha)=(w_1,\ldots,w_m,\alpha_1,\ldots,\alpha_T)=\hat{\mathbf{c}}_\mathbf{B}\mathbf{B}^{-1}$ are known, where $\hat{\mathbf{c}}_\mathbf{B}$ is the cost of the basic variables ($\hat{c}_{ij}=\mathbf{c}_i\mathbf{x}_{ij}$ for λ_{ij} and $\hat{c}_{ij}=\mathbf{c}_i\mathbf{d}_{ij}$ for μ_{ij}). This solution is optimal if $z_{ij}-\hat{c}_{ij}\leq 0$ for each variable (we know that $z_{ij}-\hat{c}_{ij}=0$ for each basic variable). In particular the following optimality condition must hold at optimality:

$$\lambda_{ij}\text{nonbasic}\Rightarrow 0\geq z_{ij}-\hat{c}_{ij}=\mathbf{w}\mathbf{A}_i\mathbf{x}_{ij}+\alpha_i-\mathbf{c}_i\mathbf{x}_{ij} \tag{3.9}$$

$$\mu_{ij}\text{nonbasic}\Rightarrow 0\geq z_{ij}-\hat{c}_{ij}=\mathbf{w}\mathbf{A}_i\mathbf{d}_{ij}-\mathbf{c}_i\mathbf{d}_{ij} \tag{3.10}$$

Whether conditions (3.9) and (3.10) hold or not, can be easily verified by solving the following subproblems:

$$\begin{aligned}&\text{Maximize} && (\mathbf{w}\mathbf{A}_i-\mathbf{c}_i)\mathbf{x}_i+\alpha_i\\ &\text{subject to:}\\ &&& \mathbf{x}_i\in X_i\end{aligned}$$

If the optimal solution value is unbounded, then an extreme direction $\mathbf{d}_{ik}$ is found such that $(\mathbf{w}\mathbf{A}_i-\mathbf{c}_i)\mathbf{d}_{ik}>0$; that is, condition (3.10) is violated and introducing μ_{ik} will improve the objective function since $z_{ik}-\hat{c}_{ik}=(\mathbf{w}\mathbf{A}_i-\mathbf{c}_i)\mathbf{d}_{ik}>0$. If the optimal solution is bounded, then automatically condition (3.10) holds for subproblem i. Let $\mathbf{x}_{ik}$ be an optimal extreme point. If the optimal objective value $z_{ik}-\hat{c}_{ik}=\mathbf{w}\mathbf{A}_i\mathbf{x}_{ik}+\alpha_i-\mathbf{c}_i\mathbf{x}_{ik}\leq 0$, then condition (3.9) holds for subproblem i. Otherwise, λ_{ik} can be introduced in the basis. When all subproblems have $z_{ik}-\hat{c}_{ik}\leq 0$, then an optimal solution to the original problem is obtained. If the master problem contains other explicit variables including slack variables, then we must also check the $z_j-\hat{c}_j$ for these variables before terminating.

When we have various candidates to enter the master basis we may use the rule of the most positive $z_{ik}-\hat{c}_{ik}$. On selecting the candidate, we update the entering column, pivot on the master array, and repeat the process.

Notice that we can compute lower bounds for the case of bounded subproblems. Indeed, let $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_T$ represent a bfs of the overall problem so that $\mathbf{x}_i\in X_i$ for each i and $\sum_i\mathbf{A}_i\mathbf{x}_i=\mathbf{b}$. By definition of $z_{ik}-\hat{c}_{ik}$ we have $(\mathbf{w}\mathbf{A}_i-\mathbf{c}_i)\mathbf{x}_i+\alpha_i\leq(z_{ik}-\hat{c}_{ik})$ or $\mathbf{c}_i\mathbf{x}_i\geq\mathbf{w}\mathbf{A}_i\mathbf{x}_i+\alpha_i-(z_{ik}-\hat{c}_{ik})$.

Summing on i we get $\sum_i\mathbf{c}_i\mathbf{x}_i\geq\mathbf{w}\sum_i\mathbf{A}_i\mathbf{x}_i+\sum_i\alpha_i-\sum_i(z_{ik}-\hat{c}_{ik})$ but $\sum_i\mathbf{c}_i\mathbf{x}_i=\mathbf{c}\mathbf{x}$ and $\sum_i\mathbf{A}_i\mathbf{x}_i=\mathbf{b}$. Thus we get $\mathbf{c}\mathbf{x}\geq\mathbf{w}\mathbf{b}+\alpha\mathbf{1}-\sum_i(z_{ik}-\hat{c}_{ik})$ or $\mathbf{c}\mathbf{x}\geq\hat{\mathbf{c}}_\mathbf{B}\bar{\mathbf{b}}-\sum_i(z_{ik}-\hat{c}_{ik})$ the right-hand-side in this inequality provides a lower bound on the problem. This is a natural extension of the case for one bounded subproblem presented in Section 3.2.3.

3.5 Benders' decomposition for mixed integer optimization problems

Benders' decomposition which is equivalent for linear programming problems to the Dantzig-Wolfe decomposition procedure is mainly applied to solve large integer optimization problems. Its aim is to decompose a mixed integer program into its integer and its continuous parts. Consider the problem:

$$\begin{array}{ll} \text{Minimize} & z(\mathbf{x},\mathbf{y}) = \mathbf{cx}+\mathbf{dy} \\ \text{subject to:} & \\ & \mathbf{Ax}+\mathbf{Dy} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0},\ \mathbf{y} \geq \mathbf{0},\ \mathbf{y} \text{ an integer vector} \end{array} \tag{3.11}$$

Let $S = \{\mathbf{y} : \mathbf{y} \geq \mathbf{0}$ and $\mathbf{y}$ is an integer vector$\}$. For a fixed $\mathbf{y} \in S$, problem (3.11) is equivalent to:

$$\begin{array}{ll} & \mathbf{dy} + \min(\mathbf{cx}) \\ \text{subject to:} & \\ & \mathbf{Ax} \leq \mathbf{b}-\mathbf{Dy} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{3.12}$$

A vector $\mathbf{y}$ is said to be *admissible* if there exists an $\mathbf{x}$ such that ($\mathbf{x}$,$\mathbf{y}$) satisfies (3.11), that is, if (3.12) is feasible in $\mathbf{x}$. If there is no admissible $\mathbf{y}$, (3.11) is infeasible.

Problem (3.12) is linear. Its dual problem is:

$$\begin{array}{ll} v(\mathbf{y}) = \max\limits_{\pi} v(\pi,\mathbf{y}) = \mathbf{dy}+\pi(\mathbf{Dy}-\mathbf{b}) & \\ \text{subject to:} & \\ & -\pi\mathbf{A} \leq \mathbf{c} \\ & \pi \geq \mathbf{0} \end{array} \tag{3.13}$$

where π denotes the vector of dual variables. The constraints in (3.13) are independent of $\mathbf{y}$. If (3.13) is infeasible, by the duality theorem [2], the objective function in (3.12) is unbounded below for every admissible $\mathbf{y} \in S$, that is, either (3.11) is infeasible or $\mathbf{cx}+\mathbf{dy}$ is unbounded below in it. Hence (3.11) does not have a finite optimum solution in this case.

Otherwise, if (3.13) is feasible, since for any fixed $\mathbf{y} \in S$, (3.11) is equivalent to (3.12), and (3.13) is the dual of (3.12), we conclude from the duality theorem that (3.11) is equivalent to:

$$\begin{array}{ll} \text{Minimize} & v(\mathbf{y}) \\ \text{over admissible} & \mathbf{y} \in S \end{array} \tag{3.14}$$

Let's check when $\mathbf{y} \in S$ is admissible. Since (3.13) is feasible, the objective function in it is unbounded above if and only if (3.12) is infeasible (by the duality theorem), that is, if and only if $\mathbf{y}$ is inadmissible. On the other hand, we know [5], that the objective function in (3.13) is unbounded above in this case if and only if there exists an extreme homogeneous solution μ corresponding to (3.13), which satisfies $\mu(\mathbf{Dy}-\mathbf{b}) > 0$. A vector $\mathbf{x}$ is termed as an extreme homogeneous solution if and only if it is a bfs to the normalized homogeneous system $\mathbf{Ax}$=0, $\mathbf{1}^t\mathbf{x} = 1$, $\mathbf{x} \geq \mathbf{0}$. Let $\mu^1,\ldots,\mu^p$ be all the extreme homogeneous solutions corresponding to (3.13). Then we have that $\mathbf{y} \in S$ is admissible if and only if:

$$\mu^t(\mathbf{Dy}-\mathbf{b}) \leq 0, \text{ for all } t = 1,\ldots,p,\ \mathbf{y} \in S \tag{3.15}$$

Therefore, we conclude that (3.11) is equivalent to:

$$\begin{array}{ll} \text{Minimize} & v(\mathbf{y}) \\ \text{subject to:} & \\ & \mu^t(\mathbf{Dy}-\mathbf{b}) \leq 0, \quad \text{for all } t = 1,\ldots,p \\ & \mathbf{y} \in S \end{array} \tag{3.16}$$

Let $\pi^1,\ldots,\pi^l$ be all the bfs of (3.13). If (3.15) is satisfied, then

$$v(\mathbf{y}) = \max_{j=1,\ldots,l} \{\mathbf{dy}+\pi^j(\mathbf{Dy}-\mathbf{b})\} \tag{3.17}$$

From (3.16) and (3.17) we conclude that the problem (3.11) is equivalent to:

$$\begin{array}{ll} \text{Minimize} & z_0 \\ \text{subject to:} & \\ & z_0 - \mathbf{d}\mathbf{y} - \pi^j(\mathbf{D}\mathbf{y} - \mathbf{b}) \geq 0, \quad \text{for all } j = 1, \ldots, l \\ & \mu^t(\mathbf{D}\mathbf{y} - \mathbf{b}) \leq 0, \quad \text{for all } t = 1, \ldots, p \\ & \mathbf{y} \geq 0, \text{ integer} \end{array} \tag{3.18}$$

Note that (3.18) is a pure integer problem with $l + p$ constraints. Hence, in general, it is a large problem. The idea of the Bender's algorithm is to consider only a subset of the constraints of (3.18) in each stage, obtaining a *restricted problem*.

***Initial step*:**

Select a $\bar{\mathbf{y}} \in S$, say $\bar{\mathbf{y}} = \mathbf{0}$, and solve (3.13) for that $\bar{\mathbf{y}}$. If (3.13) has an optimal solution in this case, let π^1 be an optimal bfs for it. On the other hand, if the objective function in (3.13) is unbounded above in this case, let μ^1 be the extreme homogeneous solution corresponding to the extreme half-line along which $v(\pi, \bar{\mathbf{y}})$ diverges to $+\infty$. In the initial stage the restricted problem is (3.18) after eliminating all the constraints in it excepting those corresponding to some known bfs π^j, and the one constraint corresponding to π^1 or μ^1 as the case may be, and $\mathbf{y} \in S$.

***Main step*:**

Suppose that the restricted problem is:

$$\begin{array}{ll} \text{Minimize} & z_0 \\ \text{subject to:} & \\ & z_0 - \mathbf{d}\mathbf{y} - \pi^j(\mathbf{D}\mathbf{y} - \mathbf{b}) \geq 0, \quad \text{for all } j = 1, \ldots, r \\ & \mu^t(\mathbf{D}\mathbf{y} - \mathbf{b}) \leq 0, \quad \text{for all } t = 1, \ldots, k \\ & \mathbf{y} \geq 0, \text{ integer} \end{array} \tag{3.19}$$

The computations involved in this stage are the following:

1. Solve (3.19) by using some pure integer programming method. If (3.19) is infeasible, (3.18) and hence the original problem must be infeasible. Terminate. Otherwise go to step 2.

2. Let $(\hat{z}_0, \hat{\mathbf{y}})$ be an optimal solution of (3.19). $\hat{z}_0$ is a lower bound for the minimum objective value in (3.18) or (3.11).

Solve (3.13) for $\mathbf{y} = \hat{\mathbf{y}}$. In this case, if (3.13) has the objective function unbounded above, let μ^{k+1} be the extreme homogeneous solution corresponding to the extreme half-line along which $v(\pi, \hat{\mathbf{y}})$ diverges to $+\infty$. Include the constraint $\mu^{k+1}(\mathbf{D}\mathbf{y} - \mathbf{b}) \leq 0$ in (3.19) and with this as the new restricted problem repeat the main step.

On the other hand, if (3.13) has a finite optimal solution in this case, let π^{r+1} be an optimal bfs for it. The optimum objective value is $\mathrm{v}(\pi^{r+1}, \hat{\mathbf{y}})$=$\mathrm{v}(\hat{\mathbf{y}})$. If $\hat{z}_0 \geq \mathrm{v}(\hat{\mathbf{y}})$, then $\hat{z}_0$ is the optimum objective value in (3.18) or (3.11). Let $\hat{\mathbf{x}}$ be an opitmum solution of (3.12) with $\mathbf{y} = \hat{\mathbf{y}}$. Then $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is an optimum solution of (3.11). Terminate.

Otherwise, if $\hat{z}_0 < \mathrm{v}(\hat{\mathbf{y}})$, the optimum objective value of (3.18) or (3.11) lies in the interval $[\hat{z}_0, \mathrm{v}(\hat{\mathbf{y}})]$. Include the constraint $z_0 - \mathbf{d}\mathbf{y} - \pi^{r+1}(\mathbf{D}\mathbf{y} - \mathbf{b}) \geq 0$ in (3.19) and with this as the new restricted problem repeat the main step. The new constraint included here prevents $\hat{\mathbf{y}}$ from appearing as an optimum solution of the restricted problem in subsequent stages.

3.6 Column generation technique: A variant involving retained columns

In this variant of the method, the algorithm retains in memory all or some of the columns that have been generated in the past, and proceeds in terms of restricted linear programming problems that involve only the retained columns.

We may describe the algorithm as a sequence of master iterations. At the beginning of a master iteration, we have a bfs to the original problem, and an associated basis matrix. We search for a variable with negative reduced cost $c_i - z_i$, possibly by minimizing $c_i - z_i$ over all i; if none is found, the algorithm terminates. Suppose that we have found some j such that $c_j - z_j < 0$. We then form a collection of columns $\mathbf{A}_i$, $i \in I$, which contains all of the basic columns, the entering column $\mathbf{A}_j$, and possibly some other columns as well. Let's define the restricted problem:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i \in I} c_i x_i \\ \text{subject to:} \quad & \\ & \sum_{i \in I} \mathbf{A}_i x_i = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Recall that the basic variables at the current bfs to the original problem are among the columns that have been kept in the restricted problem. We therefore have a bfs to the restricted problem, which can be used as a starting point for its solution. We then perform as many simplex iterations as needed, until the restricted problem is solved to optimality. At that point, we are ready to start with the next master iteration.

The method we have just described is a special case of the revised simplex method, in conjunction with some special rules for choosing the entering variable that give priority to the variables x_i, $i \in I$; it is only when the reduced costs of these variables are all nonnegative (which happens at an optimal solution to the restricted problem) that the algorithm examines the reduced costs of the remaining variables. The motivation is that we may wish to give priority to variables for which the corresponding columns have already been generated and stored in memory, or to variables that are more likely to have negative reduced cost. There are several variants of this method, depending on the manner that the set I is chosen at each iteration.

- At one extreme, I is just the set of indices of the current basic variables, together with the entering variable; a variable that exits the basis is immediately dropped from the set I. Since the restricted problem has $m+1$ variables and m constraints, its feasible set is at most one-dimensional, and it gets solved in a single simplex iteration, that is, as soon as the column $\mathbf{A}_j$ enters the basis.

- At the other extreme, we let I be the set of indices of all variables that have become basic at some point in the past; equivalently, no variables are ever dropped, and each entering variable is added to I. If the number of master iterations is large, this option can be problematic because I keeps growing.

- Finally, there are intermediate options in which the set I is kept to a moderate size by dropping from I those variables that have exited the basis in the remote past and have not reentered since.

In the absence of degeneracy, all of the above variants are guaranteed to terminate because they are special cases of the revised simplex method. In the presence of degeneracy, cycling can be avoided by using the revised simplex method in conjunction with the lexicographic tie breaking rule.

3.7 The cutting stock problem

The cutting stock problem is a classical example of application of the column generation technique. Consider a paper company that has a supply of large rolls of paper of width V where V is a positive integer. However, customer demand is for smaller widths of paper; in particular we need to produce b_i rolls of width v_i, $i = 1, 2, \ldots, m$. We assume that $v_i \leq V$ for each i, and that each v_i is an integer. Smaller rolls are obtained by slicing a large roll in a certain way, called a *pattern*. For example, a large roll of width 70 can be cut into three rolls of width $v_1 = 17$ and one roll of width $v_2 = 15$, with a waste of 4.

The jth pattern can be represented by a column vector $\mathbf{A}_j$ whose ith entry a_{ij} indicates how many rolls of width v_i are produced by that pattern. For example the pattern described earlier is represented by the vector $(3,1,0,\ldots,0)$. For a vector $(a_{1j},\ldots,a_{mj})$ to be a representation of a feasible pattern, its components must be nonnegative integers and we must also have

$$\sum_{i=1}^{m} a_{ij} v_i \leq V \tag{3.20}$$

Let n be the number of all feasible patterns and consider the $m \times n$ matrix $\mathbf{A}$ with columns $\mathbf{A}_j$, $j = 1,\ldots,n$. Note that n can be a very large number. The goal of the company is to minimize the number of large rolls used while satisfying customer demand. Let x_j be the number of large rolls cut according to pattern j. Then the problem under consideration is:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^{n} x_j \\ \text{subject to:} \quad & \\ & \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1,\ldots,m \\ & x_j \geq 0, \qquad\quad j = 1,\ldots,n \end{aligned} \tag{3.21}$$

This problem does not take into account the fact that each x_j should be an integer and so we have an integer programming problem. However, an optimal solution to the linear programming problem (3.21) often provides a feasible solution to the integer programming problem (by rounding or other methods), which is fairly close to optimal, at least if the demands b_i are reasonably large.

Solving the linear problem (3.21) is a difficult computational task. Even if m is comparatively small, the number of feasible patterns n can be huge, so that forming all the coefficients of the matrix $\mathbf{A}$ is impractical. However, we will now show that the problem can be solved efficiently, by using the revised simplex method and by generating columns of $\mathbf{A}$ as needed rather than in advance.

Finding an initial bfs is easy for this problem. For $j = 1,\ldots,m$, we may let the jth pattern consist of one roll of width v_j and none of the other widths. Then, the first m columns of $\mathbf{A}$ form a basis that leads to a bfs, whose the corresponding basis matrix is the identity.

Suppose now that we have a basis matrix $\mathbf{B}$ and an associated bfs, and that we wish to carry out the next iteration of the revised simplex method. Because the cost coefficient of every variable x_j is unity, every component of the vector $\mathbf{c_B}$ is equal to 1. We compute the dual variables $\mathbf{w} = \mathbf{c_B}\mathbf{B}^{-1}$. Next, instead of computing the reduced cost $c_j - z_j = 1 - \mathbf{w}\mathbf{A}_j$ associated with every column (pattern) $\mathbf{A}_j$, we consider the problem of minimizing $1 - \mathbf{w}\mathbf{A}_j$ over all j. This is the same as maximizing $\mathbf{w}\mathbf{A}_j$ over all j. If the maximum is less than or equal to 1, all reduced costs are nonnegative and we have and optimal solution. If on the other hand, the maximum is greater than 1, the column $\mathbf{A}_j$ corresponding to a maximizing j has negative reduced cost and enters the basis.

We are now left with the task of finding a pattern j that maximizes $\mathbf{w}\mathbf{A}_j$. Given our earlier description of what constitutes an admissible pattern we have to solve the problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{m} w_i a_i \\ \text{subject to:} \quad & \\ & \sum_{i=1}^{m} v_i a_i \leq V \\ & a_i \geq 0, \qquad i = 1,\ldots,m \\ & a_i \text{ integer}, \quad i = 1,\ldots,m \end{aligned}$$

This is the well-known *integer knapsack problem*, where w_i refers to the value of the ith item, and v_i is the weight of the ith item. We seek to fill a knapsack and maximize its value without the total weight exceeding V. Solving the knapsack problem requires some effort, but for the range of numbers that arise in the cutting stock problem, this can be done fairly efficiently.

Chapter 4

Applications

4.1 Introduction

In this chapter we will discuss two real life examples of the application of the methods described in this project. Our aim is to show that nowadays, these methods are actually widely applied. They are mainly used as an structure embedded in elaborated algorithms for solving complex integer problems. Among the several examples one may find in the literature, we have selected two papers. The first paper which is related to the school bus routing problem, is described in [8] and [11]. The second paper, whose aim is to determine the best location for a certain number of clinics, is described in [10].

4.2 School bus routing

The papers by Schittekat, Kinable, Sörensen, Sevaux, Spieksma and Springael [11] and Kinable, Spieksma, and Vanden Berghe [8], analyse the school bus routing problem (SBRP) as a variant of the vehicle routing problem in which the three decisions to be made are:

1. Determine the set of stops
2. Determine which student is assigned to which stop
3. Determine routes among the chosen stops

The objective of the problem is to minimize the total traveled distance.

The problem was motivated by the Flemish region of Belgium, where students that live within certain minimum and maximum distances of their school get free transport to school. It is required that a bus stop is at a distance of, at most, 750 meters from the home of each student. In each country, possible locations for the bus stops are restricted by local policies, such as the maximum walking distance to the stop or safety regulations. Each school term, an initial set of potential stops is determined in such a way that each student lives within 750 meters of at least one stop. Then routes are determined, assuring that the capacity of the buses is not exceeded. The transportation company has to solve this problem taking into account all the students from different schools.

The main difference among this problem and most vehicle routing problems is the fact that the set of stops is not fixed but only potential. When a student can be assigned to multiple stops along the same route, it is done arbitrarily. On the other hand, if a student can be assigned to multiple stops in different routes then he is distributed in such a way that the capacity of the buses is not exceeded.

The paper focuses on a single-school SBRP, that can be described as follows. Let V be a given set of bus stops (including the school) with a distance for each ordered pair of stops, and S a set of students. For each student $s \in S$, the set $V_s \subseteq V$ represents the set of stops to which the student can be asigned. Assigning student s to a stop in V_s is called a *feasible* assignment. All the vehicles have capacity Q. A *route* is defined as a sequence of stops ending with the school. The aim of the problem is to find a

feasible assignment of students to stops, and find routes for the vehicles verifying (a) the capacity of each vehicle is respected, (b) each student is picked up, and (c) total length of the routes is minimized.

The earliest papers discussing school bus routing tried to solve the problem via decomposition, that is, the selection, assignment and routing prolems are solved independently. This approach results in excessive and suboptimal routes. The discussions of these strategies motivate the use of a more integrated approach that treats the SBRP problem as an integral problem.

4.2.1 Problem formulation

The SBRP can be formulated as follows:

$$
\begin{aligned}
&\text{Minimize} && \sum_{p\in P} \delta_p z_p \\
&\text{subject to:} && \\
& && \sum_{p\in P} t_{sp} z_p \geq 1,\ \forall s \in S \\
& && \sum_{p\in P} r_{vp} z_p \leq 1,\ \forall v \in V \\
& && \sum_{p\in P} z_p \leq U \\
& && \sum_{p\in P} z_p \geq L \\
& && z_p \in \{0,1\},\ \forall p \in P
\end{aligned}
$$

where:

Variable/Parameter	Description
z_p	1 if bus schedule $p \in P$ is used, 0 otherwise
V	Set of bus stops (including the school v_0)
S	Set of students
V_s	The set of stops student $s \in S$ can reach
P	Set of all bus schedules
p	Index corresponding to a bus schedule
t_{sp}	1 if student s is picked up in bus schedule p, 0 otherwise
r_{vp}	1 if stop v is part of bus schedule p, 0 otherwise
Q	Maximum capacity of the buses
δ_p	Cost induced by schedule p
L, U	Lower and upper bound on the number of buses

Note that a *bus schedule* p is an ordered sequence of stops the bus driver should visit, ending at the school, as well as the specific students who should be picked up at the corresponding stops. We will denote this formulation the master problem MP.

4.2.2 Column generation

The foregoing problem requires an exponentially large set of columns P. The first step to solve it is to replace the integrality constraints by $z_p \geq 0$ obtaining a relaxation, denoted LPM, which is solved via a column generation procedure. The set of columns P is replaced by a subset $P' \subseteq P$, $|P'| \ll |P|$, therefore a reduced version of the LPM called the restricted MP (RMP) is solved. P' contains an initial feasible solution that is produced by the heuristic proposed in [11]. The column generation procedure can be improved by using stabilization and adding a column pool manager (see [8]). This procedure has empirically been proved to significantly improve computation times.

4.3 Locating roadside clinics in Africa

The paper by Núñez Ares, De Vries and Huisman [10], deals with the problem of locating the so-called Roadside Wellness Centers (RWCs) so as to maximize the expected patient volume, ensure continuity of access along the routes and provide equal access. The paper considers the problem to locate a fixed number of RWCs based on these effectiveness and equity objectives.

The problem was motivated by the extreme vulnerability to infectious diseases, such as HIV, of long truck drivers in Sub-Saharan Africa. The determinants seem to be loneliness, long separation from home, monotony, and stress, which make them engage in high-risk sexual behaviors. This situation brings health, social and economic risks to the population. Traditional healthcare facilities are generally incapable of controlling HIV spread by providing prevention treatment and care services. The main reasons are that truck drivers are not allowed to deviate from their routes, the facilities are not accessible by truck, and that the schedule of the clinics are inconvenient to the drivers.

North Star Alliance, a non-governmental organization aims to provide RWCs in order to solve the problem discussed. A requirement for health services to be effective is for drivers to be sufficiently close to an RWC at every moment during their trip.

Given a number of RWCs, choosing their location lead to complex optimization problems. In [12] a mixed-integer programming (MIP) formulation is proposed. The aim of this problem is to locate a given number of new RWCs and deciding which health services will be offered by these RWCs, while maximizing the impact of the new RWCs in terms of the patient volume served and health service effectiveness. This formulation has proved to significantly improve the location decisions taken by North Star.

The paper focuses on the problem of selecting locations for a given number of RWCs, without considering the decisions on the health services to offer.

4.3.1 Problem formulation

The paper proposes a set partitioning type of problem formulation. In contrast with the direct formulation, this formulation uses binary variables to indicate whether or not to establish a configuration of RWCs along a given route. The location problem can be formulated as follows:

$$\text{Maximize} \quad w_{PV} \sum_{k\in KP} d_k x_k + w_{CA} \sum_{q\in Q} \sum_{n\in N_q} f_q c_q^n y_q^n - w_{EQ} \sum_{q_1\in Q} \sum_{q_2\in Q,\ q_2>q_1} (\triangle^+_{q_1q_2} + \triangle^-_{q_1q_2}) f_{q_1} f_{q_2}$$

subject to:

$$\sum_{n\in N_q} y_q^n = 1,\ q \in Q$$

$$\sum_{k\in KP} x_k = p$$

$$x_k - \sum_{n\in N_q^k} y_q^n = 0,\ k \in KP,\ q \in Q$$

$$\sum_{n\in N_{q_1}} c_{q_1}^n y_{q_1}^n - \sum_{n\in N_{q_2}} c_{q_2}^n y_{q_2}^n = \triangle^+_{q_1q_2} - \triangle^-_{q_1q_2},\ q_1, q_2 \in Q,\ q_2 > q_1$$

$$x_k, y_q^n \in \{0,1\},\ k \in KP,\ q \in Q,\ n \in N_q$$

$$\triangle^+_{q_1q_2}, \triangle^-_{q_1q_2} \geq 0,\ q_1, q_2 \in Q,\ q_2 > q_1$$

Here, $\triangle^+_{q_1q_2} = \max\{c_{q_1} - c_{q_2}, 0\}$ and $\triangle^-_{q_1q_2} = \max\{-(c_{q_1} - c_{q_2}), 0\}$ so that $|c_{q_1} - c_{q_2}| = \triangle^+_{q_1q_2} + \triangle^-_{q_1q_2}$. The objective function maximizes a weighted sum of the patient volume, the continuous access score and the equity score. The notations used in the set partitioning formulation are:

Sets	Description
KP	Set of potential locations
Q	Set of routes
N_q	Set of possible configurations of RWCs along route q

N_q^k	Subset of configurations os RWCs along route q with an RWC located at potential RWC location k
Decision variables	Description
y_q^n	Binary variable to indicate whether or not configuration $n \in N_q$ is chosen for route q
x_k	Binary variable to indicate whether or not to locate an RWC at potential location k
$\triangle^+_{q_1q_2}$, $\triangle^-_{q_1q_2}$	Continuous variables used to model the absolute difference between the coverage scores of routes q_1 and q_2
Parameters	Description
$\{w_{PV}, w_{CA}, w_{EQ}\}$	Weights of the objective functions terms
d_k	Expected daily number of patients entering an RWC at potential location k
f_q	The number of truck drivers traveling route q
c_q^n	Coverage score of route q given that $n \in N_q$ is realized (see [10])

4.3.2 Column generation

This formulation has an exponential number of variables, therefore it is proposed a column generation approach to solve it. Moreover, there are several strategies to speed up the algorithm, including dual stabilization, column pool management, and accelerated pricing. The latter solves the pricing problem to near-optimality as a sequence of shortest path problems.

Column generation is a suitable technique for solving integer linear programming problems (ILPs) that involve a large number of variables. The method starts solving the relaxation problem for a small subset of variables (the RMP), and then uses the resulting dual variables as an input to identify excluded variables that may improve the solution value. Then these variables are added to the set of included variables. The problem to find these variables is called the Pricing Problem. This process repeats till optimality is proven.

Since the set partitioning formulation requires an exponential number of variables, a column generation approach to solve its LPM relaxation is developed in the paper. The foregoing formulation has a very tight LPM relaxation. Therefore, instead of embedding the column generation approach in a computationally expensive Branch-and-Bound scheme, it is proposed a heuristic that directly constructs a feasible integer solution from the LPM relaxation solution.

Bibliography

[1] M.S. BAZARAA, H.D. SHERALI, C.M. SHETTY. *Nonlinear Programming*. Wiley 3^{rd} edition, Hoboken, 2006.

[2] M.S. BAZARAA, J.J. JARVIS, H.D. SHERALI. *Linear Programming and Network Flows*. Wiley 2^{nd} edition, Hoboken, 2010.

[3] D. BERTSIMAS, J.N. TSITSIKLIS. *Introduction to Linear Optimization*. Athena Scientific, Belmont, 1997.

[4] T.C. HU, A.B. KAHNG. *Linear and Integer Programming Made Easy*. Springer, Switzerland, 2016.

[5] K.G. MURTY. *Linear and Combinatorial Programming*. Wiley, New York, 1976.

[6] D.S. WATKINS *Fundamentals of Matrix Computations*. Wiley, New York, 2002.

[7] W.L. WINSTON. *Operations Research. Applications and Algorithms*. Thomson Brooks/Cole 4^{th} edition, Australia, 2004.

[8] J. KINABLE, F.C.R. SPIEKSMA, G. VANDEN BERGHE (2014) *School bus routing-a column generation approach*. International Transactions in Operational Research 21, 453-478.

[9] M.E. LÜBBECKE, J. DESROSIERS (2005) *Selected topics in column generation*. Operations Research 53, 1007-1023.

[10] J. NÚÑEZ ARES, H. DE VRIES, D. HUISMAN (2016) *A column generation approach for locating roadside clinics in Africa based on effectiveness and equity*. European Journal of Operational Research 254, 1002-1016.

[11] P. SCHITTEKAT, J.KINABLE, K. SÖRENSEN, M. SEVAUX, F. SPIEKSMA, J. SPRINGAEL (2013) *A metaheuristic for the school bus routing problem with bus stop selection*. European Journal of Operational Research 229, 518-528.

[12] H. DE VRIES, J.J. VAN DE KLUNDERT, A.P.M. WAGELMANS (2014) *The roadside healthcare facility location problem*. Econometric Institute Research Papers. (No. EI 2014-09).

Appendices

Appendix 1

Numerical examples

1.1 Numerical example of the revised simplex method

Consider the problem:

$$\begin{array}{ll} \text{Maximize} & Z = 3x_1 + x_2 + x_3 \\ \text{subject to:} & \\ & x_1 + x_2 + x_3 \leq 6 \\ & 2x_1 - x_3 \leq 4 \\ & x_2 + x_3 \leq 2 \\ & x_1, x_2, x_3 \geq 0 \end{array}$$

Use the product form of the inverse to perform the revised simplex method.

First, we convert the problem to standard form by adding the slack variables s_1, s_2 and s_3, obtaining the following problem:

$$\begin{array}{ll} \text{Maximize} & Z = 3x_1 + x_2 + x_3 \\ \text{subject to:} & \\ & x_1 + x_2 + x_3 + s_1 = 6 \\ & 2x_1 - x_3 + s_2 = 4 \\ & x_2 + x_3 + s_3 = 2 \\ & x_1, x_2, x_3, s_1, s_2, s_3 \geq 0 \end{array}$$

Initial basis $\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ having s_1, s_2 and s_3 as basic variables.

Non-basic variables: x_1, x_2 and x_3. We let $\mathbf{B}_i$ be the columns in the original linear program (LP) that correspond to the basic variables for tableau i. Then

$$\mathbf{B}_0^{-1} = \mathbf{B}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We can now determine which nonbasic variable should enter the basis by computing the reduced cost of each nonbasic variable.

$$\mathbf{c_B}\mathbf{B}_0^{-1} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

Reduced costs:

$x_1 \rightarrow c_1 - z_1 = c_1 - \mathbf{c_B}\mathbf{B}_0^{-1}\mathbf{A}_1 = c_1 = 3 > 0$

$x_2 \rightarrow c_2 - z_2 = c_2 - \mathbf{c_B}\mathbf{B}_0^{-1}\mathbf{A}_2 = c_2 = 1 > 0$

$x_3 \rightarrow c_3 - z_3 = c_3 - \mathbf{c_B}\mathbf{B}_0^{-1}\mathbf{A}_3 = c_3 = 1 > 0$

$\max\{3, 1, 1\} = 3 \Rightarrow x_1$ enters the basis. To continue, all we need to know is the new set of basic

variables, and the corresponding $\mathbf{B}_1^{-1}$. To determine the row in which x_1 enters the basis we compute x_1's column in the current tableau $\mathbf{B}_0^{-1}\mathbf{A}_1$ and the right-hand side of the current tableau $\mathbf{B}_0^{-1}\mathbf{b}$.

$$\text{Column for } x_1 \text{ in current tableau} = \mathbf{B}_0^{-1}\mathbf{A}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

$$\text{Right-hand side of current tableau} = \mathbf{B}_0^{-1}\mathbf{b} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix}$$

We now use the ratio test to determine the row in which x_1 should enter:
$\min\left\{\frac{6}{1}, \frac{4}{2}\right\} = 2 \Rightarrow x_1$ enters the basis in row $2 \Rightarrow s_2$ leaves the basis. Now we compute $\mathbf{B}_1^{-1}$ using the product form of the inverse.

$$\mathbf{E}_0 = \begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{B}_1^{-1} = \mathbf{E}_0\mathbf{B}_0^{-1} = \begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The new basic variables are: s_1, x_1 and s_3, and the non-basic variables: s_2, x_2 and x_3.

We now repeat the main step.

Determine which nonbasic variable enters the basis by computing the reduced cost of each nonbasic variable.

$$\mathbf{c_B}\mathbf{B}_1^{-1} = \begin{pmatrix} 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & \frac{3}{2} & 0 \end{pmatrix}$$

Reduced costs:

$$s_2 \rightarrow \text{Reduced cost of } s_2 = 0 - \mathbf{c_B}\mathbf{B}_1^{-1}\mathbf{A}_5 = 0 - \begin{pmatrix} 0 & \frac{3}{2} & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = -\frac{3}{2} < 0$$

$$x_2 \rightarrow c_2 - z_2 = c_2 - \mathbf{c_B}\mathbf{B}_1^{-1}\mathbf{A}_2 = 1 - \begin{pmatrix} 0 & \frac{3}{2} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 1 > 0$$

$$x_3 \rightarrow c_3 - z_3 = c_3 - \mathbf{c_B}\mathbf{B}_1^{-1}\mathbf{A}_3 = 1 - \begin{pmatrix} 0 & \frac{3}{2} & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} = \frac{5}{2} > 0$$

$\max\left\{1, \frac{5}{2}\right\} = \frac{5}{2} \Rightarrow x_3$ enters the basis. We need to know the new set of basic variables, and the corresponding $\mathbf{B}_2^{-1}$. To determine the row in which x_3 enters the basis we compute x_3's column in the current tableau $\mathbf{B}_1^{-1}\mathbf{A}_3$ and the right-hand side of the current tableau $\mathbf{B}_1^{-1}\mathbf{b}$.

$$\text{Column for } x_3 \text{ in current tableau} = \mathbf{B}_1^{-1}\mathbf{A}_3 = \begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ -\frac{1}{2} \\ 1 \end{pmatrix}$$

$$\text{Right-hand side of current tableau} = \mathbf{B}_1^{-1}\mathbf{b} = \begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 2 \end{pmatrix}$$

We now use the ratio test to determine the row in which x_3 should enter:

$\min\left\{\frac{4}{\frac{3}{2}}, \frac{2}{1}\right\} = 2 \Rightarrow x_3$ enters the basis in row 3$\Rightarrow s_3$ leaves the basis. Now we compute $\mathbf{B}_2^{-1}$ using the product form of the inverse.

$$\mathbf{E}_1 = \begin{pmatrix} 1 & 0 & -\frac{3}{2} \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{B}_2^{-1} = \mathbf{E}_1\mathbf{E}_0\mathbf{B}_0^{-1} = \mathbf{E}_1\mathbf{B}_1^{-1} = \begin{pmatrix} 1 & 0 & -\frac{3}{2} \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{3}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

The new basic variables are: s_1, x_1 and x_3, and the non-basic variables: s_2, x_2 and s_3. We now repeat the main step.

Determine which nonbasic variable enters the basis by computing the reduced cost of each nonbasic variable.

$$\mathbf{c_B}\mathbf{B}_2^{-1} = \begin{pmatrix} 0 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{3}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & \frac{3}{2} & \frac{5}{2} \end{pmatrix}$$

Reduced costs:

$$s_2 \rightarrow \text{Reduced cost of } s_2 = 0 - \mathbf{c_B}\mathbf{B}_2^{-1}\mathbf{A}_5 = 0 - \begin{pmatrix} 0 & \frac{3}{2} & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = -\frac{3}{2} < 0$$

$$x_2 \rightarrow c_2 - z_2 = c_2 - \mathbf{c_B}\mathbf{B}_2^{-1}\mathbf{A}_2 = 1 - \begin{pmatrix} 0 & \frac{3}{2} & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = -\frac{3}{2} < 0$$

$$s_3 \rightarrow \text{Reduced cost of } s_3 = c_3 - \mathbf{c_B}\mathbf{B}_2^{-1}\mathbf{A}_6 = 1 - \begin{pmatrix} 0 & \frac{3}{2} & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = -\frac{5}{2} < 0$$

All the reduced costs are $\leq 0 \Rightarrow$ The current bfs is optimal.

$$Z = \mathbf{c_B}\mathbf{B}_2^{-1}\mathbf{b} = \begin{pmatrix} 0 & \frac{3}{2} & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix} = 11$$

$$\text{Right-hand side of current tableau} = \begin{pmatrix} s_1 \\ x_1 \\ x_3 \end{pmatrix} = \mathbf{B}_2^{-1}\mathbf{b} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{3}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$$

Therefore, the optimal solution is: $x_1 = 3, x_2 = 0, x_3 = 2, Z = 11$.

1.2 Numerical example of the simplex method for bounded variables

Consider the problem:

$$\begin{array}{ll} \text{Maximize} & 3x_1+5x_2+2x_3 \\ \text{subject to:} & \\ & x_1+2x_2+2x_3+x_4=10 \\ & 2x_1+4x_2+3x_3+x_5=15 \\ & 0\le x_1\le 4 \\ & 0\le x_2\le 2 \\ & 2\le x_3\le 4 \\ & 0\le x_4\le 8 \\ & 0\le x_5\le 10 \end{array}$$

Initial basis $\boldsymbol{B}=\begin{pmatrix}1&0\\0&1\end{pmatrix}$ having x_4 and x_5 as basic variables.
Non-basic variables:
$x_1=l_1=0$;
$x_2=l_2=0$;
$x_3=l_3=2$
$\Rightarrow 0\le x_4=10-x_1-2x_2-2x_3=6\le 8$
$\Rightarrow 0\le x_5=15-2x_1-4x_2-3x_3=9\le 10$
$\Rightarrow$ bfs

The initial tableau is:

Max	$x_1=l_1=0$	$x_2=l_2=0$	$x_3=l_3=2$	x_4	x_5	$\overline{\boldsymbol{b}}$
x_4	1	2	2	1	0	6
x_5	2	4	3	0	1	9
	3	5	2	0	0	

Candidates:
$x_1\uparrow$, $c_1-z_1=3>0\rightsquigarrow$ Yes
$x_2\uparrow$, $c_2-z_2=5>0\rightsquigarrow$ Yes
$x_3\uparrow$, $c_3-z_3=2>0\rightsquigarrow$ Yes
$\Rightarrow$ Among the candidates, x_2 is the variable with the greatest reduced cost.
$x_2\uparrow$: $x_2=0+\triangle_2$, $\triangle_2\ge 0$
$0\le 0+\triangle_2\le 2\rightsquigarrow$ $\triangle_2\le 2$

$$\begin{pmatrix}0\\0\end{pmatrix}\le\begin{pmatrix}x_4\\x_5\end{pmatrix}=\begin{pmatrix}6\\9\end{pmatrix}-\triangle_2\begin{pmatrix}2\\4\end{pmatrix}=\begin{pmatrix}6-2\,\triangle_2\\9-4\,\triangle_2\end{pmatrix}\le\begin{pmatrix}8\\10\end{pmatrix}$$

$6-2\,\triangle_2\ge 0\rightsquigarrow\triangle_2\le 3$
$9-4\,\triangle_2\ge 0\rightsquigarrow\triangle_2\le\frac{9}{4}$
$\Rightarrow\triangle_2=\min\left\{2,3,\frac{9}{4}\right\}=2$
$\Rightarrow x_2$ is still non-basic, with value its upper bound. The new tableau is equal to the previous one except for the basic and non-basic variables values.

Max	$x_1 = l_1 = 0$	$x_2 = u_2 = 2$	$x_3 = l_3 = 2$	x_4	x_5	$\overline{b}$
x_4	1	2	2	1	0	2
x_5	②	4	3	0	1	1
	3	5	2	0	0	

$\longrightarrow$

$\uparrow$

Candidates:
$x_1 \uparrow,\ c_1 - z_1 = 3 > 0 \rightsquigarrow$ Yes
$x_2 \downarrow,\ c_2 - z_2 = 5 > 0 \rightsquigarrow$ No
$x_3 \uparrow,\ c_3 - z_3 = 2 > 0 \rightsquigarrow$ Yes
$\Rightarrow$ Among the candidates, x_1 is the variable with the greatest reduced cost.
$x_1 \uparrow$: $x_1 = l_1 + \triangle_1, \quad \triangle_1 \geq 0$
$0 \leq 0 + \triangle_1 \leq 4 \rightsquigarrow \quad \triangle_1 \leq 4$

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} - \triangle_1 \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 - \triangle_1 \\ 1 - 2\,\triangle_1 \end{pmatrix} \leq \begin{pmatrix} 8 \\ 10 \end{pmatrix}$$

$2 - \triangle_1 \geq 0 \rightsquigarrow \triangle_1 \leq 2$
$1 - 2\,\triangle_1 \geq 0 \rightsquigarrow \triangle_1 \leq \frac{1}{2}$
$\Rightarrow \triangle_1 = \min\{4, 2, \frac{1}{2}\} = \frac{1}{2}$
$\Rightarrow x_1$ enters the basis and x_5 leaves the basis with value 0, that is, its lower-bound. The new tableau is:

Max	x_1	$x_2 = u_2 = 2$	$x_3 = l_3 = 2$	x_4	$x_5 = l_5 = 0$	$\overline{b}$
x_4	0	0	$\frac{1}{2}$	1	$-\frac{1}{2}$	$\frac{3}{2}$
x_1	1	②	$\frac{3}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
	0	-1	$-\frac{5}{2}$	0	$-\frac{3}{2}$	

$\longrightarrow$

$\uparrow$

Candidates:
$x_2 \downarrow,\ c_2 - z_2 = -1 < 0 \rightsquigarrow$ Yes
$x_3 \uparrow,\ c_3 - z_3 = -\frac{5}{2} < 0 \rightsquigarrow$ No
$x_5 \uparrow,\ c_5 - z_5 = -\frac{3}{2} < 0 \rightsquigarrow$ No
$\Rightarrow x_2$ is selected.
$x_2 \downarrow$: $x_2 = u_2 - \triangle_2, \quad \triangle_2 \geq 0$
$0 \leq 0 - \triangle_2 \leq 2 \rightsquigarrow \quad \triangle_2 \leq 2$

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} x_4 \\ x_1 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ \frac{1}{2} \end{pmatrix} + \triangle_2 \begin{pmatrix} 0 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ \frac{1}{2} + 2\,\triangle_2 \end{pmatrix} \leq \begin{pmatrix} 8 \\ 4 \end{pmatrix}$$

$\frac{1}{2} + 2\,\triangle_2 \leq 4 \rightsquigarrow \triangle_2 \leq \frac{7}{4}$
$\Rightarrow \triangle_2 = \min\{2, \frac{7}{4}\} = \frac{7}{4}$
$\Rightarrow x_2$ enters the basis and x_1 leaves the basis with value 4, that is, its upper-bound.

Max	$x_1 = u_1 = 4$	x_2	$x_3 = l_3 = 2$	x_4	$x_5 = l_5 = 0$	$\overline{b}$
x_4	0	0	$\frac{1}{2}$	1	$-\frac{1}{2}$	$\frac{3}{2}$
x_2	$\frac{1}{2}$	1	$\frac{3}{4}$	0	$\frac{1}{4}$	$\frac{1}{4}$
	$\frac{1}{2}$	0	$-\frac{7}{4}$	0	$-\frac{5}{4}$	

Candidates:

$x_1 \downarrow,\ c_1 - z_1 = \frac{1}{2} > 0 \rightsquigarrow$ No

$x_3 \uparrow,\ c_3 - z_3 = -\frac{7}{4} < 0 \rightsquigarrow$ No

$x_5 \uparrow,\ c_5 - z_5 = -\frac{5}{4} < 0 \rightsquigarrow$ No

$\Rightarrow$ Optimal solution

$x_1 = 4,\ x_2 = \frac{1}{4},\ x_3 = 2,\ x_4 = \frac{3}{2},\ x_5 = 0$

$z = 3x_1 + 5x_2 + 2x_3 = 12 + \frac{5}{4} + 4 = \frac{69}{4}$

1.3 Numerical example of the decomposition principle

Consider the problem:

$$\begin{aligned} \text{Maximize} \quad & Z = 7x_1 + 5x_2 + 3x_3 \\ \text{subject to:} \quad & \\ & x_1 + 2x_2 + x_3 && \leq 10 \\ & x_1 && \leq 3 \\ & x_2 + x_3 && \leq 5 \\ & 2x_2 + x_3 && \leq 8 \\ & x_i \geq 0, && i = 1,2,3 \end{aligned}$$

$X_1 = \{x_1 : 0 \leq x_1 \leq 3\}$

$X_2 = \{(x_2, x_3) : x_2 + x_3 \leq 5\ ;\ 2x_2 + x_3 \leq 8\ ;\ x_2, x_3 \geq 0\}$

X_1 is the closed interval [0 , 3] and X_2 is a bounded polyhedron with extreme points $(0,0)$, $(4,0)$, $(0,5)$ and $(3,2)$. Both regions are bounded, therefore they do not have any extreme directions. Hence, the problem is equivalent to:

$$\text{Maximize} \ \sum_{j=1}^{k_1} (\mathbf{c}_1 \mathbf{x}_{1j}) \lambda_{1j} + \sum_{j=1}^{k_2} (\mathbf{c}_2 \mathbf{x}_{2j}) \lambda_{2j}$$

subject to:

$$\sum_{j=1}^{k_1} (\mathbf{A}_1 \mathbf{x}_{1j}) \lambda_{1j} + \sum_{j=1}^{k_2} (\mathbf{A}_2 \mathbf{x}_{2j}) \lambda_{2j} \leq 10$$

$$\sum_{j=1}^{k_1} \lambda_{1j} = 1$$

$$\sum_{j=1}^{k_2} \lambda_{2j} = 1$$

$$\lambda_{1j} \geq 0,\ j = 1,2,\ldots,k_1$$

$$\lambda_{2j} \geq 0,\ j = 1,2,\ldots,k_2$$

where $\mathbf{c}_1 = (\ 7\ )$, $\mathbf{c}_2 = (\ 5\ ,\ 3\ )$, $\mathbf{A}_1 = (\ 1\ )$, $\mathbf{A}_2 = (\ 2\ ,\ 1\ )$ and $\mathbf{x}_{1j}$, $\mathbf{x}_{2j}$ are the extreme points of X_1 and X_2 respectively.

We consider the intial bfs formed by the slack variable of the first constraint x_4 and the artificial variables y_1 and y_2 of the second and third constraints.
$(x_4, y_1, y_2) = (10, 1, 1)$
$\mathbf{B}^{-1} = \mathbf{I}_{3\times 3}$
$(\mathbf{w}, \alpha) = \hat{\mathbf{c}}_\mathbf{B}\mathbf{B}^{-1} = (0, -M, -M)\mathbf{I}_{3\times 3} = (0, -M, -M) \Rightarrow \mathbf{w}_1 = 0\ ;\ \alpha_1 = -M\ ;\ \alpha_2 = -M$

Iteration 1:

Subproblem 1

$$\begin{aligned} &\text{Maximize} && [7 - 0\cdot 1]x_1 - (-M) = 7x_1 + M \\ &\text{subject to:} && \\ & && x_1 \in X_1 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{11} = x_1 = 3$, at this point the value of the objective function is $21 + M > 0 \Rightarrow \lambda_{11}$ is a candidate to enter the basis.

Subproblem 2

$$\begin{aligned} &\text{Maximize} && [(5,3) - 0\cdot(2,1)] \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} - (-M) = 5x_2 + 3x_3 + M \\ &\text{subject to:} && \\ & && (x_2, x_3) \in X_2 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{21} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$, at this point the value of the objective function is $21 + M > 0 \Rightarrow \lambda_{21}$ is a candidate to enter the basis.
Since both candidates have the same reduced cost, we select λ_{11}. Its transformed column vector is:

$$\mathbf{B}^{-1}\begin{pmatrix} 1\cdot 3 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 0 \end{pmatrix}$$

$\min\left\{\frac{10}{3}, \frac{1}{1}\right\} = 1 \Rightarrow$ the artificial variable y_1 leaves the basis.
The tableau with the entering variable is:

Max	x_4	y_1	y_2	$\overline{b}$	λ_{11}	
x_4	1	0	0	10	3	
y_1	0	1	0	1	(1)	$\longrightarrow$
y_2	0	0	1	1	0	
-Z	0	0	0	2M	21+M	
					$\uparrow$	

The new tableau is:

Max	x_4	y_1	y_2	$\bar{b}$	λ_{22}
x_4	1	-3	0	7	(8)
λ_{11}	0	1	0	1	0
y_2	0	0	1	1	1
-Z	0	-21-M	0	-21+M	21+M

$\longrightarrow$

$\uparrow$

$w_1 = 0$, $\alpha_1 = 21 + M - M = 21$, $\alpha_2 = -M + 0 = -M$

In this tableau the variable λ_{22} has been added in red because, as we will see below, it is the entering variable.

Iteration 2:

Subproblem 1

$$\begin{array}{ll} \text{Maximize} & [7 - 0 \cdot 1]x_1 - 21 = 7x_1 - 21 \\ \text{subject to:} & \\ & x_1 \in X_1 \end{array}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{12} = 3$, at this point the value of the objective function is $21 - 21 = 0 \Rightarrow$ There exists no candidate to enter the basis.

Subproblem 2

$$\begin{array}{ll} \text{Maximize} & [(5,3) - 0 \cdot (2,1)] \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} - (-M) = 5x_2 + 3x_3 + M \\ \text{subject to:} & \\ & (x_2, x_3) \in X_2 \end{array}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{22} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$, at this point the value of the objective function is $21 + M > 0 \Rightarrow \lambda_{22}$ is a candidate to enter the basis.

$\Rightarrow \lambda_{22}$ enters the basis. Its transformed column vector is:

$$\mathbf{B}^{-1} \begin{pmatrix} \begin{pmatrix} 2 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & -3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 8 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 0 \\ 1 \end{pmatrix}$$

(As indicated, in the preceding table we may see in red the variable that enters and the variable that exits the basis).

$\min \left\{ \frac{7}{8}, \frac{1}{1} \right\} = \frac{7}{8} \Rightarrow x_4$ leaves the basis. The new tableau is:

Max	x_4	y_1	y_2	$\bar{b}$	λ_{23}
λ_{22}	$\frac{1}{8}$	$-\frac{3}{8}$	0	$\frac{7}{8}$	0
λ_{11}	0	1	0	1	0
y_2	$-\frac{1}{8}$	$\frac{3}{8}$	1	$\frac{1}{8}$	(1)
-Z	$\frac{-21-M}{8}$	$-\frac{105}{8}-\frac{5}{8}M$	0	$-\frac{315}{8}+\frac{M}{8}$	M

$\longrightarrow$

$\uparrow$

$w_1 = \frac{21+M}{8}$, $\alpha_1 = -M + \frac{105}{8} + \frac{5}{8}M = \frac{105}{8} - \frac{3}{8}M$, $\alpha_2 = -M$

Iteration 3:

(x_4 is not a candidate)

Subproblem 1

$$\begin{aligned} &\text{Maximize} && \left[7 - \frac{21+M}{8}\cdot 1\right] x_1 - \left(\frac{105}{8} - \frac{3}{8}M\right) = \left(\frac{35-M}{8}\right) x_1 - \frac{105}{8} + \frac{3}{8}M \\ &\text{subject to:} \\ &&& x_1 \in X_1 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{13} = x_1 = 0$, at this point the value of the objective function is $-\frac{105}{8} + \frac{3}{8}M > 0 \Rightarrow \lambda_{13}$ is a candidate to enter the basis.

Subproblem 2

$$\begin{aligned} &\text{Maximize} && [(5,3) - \frac{21+M}{8}\cdot(2,1)] \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} - (-M) = \frac{-1-M}{4}x_2 + \frac{3-M}{8}x_3 + M \\ &\text{subject to:} \\ &&& (x_2, x_3) \in X_2 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{23} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, at this point the value of the objective function is $M > 0 \Rightarrow \lambda_{23}$ is a candidate to enter the basis.
$\Rightarrow \lambda_{23}$ is the variable that has the greatest reduced cost, therefore it is the one that enters the basis. Its transformed column vector is:

$$\mathbf{B}^{-1}\begin{pmatrix} \begin{pmatrix} 2 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{8} & -\frac{3}{8} & 0 \\ 0 & 1 & 0 \\ -\frac{1}{8} & \frac{3}{8} & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

(In the immediately preceding table we may see in red the variable that enters and the variable that exits the basis).

$\min\left\{\frac{\frac{1}{8}}{1}\right\} = \frac{1}{8} \Rightarrow y_2$ leaves the basis. The new tableau is:

Max	x_4	y_1	y_2	$\overline{b}$	λ_{24}
λ_{22}	$\frac{1}{8}$	$-\frac{3}{8}$	0	$\frac{7}{8}$	$\frac{5}{8}$
λ_{11}	0	1	0	1	0
λ_{23}	$-\frac{1}{8}$	$\frac{3}{8}$	1	$\frac{1}{8}$	$\frac{3}{8}$
-Z	$\frac{-21}{8}$	$-\frac{105}{8}-M$	-M	$-\frac{315}{8}$	$\frac{15}{8}$

$\longrightarrow$

$\uparrow$

$w_1 = \frac{21}{8}$, $\alpha_1 = \frac{105}{8}$, $\alpha_2 = 0$

Iteration 4:

(x_4 is not a candidate)

Subproblem 1

$$\begin{aligned} &\text{Maximize} && [7-\frac{21}{8}\cdot 1]x_1 - \frac{105}{8} = \frac{35}{8}x_1 - \frac{105}{8} \\ &\text{subject to:} && \\ & && x_1 \in X_1 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{14} = 3$, at this point the value of the objective function is $\frac{105}{8} - \frac{105}{8} = 0 \Rightarrow$ There exists no candidate to enter the basis.

Subproblem 2

$$\begin{aligned} &\text{Maximize} && [(5,3) - \frac{21}{8}\cdot(2,1)]\begin{pmatrix} x_2 \\ x_3 \end{pmatrix} - 0 = -\frac{1}{4}x_2 + \frac{3}{8}x_3 \\ &\text{subject to:} && \\ & && (x_2, x_3) \in X_2 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{24} = \begin{pmatrix} 0 \\ 5 \end{pmatrix}$, at this point the value of the objective function is $\frac{15}{8} > 0 \Rightarrow \lambda_{24}$ is a candidate to enter the basis.
$\Rightarrow \lambda_{24}$ enters the basis. Its transformed column vector is:

$$\mathbf{B}^{-1}\begin{pmatrix} \begin{pmatrix} 2 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 5 \end{pmatrix} \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{8} & -\frac{3}{8} & 0 \\ 0 & 1 & 0 \\ -\frac{1}{8} & \frac{3}{8} & 1 \end{pmatrix}\begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{5}{8} \\ 0 \\ \frac{3}{8} \end{pmatrix}$$

(In the preceding table we may see in red the variable that enters and the variable that exits the basis).

$\min\left\{\frac{\frac{7}{8}}{\frac{5}{8}}, \frac{\frac{1}{8}}{\frac{3}{8}}\right\} = \frac{1}{3} \Rightarrow \lambda_{23}$ leaves the basis. (It entered the basis in the preceding iteration). The new tableau is:

Max	x_4	y_1	y_2	$\bar{b}$	
λ_{22}	$\frac{1}{3}$	-1	$-\frac{5}{3}$	$\frac{2}{3}$	
λ_{11}	0	1	0	1	
λ_{24}	$-\frac{1}{3}$	1	$\frac{8}{3}$	$\frac{1}{3}$	
-Z	-2	$-15-M$	$-5-M$	-40	

$w_1 = 2$, $\alpha_1 = 15$, $\alpha_2 = 5$

Iteration 5:

(x_4 is not a candidate)

Subproblem 1

$$\begin{aligned} &\text{Maximize} && [7-2\cdot 1]x_1 - 15 = 5x_1 - 15 \\ &\text{subject to:} && \\ & && x_1 \in X_1 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{15} = 3$, at this point the value of the objective function is 0 $\Rightarrow$ There exists no candidate to enter the basis.

Subproblem 2

$$\begin{aligned} &\text{Maximize} && [(5,3) - 2\cdot(2,1)] \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} - 5 = x_2 + x_3 - 5 \\ &\text{subject to:} && \\ & && (x_2, x_3) \in X_2 \end{aligned}$$

$\Rightarrow$ We reach the maximum at the extreme point $\mathbf{x}_{25} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$, at this point the value of the objective function is 0 $\Rightarrow$ There exists no candidate to enter the basis.
$\Rightarrow$ The optimal solution is:
$x_1 = \lambda_{11}\mathbf{x}_{11} = 3$
$\begin{pmatrix} x_2 \\ x_3 \end{pmatrix} = \lambda_{22}\mathbf{x}_{22} + \lambda_{24}\mathbf{x}_{24} = \frac{2}{3}\begin{pmatrix} 3 \\ 2 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 \\ 5 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$
$\Rightarrow x_1 = 3, x_2 = 2, x_3 = 3, Z = 40$

1.4 Numerical example of the cutting stock problem

We use the column generation technique to solve a cutting stock problem in which a costumer demands 20 3-ft boards, 25 4-ft boards, and 30 5-ft boards, and demand is met by cutting up 14-ft boards.

We must decide how each 14-ft boards should be cut. Hence, each decision corresponds to a way in which a 14-ft board can be cut. For example, one decision variable would correspond to a board being cut into four 3-ft boards, which would incur waste of $14 - 4\cdot 3 = 2$ ft. Many possible ways of cutting a board need not be considered. For example, it would be foolish to cut a board into two 5-ft pieces; we could just as easily cut the board into two 5-ft pieces, and a 4-ft piece. In general, any cutting pattern that leaves 3-ft ore more waste need not be considered because we could use the waste to obtain one or more 3-ft boards. The following table lists the sensible ways to cut a 14-ft board.

Combination	Number of 3-ft boards	Number of 4-ft boards	Number of 5-ft boards	Waste
1	0	1	2	0
2	0	2	1	1
3	1	1	1	2
4	3	0	1	0
5	3	1	0	1
6	0	3	0	2
7	2	2	0	0
8	1	0	2	1
9	4	0	0	2

We now define

$$x_i = \text{Number of 14-ft boards cut according to combination } i$$

and formulate the LP:

$$\textit{Waste} + \textit{Total costumer demand} = \textit{Total length of board cut}$$

Since

$$\textit{Total costumer demand} = 20 \cdot 3 + 25 \cdot 4 + 30 \cdot 5 = 310 \text{ ft}$$

We write

$$\textit{Waste (in feet)} = 14x_1 + 14x_2 + 14x_3 + 14x_4 + 14x_5 + 14x_6 + 14x_7 + 14x_8 + 14x_9$$

Then the objective function is to minimize

$$\text{Minimize } 14x_1 + 14x_2 + 14x_3 + 14x_4 + 14x_5 + 14x_6 + 14x_7 + 14x_8 + 14x_9$$

This is equivalent to minimizing

$$14(x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9)$$

which is equivalent to minimizing

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9$$

Hence the objective function is:

$$\text{Minimize } Z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9$$

This means that we can minimize the total waste by minimizing the numbers of 14-ft boards that are cut.

The constraints are:

- Constraint 1: At least 20 3-ft boards mus be cut
- Constraint 2: At least 25 4-ft boards mus be cut

- Constraint 3: At least 30 5-ft boards mus be cut

Since the total number of 3-ft boards that are cut is given by $x_3 + 3x_4 + 3x_5 + 2x_7 + x_8 + 4x_9$, Constraint 1 becomes $x_3 + 3x_4 + 3x_5 + 2x_7 + x_8 + 4x_9 \geq 20$.

Similarly, Constraint 2 becomes $x_1 + 2x_2 + x_3 + x_5 + 3x_6 + 2x_7 \geq 25$, and Constraint 3 becomes $2x_1 + x_2 + x_3 + x_4 + 2x_8 \geq 30$.

Note that the coefficient of x_i in the constraint is just the number of k-ft boards yielded if a board is cut according to combination i. It is clear that the x_i should be required to assume integer values. Despite this fact, in problems with large demands, a near-optimal solution can be obtained by solving the cutting stock problem as an LP and then rounding all fractional variables upward. This procedure may not yield the best possible integer solution, but it usually yields a near-optimal integer solution. For this reason, we concentrate on the LP version of the cutting stock problem. We obtain the following LP:

$$\begin{aligned} \text{Minimize} \quad & Z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 \\ \text{subject to:} \quad & \\ & x_3 + 3x_4 + 3x_5 + 2x_7 + x_8 + 4x_9 \geq 20 \\ & x_1 + 2x_2 + x_3 + x_5 + 3x_6 + 2x_7 \geq 25 \\ & 2x_1 + x_2 + x_3 + x_4 + 2x_8 \geq 30 \\ & x_i \geq 0,\ i = 1, \ldots, 9 \end{aligned}$$

Note that x_6 occurs only in the 4-ft constraint and x_9 only in the 3-ft constraint, this means that x_6 and x_9 can be used as starting basic variables for the 4-ft and 3-ft constraints.

To avoid having to add an artificial variable to the 5-ft constraint we define combination 10 to be the cutting combination that yields only one 5-ft board. Also, define x_{10} to be the number of boards cut according to combination 10. Clearly, x_{10} will be equal to zero in the optimal solution, but inserting x_{10} in the starting basis allows us to avoid using the Big M or the two-phase simplex method. Note that the column for x_{10} in the LP constraints will be $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ and a term $x_1 0$ will be added to the objective function. Now the starting basis has as basic variables x_9, x_6 and x_{10}.

If we let the tableau for this basis be tableau 0, then we have:

$$\mathbf{B}_0 = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{B}_0^{-1} = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Then

$$\mathbf{c_B B}_0^{-1} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & \frac{1}{3} & 1 \end{pmatrix}$$

If we now computed the reduced cost of each nonbasic variable it would tell us which variable should enter the basis. However, in a large-scale cutting stock problem, there may be thousands of variables, so this would be an extremely tedious chore. This is the kind of situation in which column generation comes into play.

In the cutting stock problem, each column, or variable, represents a combination for cutting up a board: A variable is specified by three numbers: a_3, a_4 and a_5, where a_i is the number of i-ft boards yielded by cutting one 14-ft board according to the given combination. For example, the variable x_1 is specified by $a_3 = 0$, $a_4 = 1$ and $a_5 = 2$. The idea of column generation is to search efficiently for a column that has a negative reduced cost (since we are solving a minimization problem). For our current basis, a combination specified by a_3, a_4 and a_5 has a reduced cost

$$1 - \mathbf{c_B}\mathbf{B}_0^{-1}\begin{pmatrix} a_3 \\ a_4 \\ a_5 \end{pmatrix} = 1 - \frac{1}{4}a_3 - \frac{1}{3}a_4 - a_5$$

Note that a_3, a_4 and a_5 must be chosen so they do not use more than 14-ft of wood. We also know that a_3, a_4 and a_5 must be nonnegative integers. We can now find the combination that minimizes the reduced cost by solving the following knapsack problem:

$$\begin{aligned} \text{Maximize} \quad & Z = \frac{1}{4}a_3 + \frac{1}{3}a_4 + a_5 - 1 \\ \text{subject to:} \quad & \\ & 3a_3 + 4a_4 + 5a_5 \leq 14 \\ & a_3, a_4, a_5 \geq 0;\ a_3, a_4, a_5 \text{ integer} \end{aligned}$$

Note that the constant 1 can be omitted. We use LINGO to solve this integer problem

```
max =(1/4)*a3+(1/3)*a4+a5;
[R1] 3*a3+4*a4+5*a5<14;
@GIN(a3);
@GIN(a4);
@GIN(a5);
```

obtaining the optimal solution

$$Z = \frac{4}{3},\ a_3 = 0,\ a_4 = 1 \text{ and } a_5 = 2$$

This corresponds to combination 1 and variable x_1. Hence, x_1 prices out $\frac{4}{3}$, and entering x_1 into the basis will decrease the waste. To enter x_1 into the basis, we create the right-hand side of the current tableau and the x_1 column of the current tableau.

$$\text{Column for } x_1 \text{ in current tableau} = \mathbf{B}_0^{-1}\mathbf{A}_1 = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{3} \\ 2 \end{pmatrix}$$

$$\text{Right-hand side of current tableau} = \mathbf{B}_0^{-1}\mathbf{b} = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 20 \\ 25 \\ 30 \end{pmatrix} = \begin{pmatrix} 5 \\ \frac{25}{3} \\ 30 \end{pmatrix}$$

We use the ratio test to determine which variable leaves the basis: $\min\left\{\dfrac{\frac{25}{3}}{\frac{1}{3}}, \dfrac{30}{2}\right\} = \min\{25, 15\} =$ $15 \Rightarrow x_{10}$ leaves the basis, that is, x_1 enters the basis in row 3. Now we compute $\mathbf{B}_1^{-1}$ using the product form of the inverse.

$$\mathbf{E}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

$$\mathbf{B}_1^{-1} = \mathbf{E}_0\mathbf{B}_0^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} \end{pmatrix}\begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

The new basic variables are: x_9, x_6 and x_1. Now we repeat the procedure.

$$\mathbf{c_B B}_1^{-1} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

With our new set of dual variables values we can again use column generation to determine whether there is a combination that should be entered into the basis. Similarly as above:

$$1 - \mathbf{c_B B}_1^{-1} \begin{pmatrix} a_3 \\ a_4 \\ a_5 \end{pmatrix} = 1 - \frac{1}{4}a_3 - \frac{1}{3}a_4 - \frac{1}{3}a_5$$

For the current tableau, the column generation procedure yields the following knapsack problem:

$$\begin{aligned} \text{Maximize} \quad & Z = \frac{1}{4}a_3 + \frac{1}{3}a_4 + \frac{1}{3}a_5 - 1 \\ \text{subject to:} \quad & \\ & 3a_3 + 4a_4 + 5a_5 \leq 14 \\ & a_3,\, a_4,\, a_5 \geq 0;\; a_3,\, a_4,\, a_5 \text{ integer} \end{aligned}$$

We use LINGO to solve this integer problem

```
max =(1/4)*a3+(1/3)*a4+(1/3)*a5;
[R1] 3*a3+4*a4+5*a5<14;
@GIN(a3);
@GIN(a4);
@GIN(a5);
```

obtaining the optimal solution

$$Z = \frac{5}{3},\; a_3 = 2,\; a_4 = 2 \text{ and } a_5 = 0$$

This corresponds to combination 7 and variable x_7. Hence, x_7 prices out $\frac{5}{3}$, and entering x_7 into the basis will decrease the waste. To enter x_7 into the basis, we create the right-hand side of the current tableau and the x_7 column of the current tableau.

$$\text{Column for } x_7 \text{ in current tableau} = \mathbf{B}_1^{-1}\mathbf{A}_7 = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{2}{3} \\ 0 \end{pmatrix}$$

$$\text{Right-hand side of current tableau} = \mathbf{B}_1^{-1}\mathbf{b} = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 20 \\ 25 \\ 30 \end{pmatrix} = \begin{pmatrix} 5 \\ \frac{10}{3} \\ 15 \end{pmatrix}$$

We use the ratio test to determine which variable leaves the basis: $\min\left\{\frac{5}{\frac{1}{2}}, \frac{\frac{10}{3}}{\frac{2}{3}}\right\} = \min\{10, 5\} =$ $5 \Rightarrow x_6$ leaves the basis, that is, x_7 enters the basis in row 2. Now we compute $\mathbf{B}_2^{-1}$ using the product form of the inverse.

$$\mathbf{E}_1 = \begin{pmatrix} 1 & -\frac{3}{4} & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{B}_2^{-1} = \mathbf{E}_1\mathbf{B}_1^{-1} = \begin{pmatrix} 1 & -\frac{3}{4} & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{1}{6} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{8} \\ 0 & \frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

The new basic variables are: x_9, x_7 and x_1. Now we repeat the procedure.

$$\mathbf{c_B}\mathbf{B}_2^{-1} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{8} \\ 0 & \frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{3}{8} \end{pmatrix}$$

For this set of dual variables values we can again use column generation to determine whether there is a combination that should be entered into the basis. Similarly as above we need to minimize $1 - \frac{1}{4}a_3 - \frac{1}{4}a_4 - \frac{3}{8}a_5$. Thus, the column generation procedure requires us to solve the following knapsack problem:

$$\begin{aligned} \text{Maximize} \quad & Z = \frac{1}{4}a_3 + \frac{1}{4}a_4 + \frac{3}{8}a_5 - 1 \\ \text{subject to:} \quad & \\ & 3a_3 + 4a_4 + 5a_5 \leq 14 \\ & a_3, a_4, a_5 \geq 0;\ a_3, a_4, a_5 \text{ integer} \end{aligned}$$

We use LINGO to solve this integer problem

```
max =(1/4)*a3+(1/4)*a4+(3/8)*a5;
[R1] 3*a3+4*a4+5*a5<14;
@GIN(a3);
@GIN(a4);
@GIN(a5);
```

obtaining the optimal solution

$$Z = 0.125,\ a_3 = 3,\ a_4 = 0 \text{ and } a_5 = 1$$

This corresponds to combination 4 and variable x_4. Hence, x_4 prices out 0.125, and entering x_4 into the basis will decrease the waste. To enter x_4 into the basis, we create the right-hand side of the current tableau and the x_4 column of the current tableau.

$$\text{Column for } x_4 \text{ in current tableau} = \mathbf{B}_2^{-1}\mathbf{A}_4 = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{8} \\ 0 & \frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{7}{8} \\ -\frac{1}{4} \\ \frac{1}{2} \end{pmatrix}$$

$$\text{Right-hand side of current tableau} = \mathbf{B}_2^{-1}\mathbf{b} = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{8} \\ 0 & \frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 20 \\ 25 \\ 30 \end{pmatrix} = \begin{pmatrix} \frac{5}{2} \\ 5 \\ 15 \end{pmatrix}$$

We use the ratio test to determine which variable leaves the basis: $\min\left\{\frac{\frac{5}{2}}{\frac{7}{8}}, \frac{15}{\frac{1}{2}}\right\} = \min\left\{\frac{20}{7}, 30\right\} = \frac{20}{7} \Rightarrow x_9$ leaves the basis, that is, x_4 enters the basis in row 1. Now we compute $\mathbf{B}_3^{-1}$ using the product form of the inverse.

$$\mathbf{E}_2 = \begin{pmatrix} \frac{8}{7} & 0 & 0 \\ \frac{2}{7} & 1 & 0 \\ -\frac{4}{7} & 0 & 1 \end{pmatrix}$$

$$\mathbf{B}_3^{-1} = \mathbf{E}_2\mathbf{B}_2^{-1} = \begin{pmatrix} \frac{8}{7} & 0 & 0 \\ \frac{2}{7} & 1 & 0 \\ -\frac{4}{7} & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{8} \\ 0 & \frac{1}{2} & -\frac{1}{4} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{2}{7} & -\frac{2}{7} & \frac{1}{7} \\ \frac{1}{14} & \frac{3}{7} & -\frac{3}{14} \\ -\frac{1}{7} & \frac{1}{7} & \frac{3}{7} \end{pmatrix}$$

The new basic variables are: x_4, x_7 and x_1. Now we repeat the procedure.

$$\mathbf{c_B}\mathbf{B}_3^{-1} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \frac{2}{7} & -\frac{2}{7} & \frac{1}{7} \\ \frac{1}{14} & \frac{3}{7} & -\frac{3}{14} \\ -\frac{1}{7} & \frac{1}{7} & \frac{3}{7} \end{pmatrix} = \begin{pmatrix} \frac{3}{14} & \frac{2}{7} & \frac{5}{14} \end{pmatrix}$$

For this set of dual variables values we can again use column generation to determine whether there is a combination that should be entered into the basis. Similarly as above we need to minimize $1 - \frac{3}{14}a_3 - \frac{2}{7}a_4 - \frac{5}{14}a_5$. Thus, the column generation procedure requires us to solve the following knapsack problem:

$$\begin{aligned} \text{Maximize} \quad & Z = \frac{3}{14}a_3 + \frac{2}{7}a_4 + \frac{5}{14}a_5 - 1 \\ \text{subject to:} \quad & \\ & 3a_3 + 4a_4 + 5a_5 \leq 14 \\ & a_3, a_4, a_5 \geq 0;\ a_3, a_4, a_5 \text{ integer} \end{aligned}$$

We use LINGO to solve this integer problem

```
max =(3/14)*a3+(2/7)*a4+(5/14)*a5;
[R1] 3*a3+4*a4+5*a5<14;
@GIN(a3);
@GIN(a4);
@GIN(a5);
```

obtaining the optimal solution

$$Z = 0,\ a_3 = 2,\ a_4 = 2 \text{ and } a_5 = 0$$

The optimal Z-value is found to be $Z = 0$. This means that no combination can price out favorably. Hence, our current basic solution must be an optimal solution. To find the values of the basic variables in the optimal solution, we find the right-hand side of the current tableau:

$$\text{Right-hand side of current tableau} = \mathbf{B}_3^{-1}\mathbf{b} = \begin{pmatrix} \frac{2}{7} & -\frac{2}{7} & \frac{1}{7} \\ \frac{1}{14} & \frac{3}{7} & -\frac{3}{14} \\ -\frac{1}{7} & \frac{1}{7} & \frac{3}{7} \end{pmatrix} \begin{pmatrix} 20 \\ 25 \\ 30 \end{pmatrix} = \begin{pmatrix} \frac{20}{7} \\ \frac{40}{7} \\ \frac{95}{7} \end{pmatrix}$$

Therefore, the optimal solution to this cutting stock problem is: $x_4 = \frac{20}{7} \quad x_7 = \frac{40}{7} \quad x_1 = \frac{95}{7}$ If desired, we could obtain a 'reasonable' integer solution by rounding x_4, x_7 and x_1 upward. This yields the integer solution: $x_4 = 3 \quad x_7 = 6 \quad x_1 = 14$

Appendix 2

Further applications of column generation techniques

Five decades ago, Ford and Fulkerson (1958) suggested for the first time to deal only implicitly with the variables of a multicommodity flow problem. Dantzig and Wolfe (1960) pioneered this fundamental idea and developed a strategy to extend a linear program columnwise as needed in the solution process. This technique was first put to actual use by Gilmore and Gomory (1961, 1963) as part of an efficient heuristic algorithm for solving the cutting stock problem. Column generation is nowadays a well-known method to cope with a huge number of variables. In this appendix we will give a general review of the applications of column generation techniques nowadays.

If we introduce the key words *column generation* in the web search engine Google Scholar, 3,230,000 results are found, meaning that there are 3,230,000 peer-reviewed online academic journals and books, theses, dissertations, conference papers, technical reports, and other scholarly literature, including patents, that are related to column generation. Making a more accurate search, we find that in the last year, that is, from 2016 to 2017, there are 68,700 results.

If we repeat the search in the website ScienceDirect, which provides subscription-based access to a large database of scientific and medical research, 407,783 results are found for column generation. In this case, 49,518 of the results were published in the last year

After examining the results of both sites we observed that many of them describe integer programming column generation applications: Various vehicle routing problems, crew scheduling, multiple traveling salesman problem with time windows, real-time dispatching of automobile service units, multiple pickup and delivery problem with time windows, airline crew pairing, air network design for express shipment service, airline schedule generation, fleet assignment and aircraft routing and scheduling, job grouping for flexible manufacturing systems, grouping and packaging of electronic circuits, bandwidth packing in telecommunication networks, traffic assignment in satellite communication systems, course registration at a business school, graph partitioning, single-machine multi-item lot sizing, bin packing and cutting-stock problems, integer multicommodity flows, maximum stable-set problem, probabilistic maximum satisfiability problem, minimum cut clustering, graph coloring, generalized assignment problem, etc.

Among the papers, we may also find papers that deal with enhancing the algorithms and some of them are mainly theoretical. Theoretical papers focus on the understanding of the dual point of view whose growth has brought considerable progress to the column generation theory and practice. This progress stimulated careful initializations, sophisticated solution techniques for the restricted master problem and the subproblem, as well as better overall performance. Column generation is clearly a success story in largescale integer programming. Nowadays generic integer programming column generation codes solve many large-scale problems of 'industrial difficulty', that no standard commercial mixed integer programming solver could cope with. Moreover, nonlinearities occurring in practical problems can be taken care of in the subproblem.