

Outline

1. Decision making and operations research.
 - (a) The need for optimization.
 - (b) Typical models of industry and finance.
2. Advanced linear programming (LP)
 - (a) Need for linear models.
 - (b) General form of LP problems.
 - (c) The sparse simplex method; primal and dual algorithms; computational implications.
 - (d) Economic interpretation of the optimality conditions: Sensitivity analysis.
 - (e) Interior point algorithms.
 - (f) State-of-the-art in solving LP problems.
3. Integer programming (IP)
 - (a) Situations requiring integer valued models.
 - (b) IP problem formulation.
 - (c) Solution methods; Implicit enumeration, branch and bound (B&B) method.
 - (d) Network programming (optimization in networks).
 - (e) Logic constraints and IP.
 - (f) State-of-the-art in solving IP problems.
4. Modeling
 - (a) Basics of modeling.
 - (b) Model generation and management.
 - (c) Modeling languages, modeling systems.

478 Advanced Operations Research

István Maros

Department of Computing
Imperial College, London
i.maros@imperial.ac.uk

2008 November 13

Version v3.2d

Traditional decision support (What-if type modeling)

1. Observing and formulating the problem.
2. Constructing a computational model that captures the essence of the real problem.
3. Picking (guessing) a solution to the model (also referred to as generating an alternative).
4. Evaluating the solution and analyzing its validity. Comparing it with solutions obtained so far (learning).
5. Depending on the outcome of step 4, generating a new alternative and returning to step 4 or modifying the model and returning to step 3 or accepting the currently best solution.
6. Presenting the results to decision makers.

Advantages: Gives answers to “what if” type questions, easy to use, *a priori* knowledge of model can help generate reasonable alternatives. Spreadsheets are the typical computational tools for this type of analysis.

Disadvantages: Process is slow if too many alternatives are evaluated, search for a good solution is unguided, a “guess” solution may not even be technically feasible, inaccurate if too few solutions are tested, the quality of the accepted solution is unknown (how much better could have been found).

There is a **need for optimization**.

Modern decision support using optimization technology

1. Observing and formulating the problem. It usually involves the use of logic programming methodology that can provide a natural framework for problem specification.
2. Constructing a computational model that captures the essence of the real problem.
3. Deriving a solution from the model: Solving the problem using large scale sparse optimization technology. This results in a best possible (optimal) solution under the constraints of the model.
4. Verifying and analyzing the validity of results. If necessary, the model is modified and the process is resumed at step 3.
5. Presenting the results to decision makers.

This is the area of applying advanced computational tools and techniques of operations research.

Some application areas of advanced OR

The same tools can be used in very different areas like

- Industry (petroleum, chemical, manufacturing, food, steel, paper)
- Finance
- Telecommunications
- Transport
- Agriculture
- Energy
- Defence
- Mining
- Government
- Healthcare
- Manpower Planning
- Advertising
- etc.

Characteristics of real-life decision problems

Real-life decision problems are typically

1. **Large:** Many variables, equations and constraints are needed to create realistic models.
2. **Complex:** Model may include time dependence, feedback and nonlinearity. Additional features may include logical relationships among variables and constraints, integrality requirement of some variables, etc.
3. **Sparse:** The computational model usually consists of vectors and matrices of large scale. They are sparse, that is, contain very few nonzero entries. The ratio of nonzeros to the total number of positions is often in the range of 0.05%–1.00%. By using proper computational tools, sparsity enables us to represent and solve such problems.
4. **Difficult to solve,** due to their size and complexity. Different models require completely different solution methods. For each type of model there exist several solution algorithms. They have been designed to cope with arising numerical and algorithmic difficulties.

Product Mix (simplified model)

The most typical industrial problem is to determine the optimal product mix that maximizes revenue (or profit).

A flourishing manufacturing company can produce 6 different products, Pr-1, . . . , Pr-6. They require labour, energy, and processing on machines. These resources are available in limited amounts. The company wants to determine what quantities to produce that maximize the monthly revenue, assuming that any amount can be sold. The following table describes the technological requirements of producing one unit of each product, the corresponding revenue and the monthly availability of the resources.

	Pr-1	Pr-2	Pr-3	Pr-4	Pr-5	Pr-6	Limit
Revenue	5	6	7	5	6	7	
Machine hour	2	3	2	1	1	3	1050
Labour hour	2	1	3	1	3	2	1050
Energy	1	2	1	4	1	2	1080

The **decision variables** are the unknown quantities of the products. They are denoted by x_1, \dots, x_6 .

The revenue to be maximized is

$$5x_1 + 6x_2 + 7x_3 + 5x_4 + 6x_5 + 7x_6$$

The resource constraints are:

$$2x_1 + 3x_2 + 2x_3 + 1x_4 + 1x_5 + 3x_6 \leq 1050$$

$$2x_1 + 1x_2 + 3x_3 + 1x_4 + 3x_5 + 2x_6 \leq 1050$$

$$1x_1 + 2x_2 + 1x_3 + 4x_4 + 1x_5 + 2x_6 \leq 1080$$

Since production must be nonnegative, we impose $x_j \geq 0$, $j = 1, \dots, 6$.

This is a simple LP problem that can be solved by the simplex method. The solution is $x_2 = 240$, $x_4 = 90$, $x_5 = 240$, and $x_1 = x_3 = x_6 = 0$, giving a revenue of 3330.00 units.

While the problem is simple its analysis highlights some interesting points.

- The most rewarding products (Prod-3 and Prod-6) are not included in the optimal mix.
- It is sufficient to produce not more than three products in order to achieve the maximum possible revenue.
- All resources are fully utilized (constraints are satisfied with equality).

Portfolio Optimization (simplified model)

Investor's portfolio at the beginning of the year:

Share	# of shares held	current value per share
Shiny	75	£20
Risky	1000	£2
Trusty	25	£100

There is a one-off chance to change the portfolio for a whole year ahead. It is assumed that **at the end of the year** the position of the shares will be:

Share	dividend per share	value per share
Shiny	£5	£18
Risky	£0	£3
Trusty	£2	£102

Investor does not want to change the composition of the portfolio but possibly the quantities. It is assumed that there is no fee for buying or selling stocks and fractional stocks are permitted.

Purpose: to adjust the portfolio to maximize the end-of-year dividends subject to the following restrictions:

1. Current value of the portfolio must remain the same (no more money to invest).
2. Combined end-of-year value of all holdings should be at least 5% more than the total value now (to beat inflation).
3. Keep a balanced portfolio: current value of each stock after adjustment must be at least 25% of the current value of the entire portfolio.
4. The rearrangement of the portfolio can take place only now and cannot be altered during the year.

Note: without rearrangement the end-of-year dividend would be: $75(5) + 25(2) = £425$.

Solution

First: define decision variables. Let

x_1 be the *change* in the # of shares of Shiny,
 x_2 be the *change* in the # of shares of Risky,
 x_3 be the *change* in the # of shares of Trusty.

E.g., $x_1 = 20$ means the holding of Shiny changes from its current value of 75 to 95 (buy 20), while if $x_1 = -25$, holding of Shiny changes from 75 to 50 (sell 25). Similar interpretation applies to x_2 and x_3 .

Next, express our objective: **maximize total dividends**.

Dividend from Shiny: dividend per share multiplied by the number of shares held at the end of the year, i.e. $5(75 + x_1)$. Contributions of Risky and Trusty can be determined in a similar fashion resulting in the total dividends of

$$5(75 + x_1) + 0(1000 + x_2) + 2(25 + x_3) \quad (1)$$

(1) is called the **objective function** of the problem that is to be maximized (in this example).

Note: At this stage Risky does not seem to contribute to the objective function (expected dividend is zero). However, it will affect the solution through the constraints.

A simplified form of (1) is

$$5x_1 + 2x_3 + 425$$

This is a linear function. If we maximize (or minimize) it the additive constant can be ignored.

Restrictions 1–4 impose constraints on the decision variables.

Current value of portfolio must be unaltered. Original value: $20(75) + 2(1000) + 100(25) = 6000$. For the rearranged portfolio: $20(75 + x_1) + 2(1000 + x_2) + 100(25 + x_3) = 6000$,

$$20x_1 + 2x_2 + 100x_3 = 0.$$

Value of the portfolio must increase by 5% to 6300 (inflation). $18(75 + x_1) + 3(1000 + x_2) + 102(25 + x_3) \geq 6300$, or

$$18x_1 + 3x_2 + 102x_3 \geq -600.$$

Balanced stock holding imposes one constraints for each stock. The total value of the portfolio after rearrangement at the beginning of the year is the same as the original (£6000). The value of Shiny, Risky and Trusty must be at least one quarter of it:

$$20(75 + x_1) \geq 1500 \quad \Rightarrow \quad x_1 \geq 0 \quad (2)$$

$$2(1000 + x_2) \geq 1500 \quad \Rightarrow \quad x_2 \geq -250 \quad (3)$$

$$100(25 + x_3) \geq 1500 \quad \Rightarrow \quad x_3 \geq -10 \quad (4)$$

Natural requirement: after the adjustment of the portfolio the quantities held are nonnegative. Therefore:

$$75 + x_1 \geq 0 \quad \Rightarrow \quad x_1 \geq -75 \quad (5)$$

$$1000 + x_2 \geq 0 \quad \Rightarrow \quad x_2 \geq -1000 \quad (6)$$

$$25 + x_3 \geq 0 \quad \Rightarrow \quad x_3 \geq -25 \quad (7)$$

Note, equations (5)–(7) are less restrictive than (2)–(4) and, therefore, do not impose additional constraints on the variables.

Final form of the problem

$$\begin{array}{rcl}
 \max & 5x_1 & + 2x_3 \\
 \text{subject to} & 20x_1 + 2x_2 + 100x_3 & = 0 \\
 & 18x_1 + 3x_2 + 102x_3 & \geq -600 \\
 & x_1 & \geq 0 \\
 & & x_2 \geq -250 \\
 & & x_3 \geq -10
 \end{array}$$

This is a **linear programming (LP)** problem.

Without solving the problem, it is clear that we cannot sell Shiny ($x_1 \geq 0$ must hold), we can sell at most 250 shares of Risky and 10 shares of Trusty. The exact amounts become known if the problem is solved by an appropriate solution algorithm. They are: $x_1 = 75$, $x_2 = -250$, $x_3 = -10$.

Interpretation: buy 75 shares of Shiny, sell 250 shares of Risky and 10 shares of Trusty. Thus the holding of these shares at the end of the year (when dividends are payable):

Shiny: 75 original plus 75 purchased, total of 150,
 Risky: 1000 original minus 250 sold, total of 750,
 Trusty: 25 original minus 10 sold, total of 15.

Value of the portfolio: $150(18) + 750(3) + 15(102) = £6480$ which is well above the minimum requirement of £6300.

The resulting **dividends:** $150(5) + 15(2) = £780 (> £425)$.

LINEAR PROGRAMMING (LP)

Verbal description:

Linear Programming is a general technique to solve a wide range of optimization problems. It is the key engine of most decision support systems. LP is directly applicable when the model of a problem shows a linear relationship among the decision variables. (Reminder: usually several different models can be constructed for a given decision problem.)

Additionally, LP is often the hidden engine behind the solution algorithms of many other (nonlinear, integer valued, stochastic) decision support models and techniques (details to come).

Finally, there are cases when a problem can be transformed into an equivalent linear model that can be solved using LP technology.

In practice, many real life decision problems can be expressed or approximated by linear models, therefore the importance of LP is enormous.

The main constituent parts of an LP problem are the **decision variables** the values of which are to be determined in such a way that certain restrictions are not violated and some goal is achieved. The restrictions are often referred to as **constraints** and the function expressing the goal is called **objective function**. The constraints and objective are **linear functions** of the decision variables (see product mix and portfolio problems).

Examples of objective functions in LP

Maximize	Minimize
profit	cost
utility	completion time
return on investment	risk
net present value	losses
number of employees	number of employees
turnover	redundancies
customer satisfaction	distance
robustness of operating plan	

Typical LP constraints

- Productive capacity
- Raw material availabilities
- Marketing demands and limitations
- Material balance
- Quality stipulations

Nature of constraints

Deterministic	Hard
Stochastic (chance)	Soft

Examples of applications of LP

Organization	Description	Annual Savings
Santos, Ltd.	Optimize capital investment for producing natural gas over a 25-year period	\$3 million
Texaco, Inc.	Optimally blend available ingredients into gasoline products to meet quality and sales requirements.	\$30 million
United Airlines	Crew and aircraft scheduling	Huge undisclosed
American Airlines	Optimize fare structure, overbooking and coordinating flights to increase (maximize) revenue	\$500 million
San Francisco Police Dept.	Optimally schedule and deploy police patrol	\$11 million

Formal statement of the LP problem

Assumption: there are n decision variables, x_1, x_2, \dots, x_n and m constraints.

Standard form:

$$\begin{aligned} \min \quad & c_1x_1 + \dots + c_nx_n \quad \Rightarrow \sum_{j=1}^n c_jx_j \\ \text{s.t.} \quad & a_{i1}x_1 + \dots + a_{in}x_n = b_i \quad \Rightarrow \sum_{j=1}^n a_{ij}x_j = b_i \\ & \text{for } i = 1, \dots, m \\ & x_j \geq 0, \text{ for } j = 1, \dots, n \end{aligned}$$

where c_j is a constant associated with activity (variable) j , sometimes called *cost coefficient*, a_{ij} is the coefficient of variable j in constraint i , and b_i is the right-hand-side (RHS) coefficient of constraint i .

The first line is the **objective function**, the second is the set of **general (joint) constraints** (each involving several variables), the third is referred to as **nonnegativity constraints**.

The value of the objective function is often denoted by z , i.e., $z = c_1x_1 + \dots + c_nx_n$.

Remarks:

1. Objective function can also be maximized.
2. Any constraint can be converted into equality (see later).

Assumptions of LP

In LP the following features are assumed.

Proportionality: The contribution of each variable (activity) to the value of the objective function is proportional to the level of activity x_j , as represented by the c_jx_j term in the objective function.

Similarly, the contribution of each activity to the left-hand-side of each joint constraint is proportional to the level of activity x_j , as represented by the $a_{ij}x_j$ term in the i -th constraint.

Additivity: Every function (objective, constraint) in LP is the sum of the individual contributions of the respective activities.

Divisibility: Decision variables in an LP model are allowed to take any value (integer, fractional).

Certainty: The coefficients of an LP model (c_j, a_{ij}, b_i) are known constants.

The certainty assumption is seldom satisfied precisely. The outcome of a model depends on the actual values of the coefficients. **Sensitivity analysis** is a tool in LP that can determine the sensitivity of an optimal solution to small changes of the coefficients. As such, it can identify the critical coefficients whose value cannot be changed without changing the optimal solution.

General form of LP

The **objective function** can have an additive constant c_0 (the starting value)

$$\min z = c_0 + c_1x_1 + \dots + c_nx_n \Rightarrow c_0 + \sum_{j=1}^n c_jx_j$$

General constraints:

$$L_i \leq \sum_{j=1}^n a_{ij}x_j \leq U_i, \quad i = 1, \dots, m \quad (8)$$

Individual constraints (bounds):

$$l_j \leq x_j \leq u_j \quad j = 1, \dots, n \quad (9)$$

	x_1	\dots	x_j	\dots	x_n	
L_1		\dots	a_{1j}	\dots		U_1
\vdots			\vdots			\vdots
L_i	a_{i1}	\dots	a_{ij}	\dots	a_{in}	U_i
\vdots			\vdots			\vdots
L_m		\dots	a_{mj}	\dots		U_m
l_1	1					u_1
\vdots		\dots				\vdots
l_j			1			u_j
\vdots				\dots		\vdots
l_n					1	u_n

Bounds: special cases

1. $u_j = +\infty, l_j$ finite: (9) $\Rightarrow l_j \leq x_j$, such x_j is called **plus type variable**.
2. $l_j = -\infty, u_j$ finite: (9) $\Rightarrow x_j \leq u_j$, such x_j is called **minus type variable**.

Remark 1 A minus type variable can be converted into a plus type variable by substituting $\bar{x}_j = -x_j$ and $\bar{l}_j = -u_j \Rightarrow \bar{l}_j \leq \bar{x}_j$.

3. Both l_j and u_j are finite: (9) $\Rightarrow l_j \leq x_j \leq u_j$, such x_j is called **bounded variable**
Special sub-case: $l_j = u_j = x_j$, such x_j is called **fixed variable**
4. $l_j = -\infty$ and $u_j = +\infty$: (9) $\Rightarrow -\infty \leq x_j \leq +\infty$, such x_j is called **unrestricted (free) variable**

Constraints: special cases

1. $L_i = -\infty, U_i$ finite: (8) $\Rightarrow \sum_{j=1}^n a_{ij}x_j \leq U_i,$

also known as $\sum_{j=1}^n a_{ij}x_j \leq b_i, \leq$ **type constraint**.

Converted into equation:

$y_i + \sum_{j=1}^n a_{ij}x_j = b_i$ with $y_i \geq 0.$

2. $U_i = +\infty, L_i$ finite: (8) $\Rightarrow L_i \leq \sum_{j=1}^n a_{ij}x_j,$

also known as $\sum_{j=1}^n a_{ij}x_j \geq L_i, \geq$ **type constraint**.

Denoting $b_i = -L_i,$ equivalent form: $\sum_{j=1}^n (-a_{ij})x_j \leq b_i$

Converted into equation:

$y_i + \sum_{j=1}^n (-a_{ij})x_j = b_i$ with $y_i \geq 0.$

3. Both L_i and U_i are finite: Two general constraints

$L_i \leq \sum_{j=1}^n a_{ij}x_j$ and $\sum_{j=1}^n a_{ij}x_j \leq U_i$

They are equivalent to a general and an individual constraint:

$y_i + \sum_{j=1}^n a_{ij}x_j = U_i$ with $0 \leq y_i \leq (U_i - L_i).$

This is called a **range constraint**.

Denoting $b_i = U_i$ and $r_i = (U_i - L_i):$

$y_i + \sum_{j=1}^n a_{ij}x_j = b_i$ with $0 \leq y_i \leq r_i$

True even if $L_i = U_i$ in which case $y_i = 0.$

This is referred to as an **equality constraint**.

4. $L_i = -\infty$ and $U_i = +\infty$ then with arbitrary $b_i,$ formally:

$y_i + \sum_{j=1}^n a_{ij}x_j = b_i$ with y_i unrestricted (free).

This is referred to as a **non-binding constraint**.

Conclusion: the general constraints can be brought into the following form:

$y_i + \sum_{j=1}^n a_{ij}x_j = b_i, i = 1, \dots, m$ (10)

$y_i (i = 1, \dots, m)$ variables are called **logical variables**, $x_j (j = 1, \dots, n)$ variables are the **structural variables**.

Rule 1 (Conversion of constraints) To convert all general constraints into equalities:

1. Add a nonnegative logical variable to each \leq type constraint.
2. Multiply each \geq type constraint by -1 (RHS element included) and add a nonnegative logical variable.
3. Set the RHS of a range constraint to the upper limit and add a bounded logical variable to the constraint.
4. Add a free logical variable to each non-binding constraint.
5. Finally, for uniformity: Add a zero valued logical variable to each equality constraint.

Example 1 Convert the following constraints into equalities

$$\begin{aligned}
 3x_1 + 2x_2 + 6x_3 - x_4 &\leq 9 \\
 x_1 - x_2 + x_4 &\geq 3 \\
 2 \leq x_1 + 2x_2 + 3x_3 - 4x_4 &\leq 15 \\
 2x_1 - x_2 - x_3 + x_4 &<> 10 \\
 x_1 + 3x_2 - x_3 + 6x_4 &= 8 \\
 x_1 \geq 0, x_2 \leq 0, -4 \leq x_3 \leq -1, x_4 &\text{ free}
 \end{aligned}$$

Constraint-1: add nonnegative logical variable y_1 (Rule 1, part 1)

$y_1 + 3x_1 + 2x_2 + 6x_3 - x_4 = 9, y_1 \geq 0$

Constraint-2: multiply constraint by -1 and add a nonnegative logical variable y_2 (Rule 1, part 2)

$y_2 - x_1 + x_2 - x_4 = -3, y_2 \geq 0$

Constraint-3: set RHS to 15 and add a bounded logical variable y_3 (Rule 1, part 3)

$y_3 + x_1 + 2x_2 + 3x_3 - 4x_4 = 15, 0 \leq y_3 \leq 13$

Constraint-4: add a free logical variable y_4 (Rule 1, part 4)

$y_4 + 2x_1 - x_2 - x_3 + x_4 = 10, y_4$ free

Constraint-5: add a zero valued logical variable y_5 (Rule 1, part 5)

$y_5 + x_1 + 3x_2 - x_3 + 6x_4 = 8, y_5 = 0$

Types of variables

Assumption:

all minus type variables have been reversed and all finite lower bounds have been moved to zero. Based on their feasibility range, variables (whether logical or structural) are categorized as follows:

Feasibility range	Type	Reference
$y_i, x_j = 0$	0	Fixed
$0 \leq y_i, x_j \leq u_j$	1	Bounded
$0 \leq y_i, x_j \leq +\infty$	2	Nonnegative
$-\infty \leq y_i, x_j \leq +\infty$	3	Free

(12)

Correspondence between the types of constraints and types of the logical variables in

$$y_i + \sum_{j=1}^n a_{ij}x_j = b_i, \quad i = 1, \dots, m \quad (13)$$

- $\text{type}(y_i) = 0 \leftrightarrow$ constraint is equality
- $\text{type}(y_i) = 1 \leftrightarrow$ constraint is range constraint
- $\text{type}(y_i) = 2 \leftrightarrow$ original constraint is type \geq or \leq
- $\text{type}(y_i) = 3 \leftrightarrow$ constraint is free (non-binding)

In the standard form of LP all constraints are equalities and all variables are of type 2 (nonnegative).

Ultimate formulation of LP

Assumption:

every general constraint \Rightarrow equality and has got a logical variable, all minus type variables are reversed, all finite lower bounds are moved to zero, the resulting new variables are denoted by their corresponding original counterpart, new RHS values are also denoted by the original notation; all changes have been recorded to enable the expression of the solution in terms of the original variables and constraints. The LP problem is now:

$$\begin{aligned} \min \quad & c_0 + \sum_{j=1}^n c_j x_j \\ \text{s.t.} \quad & y_i + \sum_{j=1}^n a_{ij} x_j = b_i, \quad \text{for } i = 1, \dots, m \end{aligned}$$

and the type constraints on the variables.

c_0 plays no role in any solution algorithm and is ignored in the sequel.

With matrix notation:

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{x} & (14) \\ \text{s.t.} \quad & \mathbf{Ax} + \mathbf{Iy} = \mathbf{b} & (15) \\ & \text{and every variable is one of types 0-3.} & (16) \end{aligned}$$

From technical point of view, logical and structural variables play equal role in (14) – (16). In general, there is no need to distinguish them. We introduce a simplified notation. Vectors \mathbf{x} and \mathbf{c} are redefined as

$$\mathbf{x} := \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, \quad \mathbf{c} := \begin{bmatrix} \mathbf{c} \\ \mathbf{0} \end{bmatrix},$$

with $\mathbf{0}$ being the m dimensional null vector. The new dimension of \mathbf{x} and \mathbf{c} is defined to be $n := n + m$. The matrix on the left hand side of (15) is redefined as

$$\mathbf{A} := [\mathbf{A} \mid \mathbf{I}].$$

The general form (14) – (16) is rewritten as

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \text{type}(x_j) \in \{0, 1, 2, 3\}, \quad j = 1, \dots, n \end{aligned}$$

Every LP problem can be brought into this form.

Why are constraints ' \leq ' or ' \geq ' and not ' $<$ ' or ' $>$ '?

Because there is no solution if strict ' $<$ ' (or ' $>$ ') is required. For example,

$$\begin{aligned} \min \quad & z = x_1 \\ \text{s.t.} \quad & x_1 > 1. \end{aligned}$$

Multiple Objectives

The (14) – (16) formulation of the LP model involves a single objective function. In practice, however, there may be several objectives that the user of the model wants to take into account. E.g., the objectives can be to *minimize costs and minimize redundancy*, or to *maximize profit and minimize risk*. There are realistic cases when more than two objectives are to be observed. The different objectives are often in conflict to some extent.

Interestingly enough, the multiobjective problems can be tackled by LP modelling techniques. We discuss some practical approaches.

Separate objectives

Solve the model with each objective separately. The comparison of the different results may suggest a satisfactory solution to the problem or indicate further investigations.

Note: All potential objectives can be included as non-binding constraints (with logical variables of type-3).

Objectives as constraints

Objectives and constraints can often be interchanged, e.g., we may wish to pursue some desirable social objective so long as costs do not exceed a specified level. Alternatively, we may wish to minimize costs using the social consideration as a constraint. The general use of this approach is to treat all but one objective as constraints and solve the problem. Then swap the objective and one of the potential objectives and solve it again. Repeat this pattern as many times as necessary.

Aggregation

We can define the relative importance of the objectives in question. This is achieved by attaching weights to them. These weights can be used to take the linear combination of the objectives and create a single objective function in the following way. Assume there are p objective functions given:

$$\begin{aligned} z_1 &= c_{11}x_1 + \cdots + c_{1n}x_n \\ &\quad \dots \\ z_p &= c_{p1}x_1 + \cdots + c_{pn}x_n. \end{aligned}$$

If the weights are denoted by w_1, \dots, w_p then the single **composite objective function** $z = \sum_{i=1}^p w_i z_i$ can be used:

$$z = \left(\sum_{i=1}^p w_i c_{i1} \right) x_1 + \cdots + \left(\sum_{i=1}^p w_i c_{in} \right) x_n.$$

Again, the best way of using this approach is to experiment with different composite objectives to see how solutions change and present the desirable ones as decision alternatives.

Goal programming

If there are conflicting objectives and they are represented as constraints with some expectations (RHS values) the problem may be infeasible (no feasible solution can be found). Therefore, we need a mechanism to relax some or all of them. This can be done by **goal programming**. Now, the RHS values become goals that may or may not be achieved. The over or under achievements are represented by variables, called *overshoot* and *undershoot variables*.

Let an objective constraint be the i -th constraint in the matrix $a_{i1}x_1 + \cdots + a_{in}x_n$ and the desirable RHS value, the **goal**, b_i . If the goal is not achieved we still can make this constraint an equality by adding a nonnegative undershoot variable s_i to it. Similarly, if the goal is over achieved, we can subtract a nonnegative overshoot variable t_i from it to equate the LHS to RHS. Thus, in the goal programming framework, a constraint that is included as an achievable goal constraint will be represented as

$$a_{i1}x_1 + \cdots + a_{in}x_n + s_i - t_i = b_i$$

with the two added variables being nonnegative, $s_i \geq 0$, $t_i \geq 0$. We want to penalize the deviation from the RHS. This can be done by assigning objective coefficients to the under- and overshoot variables. In this way we even can differentiate between over and under achievement of a single goal and can also express the relative importance of the different goals by different penalties of the over- and undershoot variables in the objective function.

Example 3 *Goal programming solution to a multiple objective production planning problem.*

A manufacturing company wants to minimize its production costs while meeting its orders. The production plan should be such that some social goal is also achieved by using weekly 288 hours of grinding and 192 hours of drilling capacity (to avoid redundancy). Additionally, keeping the grinding capacity is twice as important as that of drilling. Furthermore, under achievement is twice as bad as over achievement. Keeping other parts of the model (production variables, other constraints) at symbolic level, formulate the goal programming problem.

Solution:

Introduce undershoot and overshoot variables for the grinding and drilling capacity: s_g, s_d, t_g, t_d (all nonnegative). Assuming the two capacities are included as the first two rows in the matrix, we can formulate the constraints as

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n + s_g - t_g &= 288 \\ a_{21}x_1 + \cdots + a_{2n}x_n + s_d - t_d &= 192 \end{aligned}$$

To keep some notational convention, the new variables can be indexed by the row they are attached to. So, s_g and t_g can be denoted by s_1 and t_1 in this example.

The critical issue is how to determine the penalty (objective) coefficients. We have to set them in such a way that they do not bias the model (this is a rather soft requirement). In any case, they will be a function of the magnitude of the other objective coefficients (c_1, \dots, c_n). We assume proportionality at a rate denoted by λ in the following sense ($\lambda > 0$ for min problem).

First we set the obj. coeff. of the less important capacity (drilling). For overshooting, $c_{do} = \lambda$. Since undershooting is twice as bad, $c_{du} = 2\lambda$. Grinding capacity is twice as important, therefore $c_{go} = 2\lambda$ and $c_{gu} = 4\lambda$. The objective function will be:

$$\min z = c_1x_1 + \cdots + c_nx_n + 4\lambda s_g + 2\lambda t_g + 2\lambda s_d + \lambda t_d.$$

Several actual values for λ (i.e., $\lambda = \text{average}\{|c_1|, \dots, |c_n|\}$) should be tried and the (subjectively) most suitable can be accepted.

Minimax objective

Sometimes the different objectives are taken into account in the following way.

We are given p objective functions with coefficient (row) vectors c^1, \dots, c^p . They define objective values c^1x, \dots, c^px for any given x . Function $f(x)$ is defined as the maximum of them:

$$f(x) = \max_{1 \leq k \leq p} \{c^k x\}$$

In several situations we need to find an x such that $f(x)$ is minimized, i.e., the maximum of the objectives is as small as possible, where x is restricted by some usual LP constraints. This is the **minimax** problem:

$$\min\{f(x) : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}.$$

It can be converted into the following form

$$\min\{z : z - c^k x \geq 0, k = 1, \dots, p, \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\},$$

which is an LP problem. It says we minimize a variable z that makes all $z - c^k x \geq 0$, including the largest. In other words, we minimize the maximum of $c^k x$ which is, by definition, $f(x)$.

Thus, the minimax problem can be solved by the techniques of linear programming.

The Simplex Method with all types of variables

Summary of notions.

Recalling the final form of LP:

$$\min \quad \mathbf{c}^T \mathbf{x} \quad (17)$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \quad (18)$$

$$\text{type}(x_j) \in \{0, 1, 2, 3\}, \quad j = 1, \dots, n \quad (19)$$

An $\mathbf{x} \in \mathbb{R}^n$ is a **solution** to the LP problem if it satisfies the general constraints (18). An $\mathbf{x} \in \mathbb{R}^n$ is a **feasible solution** if it is a solution and satisfies the type constraints (19).

A feasible solution \mathbf{x}^* is called an **optimal solution** if there is no feasible solution with better objective value: $\mathbf{c}^T \mathbf{x}^* \leq \mathbf{c}^T \mathbf{x}$, for all feasible \mathbf{x} .

Matrix \mathbf{A} in (18) has a full row rank of m , therefore, there exist m columns in \mathbf{A} that form a nonsingular matrix $\mathbf{B} \in \mathbb{R}^{m \times m}$. Such a \mathbf{B} is called a **basis**.

Assume: a basis \mathbf{B} is given and, WROG (without restriction of generality), it is permuted into the first m columns of \mathbf{A} . \mathbf{A} is then partitioned as $\mathbf{A} = [\mathbf{B} \mid \mathbf{N}]$, where \mathbf{N} denotes the nonbasic part of \mathbf{A} . Variables associated with the columns of the basis are called **basic variables** while the remaining variables are the **nonbasic variables**. Vectors \mathbf{c} and \mathbf{x} are partitioned accordingly. Index set of basic variables is B , that of nonbasic variables is N .

(18) can be rewritten as $[\mathbf{B} \mid \mathbf{N}] \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{bmatrix} = \mathbf{b}$, or

$$(\mathbf{A}\mathbf{x} =) \mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}. \quad (20)$$

The partitioned form of the objective function is

$$\mathbf{c}^T \mathbf{x} = \mathbf{c}_B^T \mathbf{x}_B + \mathbf{c}_N^T \mathbf{x}_N.$$

From (20) we obtain $\mathbf{B}\mathbf{x}_B = \mathbf{b} - \mathbf{N}\mathbf{x}_N$ and, since \mathbf{B}^{-1} exists,

$$\mathbf{x}_B = \mathbf{B}^{-1}(\mathbf{b} - \mathbf{N}\mathbf{x}_N). \quad (21)$$

\mathbf{x}_B is called a **basic solution** corresponding to basis B . If \mathbf{x}_B is feasible it is called **basic feasible solution (BFS)**.

(21) says that a basic solution is uniquely determined by the values of the nonbasic variables. If the nonbasic variables are set to zero (which is not necessarily the case) then $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$.

The maximum possible number of different bases is

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}.$$

One of the main theorems of LP says that if a problem has an optimal solution then there exists at least one **optimal basic solution**. Therefore, it is sufficient to deal only with basic solutions. If there are more than one optimal solutions then we talk about **alternative optima**. If there are more than one optimal basic solutions then any convex linear combination of them is also an optimal solution (usually not basic).

The **simplex method (SM)** is an iterative numerical procedure that, in the absence of degeneracy (see later), in a finite number of steps gives one of the following answers to an LP problem:

1. The problem has **no feasible solution**.
2. The problem has an **unbounded solution**.
3. An **optimal solution** has been found.

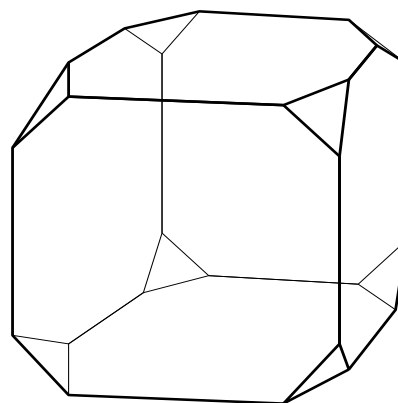
Two bases are called **neighbouring bases** if they differ only in one column. In each iteration the SM moves to a neighbouring basis with a better (or not worse) objective value.

Different views of the LP problem:

1. Combinatorial
2. Geometric
3. Algebraic

Geometry of constraints

The constraints of an LP problem usually determine a convex polyhedron. In case of contradicting constraints it is empty. Here is a three dimensional convex polytope (= bounded polyhedron).



Convex set (18) can also be unbounded.

Geometric interpretation of the simplex method

A basic solution corresponds to a **vertex** of the polyhedron. If two vertices are connected by an **edge** they are called **neighbouring vertices (bases)**. The vertices and edges form a network. SM moves along the edges of this network until termination.

Observation 1 Consider the standard LP problem:

$$\min_x \{c^T x : Ax = b, x \geq 0\}.$$

An optimal basic solution means that no more than m variables are positive in the optimum regardless of the total number of variables (n).

Corollary 1 In a production planning (product mix) model where a company can potentially produce hundreds of different products but there are only, say, 50 constraints defined then the maximum revenue can be achieved by producing not more than 50 products. The simplex method can identify those 50.

Degeneracy

If at least one component of x_B has a value equal to its finite (lower or upper) bound then the basic solution is called **degenerate**. Such a solution may prevent the SM from making an improving step (zero improvement). SM is sensitive to degeneracy since it can make the algorithm **stalling** (many non-improving steps in a row) or even **cycling** (infinite repetition of a sequence of non-improving iterations).

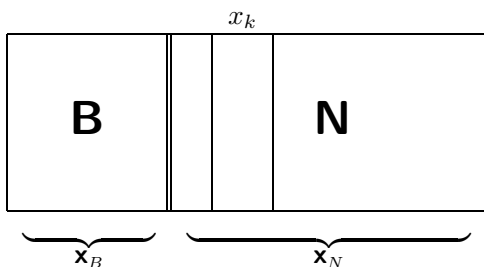
Example 4 Given x_1, x_2 and x_3 are basic variables with individual bounds $x_1 \geq 0, 0 \leq x_2 \leq 2, x_3 \geq 0$. Assume, the value of basic variables: $x_1 = 1, x_2 = 2, x_3 = 3$. This basic solution is degenerate because x_2 is at one of its bounds, namely at upper bound. If the values were $x_1 = 2, x_2 = 1, x_3 = 3$ the solution would not be degenerate, while $x_1 = 2, x_2 = 1, x_3 = 0$ is degenerate again because $x_3 = 0$.

During a simplex iteration all we need is the transformed objective row (to select an improving variable), the transformed column of the improving variable x_q and the basic solution x_B (to determine the outgoing variable by the ratio test).

OSM has all these items ready and much more. Unnecessarily, it computes *all* the transformed columns of which only one is used in the next iteration.

RSM is based on the observation that any element of the transformed tableau (TT) can be determined by the inverse of the basis since the whole TT is nothing but $B^{-1}A$. RSM computes certain parts of the TT on demand, as seen later.

For further discussion, it is useful to visualize the $[B | N]$ partitioning of matrix A .



Traditional simplex method

Note the individual bounds can be handled as normal constraints. This is the way they are handled by the **traditional (original) simplex method (OSM)**. As such, they contribute to the number of constraints (m). It is worth knowing that while the number of iterations needed to solve an LP problem is unknown in advance (in practice, it is a linear function of m), the work per iteration can be determined. Since OSM performs a complete tableau transformation in each step and it involves all matrix entries mn multiplications and additions are performed per iteration.

The above remark clearly shows the advantage of handling the individual bounds algorithmically and not as normal constraints.

Even if the matrix is sparse at the beginning, transformations tend to create nonzeros everywhere and the transformed matrix fills up. This causes serious problems.

If the original matrix has 10,000 rows, 20,000 columns and 100,000 nonzeros (moderately large problem that can be stored on midrange PCs if proper techniques are used) after some iterations the fill-in can be complete and the number of nonzeros to be stored will be 200 million which is prohibitively large. Consequently, something dramatically new is needed to be able to solve problems of this magnitude.

The solution is the **Revised Simplex Method (RSM)** with **Product Form of the Inverse (PFI)** using advanced **Sparsity Techniques**.

Optimality conditions

A nonbasic variable can be either at its lower or (finite) upper bound. Reminder: $x_B = B^{-1}(b - Nx_N)$. Substituting this x_B into $z = c_B^T x_B + c_N^T x_N$ (the objective function), we obtain $c_B^T B^{-1}(b - Nx_N) + c_N^T x_N$ which is further equal to

$$z = c_B^T B^{-1}b + (c_N^T - c_B^T B^{-1}N)x_N \tag{22}$$

The objective function can be reduced if the added part in (22) is negative. Component k of $c_N^T - c_B^T B^{-1}N$ (the multiplier of x_k in x_N) is called the **reduced cost (RC)** (transformed objective coefficient) of x_k and is denoted by d_k :

$$d_k = c_k - c_B^T B^{-1}a_k. \tag{23}$$

If B is a feasible basis, we can check whether it is optimal. The **optimality condition** of nonbasic variable x_k determines the sign of d_k and is a function of the type and status of x_k as follows:

type(x_k)	Status	d_k
0	$x_k = 0$	Immaterial
1	$x_k = 0$	≥ 0
1	$x_k = u_k$	≤ 0
2	$x_k = 0$	≥ 0
3	$x_k = 0$	$= 0$

Proof: e.g., by contradiction.

Note, for maximization problems the sign of d_k in the optimality conditions is the opposite.

Economic interpretation of the optimality conditions

Once an optimal solution has been obtained (Optimal basis \mathbf{B} and index set of nonbasic variables at upper bound J) it is worth performing **postoptimal analysis (PA)**. It serves several purposes. First of all, PA can help determine the **stability of the solution**, i.e., its **sensitivity** to small changes in the problem data. It is important because some pieces of problem data are usually inaccurate (result of observation, estimation) and we want to know whether small changes or inaccuracies would mean completely different solutions or the same solution would remain (more or less) optimal.

For easier understanding, WROG, we consider again the standard form of LP: $\min\{c^T x : Ax = b, x \geq 0\}$, i.e., all variables are of type-2. If \mathbf{B} is an optimal basis then

$$x_B = \mathbf{B}^{-1}b \geq 0 \quad (\text{feasibility})$$

$$d^T = c^T - c_B^T \mathbf{B}^{-1}A \geq 0^T \quad (\text{optimality}).$$

Here, d is the vector of reduced costs of all (including basic) variables. When data items change in the model, we check to see how the above conditions are affected. By ensuring that both feasibility and optimality are maintained, we obtain ranges for data in question within which \mathbf{B} remains an optimal basis for the modified problem.

Reminder: the definition of the reduced cost of variable x_k is

$$d_k = c_k - c_B^T \mathbf{B}^{-1} a_k. \quad (25)$$

It is easy to see that the reduced cost of a basic variable is zero. Note, $c_B^T \mathbf{B}^{-1}$ is independent of k (i.e., the same for all variables). It is denoted by π^T and is called the **simplex multiplier**:

$$\pi^T = c_B^T \mathbf{B}^{-1}. \quad (26)$$

With the simplex multiplier it is easy to compute the reduced costs. Assuming that π has been determined d_k of the nonbasic variables can be computed by

$$d_k = c_k - \pi^T a_k. \quad (27)$$

This formula simplifies for logical variables. Since the column of the i -th logical variable (which is added to constraint i) is e_i and the objective coefficient is zero, the RHS of (27) becomes $0 - \pi^T e_i = -\pi_i$. The reduced costs of logical variables at optimal solution are called **shadow prices (SPs)** and they are associated with the constraints of the LP problem. Verbally, if the LP is in standard form with \mathbf{I} included, the shadow price of constraint i is the negative of the i -th component of the simplex multiplier ($-\pi_i$). (It is always true that the reduced cost of the logical variables is $\pm\pi_i$.)

To analyze the meaning of the reduced costs and shadow prices we take a small instance of the product mix problem family.

Example 5 (from H.P. Williams) A factory can produce five types of product, P_1, \dots, P_5 . Two processes (grinding and drilling) and manpower are needed for the production. The model is:

max	Revenue $z =$	$550x_1$	$+$	$600x_2$	$+$	$350x_3$	$+$	$400x_4$	$+$	$200x_5$		
s.t.	Grinding	$12x_1$	$+$	$20x_2$	$+$	$25x_4$	$+$	$15x_5$			\leq	288
	Drilling	$10x_1$	$+$	$8x_2$	$+$	$16x_3$	$+$	$20x_4$			\leq	192
	Manpower	$20x_1$	$+$	$20x_2$	$+$	$20x_3$	$+$	$20x_4$	$+$	$20x_5$	\leq	384
		$x_1, x_2, x_3, x_4, x_5 \geq 0.$										

To convert constraints into equalities, logical variables $y_1, y_2, y_3 \geq 0$ are added to constraints 1, 2, and 3, resp. Using the simplex method an optimal solution is found: $z = 10920$ with $x_1 = 12, x_2 = 7.2, x_3 = x_4 = x_5 = 0$ and $y_1 = 0, y_2 = 14.4, y_3 = 0$. Reduced costs of the structural variables: $d_{x_1} = d_{x_2} = 0$ (basic variables), $d_{x_3} = -125, d_{x_4} = -231.25, d_{x_5} = -368.75$. RCs of the logical variables (shadow prices): $d_{y_1} = -6.25, d_{y_2} = 0, d_{y_3} = -23.75$.

Index	Structurals			Logicals				
	1	2	3	4	5	1	2	3
Solution x or y	12.00	7.20	0.00	0.00	0.00	0.00	14.40	0.00
Reduced cost d	0.00	0.00	-125.00	-231.25	-368.75	-6.25	0.00	-23.75

Example 5 highlights several important issues and it is worth analyzing it in depth.

First of all, as noted earlier, in an optimal solution no more than three (the number of constraints) products are to be produced. In our case the actual number is two.

Next, it can be seen from the tabular form of the solution that for any given variable either the solution value or the reduced cost is equal to zero. This is generally true and is known as **complementary slackness**. (The situation is slightly different if we have a nonbasic upper bounded variable $0 \leq x_k \leq u_k$ at upper bound. Then its computed nonzero reduced cost is, in fact, the reduced cost of the nonbasic slack variable of the explicit upper bound constraint $x_k \leq u_k$ which implicitly becomes $x_k + s_k = u_k, s_k \geq 0$, and $s_k = 0$ if $x_k = u_k$.)

In the example, $\mathbf{B} = \{1, 7, 2\}$, $c_B^T = [550, 0, 600]$, the optimal basis

$$\mathbf{B} = \begin{bmatrix} 12 & 0 & 20 \\ 10 & 1 & 8 \\ 20 & 0 & 20 \end{bmatrix},$$

its exact inverse

$$\mathbf{B}^{-1} = \begin{bmatrix} -0.125 & 0.000 & 0.125 \\ 0.250 & 1.000 & -0.650 \\ 0.125 & 0.000 & -0.075 \end{bmatrix}$$

and the components of the simplex multiplier (the negative of the shadow prices): $c_B^T \mathbf{B}^{-1} = \pi^T = [6.25, 0, 23.75]$.

Changes in the cost vector \mathbf{c}

Assume c_k changes to $c_k + \Delta_k$. The feasibility condition is not affected. Note: minimization problem.

If $k \in N$ (x_k **nonbasic**) the optimality condition for x_k is $\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{a}_k \leq c_k + \Delta_k$, from which

$$\Delta_k \geq \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{a}_k - c_k = -d_k. \tag{28}$$

If this condition holds (lower bound for Δ_k), the current basis remains optimal, otherwise x_k becomes profitable and is a candidate to enter the basis. For max problems: $\Delta_k \leq -d_k$ (upper bound) is defined.

If x_k is the p -th **basic** variable then the optimality conditions are

$$(\mathbf{c}_B + \Delta_k \mathbf{e}_p)^T \mathbf{B}^{-1} \mathbf{a}_j \leq c_j \quad j \in N \tag{29}$$

Since x_k is basic its reduced cost stays at zero. (29) can be expanded as $\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{a}_j + \Delta_k \mathbf{e}_p^T \mathbf{B}^{-1} \mathbf{a}_j \leq c_j$. Using notation $\alpha_j = \mathbf{B}^{-1} \mathbf{a}_j$ and noting that $\mathbf{e}_p^T \alpha_j = \alpha_{pj}$, we can write

$$\Delta_k \alpha_{pj} \leq c_j - \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{a}_j = d_j \quad j \in N.$$

Note, there can be positive and negative α_{pj} values among $j \neq k$. Therefore, if $\alpha_{pj} > 0$ then

$$\Delta_k \leq \left\{ \frac{d_j}{\alpha_{pj}} \right\} \Rightarrow \Delta_k \leq \min_{j:\alpha_{pj}>0} \left\{ \frac{d_j}{\alpha_{pj}} \right\} \tag{30}$$

and if $\alpha_{pj} < 0$ then

$$\Delta_k \geq \left\{ \frac{d_j}{\alpha_{pj}} \right\} \Rightarrow \Delta_k \geq \max_{j:\alpha_{pj}<0} \left\{ \frac{d_j}{\alpha_{pj}} \right\}. \tag{31}$$

(30) and (31) together define the following range of "indifference" for Δ_k regarding optimality:

$$\max_{j:\alpha_{pj}<0} \left\{ \frac{d_j}{\alpha_{pj}} \right\} \leq \Delta_k \leq \min_{j:\alpha_{pj}>0} \left\{ \frac{d_j}{\alpha_{pj}} \right\}. \tag{32}$$

Lower range = $-\infty$ if no $\alpha_{pj} < 0$, upper range = $+\infty$ if no $\alpha_{pj} > 0$. Verbally, lower range = the maximum of the negative ratios, upper range = the minimum of the positive ratios.

Recommendation: In case of a maximization problem, apply the verbal version of the rule which is valid in both cases.

The α_{pj} -s are the elements of the p -th row of the transformed tableau. If they are not available they can be reproduced by the help of \mathbf{B}^{-1} . Let α^p (p in superscript!) denote the p -th row in question: $\alpha^p = \mathbf{e}_p^T \mathbf{B}^{-1} \mathbf{A}$. Using notation $\mathbf{v}^T = \mathbf{e}_p^T \mathbf{B}^{-1}$, $\alpha^p = \mathbf{v}^T \mathbf{A}$. An individual α_{pj} lies in the intersection of row p and column j and can be computed from $\alpha_{pj} = \mathbf{v}^T \mathbf{a}_j$.

In our example (max!), P3, P4, and P5 are nonbasic at the optimal solution. Taking P3, its RC is -125 . Recalling the definition of RC, $d_3 = c_3 - \pi^T \mathbf{a}_3 = 350 - 475 = -125$. It is evident that P3 is underpriced. By (28) we can say how much its sales price should be increased to make it profitable. This value is the negative of the reduced cost of the variable at optimal solution: 125 .

If the price was increased beyond $350 + 125 = 475$ then d_3 would become positive, thus profitable for inclusion in the solution. Similar analysis can be applied to P4 and P5. Since the nonbasic variables are already underpriced, their further decrease will not have any effect on the optimal solution. Therefore, we can conclude that

Variable	Cost coeff.	Lower range	Upper range
P3	350	$-\infty$	475.00
P4	400	$-\infty$	631.25
P5	200	$-\infty$	568.75

To determine ranges for the objective coefficients of the basic variables we apply (32) and obtain

Variable	Cost coeff.	Lower range	Upper range
P1	550	500.0	600.0
P2	600	550.0	683.3

There is something very important here. If, for instance, the objective coefficient of P1 moves away from its current value of 550 within the range, the basic variables will not change at all, they stay at $x_1 = 12$ and $x_2 = 7.2$ in the optimal solution. If it goes beyond its upper range, some further analysis is needed. P1 may become so profitable that it should be produced in such a quantity that nothing else can be produced. If it goes below its lower range then some other variable will become more profitable and can replace P1 in the basis.

Remark: The above analysis is valid if only one coefficient is changed at a time.

Changes in the RHS vector \mathbf{b}

Suppose, the p -th component of \mathbf{b} is changed by Δ_p to $b_p + \Delta_p$, or, equivalently, \mathbf{b} is changed to $\mathbf{b} + \Delta_p \mathbf{e}_p$. The optimality conditions are not affected. Therefore, if we want to determine the range of Δ_p within which the current basis remains optimal we only have to check the feasibility constraints:

$$\mathbf{B}^{-1} (\mathbf{b} + \Delta_p \mathbf{e}_p) \geq \mathbf{0}. \tag{33}$$

If the columns of \mathbf{B}^{-1} are denoted by β_1, \dots, β_m , then the p -th column is $\beta_p = [\beta_{1p}, \dots, \beta_{mp}]^T$. (33) can be rewritten as

$$\mathbf{B}^{-1} \mathbf{b} + \Delta_p \mathbf{B}^{-1} \mathbf{e}_p = \mathbf{x}_B + \Delta_p \beta_p \geq \mathbf{0},$$

or in coordinate form:

$$x_{Bi} + \Delta_p \beta_{ip} \geq 0 \quad \text{for } i = 1, \dots, m.$$

Similarly to the case of changes in \mathbf{c} , there can be positive and negative β_{ip} coefficients that together define the possible range of Δ_p such that the optimal basis remains unchanged:

$$\max_{i:\beta_{ip}>0} \left\{ \frac{-x_{Bi}}{\beta_{ip}} \right\} \leq \Delta_p \leq \min_{i:\beta_{ip}<0} \left\{ \frac{-x_{Bi}}{\beta_{ip}} \right\}. \tag{34}$$

While \mathbf{B} does not change if Δ_p varies in this interval, the solution (\mathbf{x}_B), of course, changes (see (33)).

Exactly the same procedure applies to maximization problems because min/max has nothing to do with feasibility.

If a logical variable is zero in the optimal solution then the corresponding constraint is satisfied as an equality. In other words, this constraint blocks some variables to take more favourable values thus it prevents a decrease in the objective function.

To better understand the shadow prices it is necessary to investigate the optimal value of an LP problem as a function of the RHS. For notational simplicity, we consider the standard LP $z(\mathbf{b}) = \min\{\mathbf{c}^T\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$. If we have an optimal basis \mathbf{B} then $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$ and thus $z(\mathbf{b}) = \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b}$. Using the definition of the simplex multiplier, $z(\mathbf{b}) = \boldsymbol{\pi}^T\mathbf{b}$. If we have a modified RHS vector $\bar{\mathbf{b}}$ for which the same \mathbf{B} is optimal then $z(\bar{\mathbf{b}}) = \boldsymbol{\pi}^T\bar{\mathbf{b}}$ ($\boldsymbol{\pi}$ is unchanged since it is defined only by the basis). Now, $z(\bar{\mathbf{b}}) - z(\mathbf{b}) = \boldsymbol{\pi}^T(\bar{\mathbf{b}} - \mathbf{b})$ from which

$$z(\bar{\mathbf{b}}) = z(\mathbf{b}) + \boldsymbol{\pi}^T(\bar{\mathbf{b}} - \mathbf{b}) \tag{35}$$

Assume, shadow price of row i is positive ($-\pi_i > 0 \Rightarrow \pi_i < 0$) and consider a $\bar{\mathbf{b}}$ vector that differs from \mathbf{b} just in the i -th component in such a way that $\bar{b}_i = b_i + \Delta_i \Rightarrow \Delta_i = \bar{b}_i - b_i$. In this case (35) reduces to

$$z(\bar{\mathbf{b}}) = z(\mathbf{b}) + \pi_i\Delta_i. \tag{36}$$

This equation suggests that if we increase b_i by Δ_i (i.e., $\Delta_i > 0$) the objective value will decrease by $\pi_i\Delta_i$ as long as the same basis remains optimal.

In our example (maximization!), $d_{y1} = -6.25$. It can be interpreted as one hour extra grinding capacity can increase the revenue by £6.25. Similar interpretation is valid for the manpower constraint ($d_{y3} = -23.75$).

The logical variable of the drilling constraint is in the optimal basis with a value of 14.4. It means that we have 14.4 hours of unused drilling capacity. Clearly, increasing this capacity would not result in an improvement of the objective function. Therefore, we can say that the upper range of this RHS coefficient is $+\infty$. The lower range is its current value less the slack capacity: $192.0 - 14.4 = 177.6$. Since this logical variable is in the basis its reduced cost is zero. If we formally apply (36) to this case we obtain zero improvement as expected.

If the i -th original constraint is an equality then its logical variable is of type-0. It never becomes basic and the sign of its reduced cost tells us whether the increase or the decrease of the corresponding b_i can contribute to the improvement of the objective value.

The ranges for binding constraints (logical variable nonbasic) can be determined by (34). In this example they turn out to be

Constraint	RHS	Lower range	Upper range
Grinding	288	230.4	384.0
Manpower	384	288.0	406.1

Remark (as before): The above analysis is valid if only one RHS coefficient is changed at a time.

Stability of a solution

We say that a solution is **stable** if it is not sensitive to small changes in the cost coefficients and RHS values. In other words, for a stable solution the computed ranges are relatively large.

Using this terminology, we can say that the solution of the product mix example is stable. The importance of it becomes clear if we notice that problem data may be inaccurate or subject to changes (market price, machine breakdown, etc.). If a company commits itself to some production structure (P1 and P2 only) it wants to be certain that minor disturbances will not invalidate the pattern of production (still P1 and P2 are to be produced, maybe in slightly modified quantities). If, e.g., the range of the grinding capacity (288) had been very small, like 287 to 289, management would have to be very careful in applying the suggested solution because it clearly would depend critically on the accuracy of the figure included in the model.

Building stable models

The quantities of certain resources can be increased at some extra costs. E.g., if we have a product mix problem with a "hard" resource constraint in the form of $\sum_j a_j x_j \leq b$, it can be softened up by introducing a variable, v , for extra supply and assign some cost c_v to it in the objective function, like $\max z = \mathbf{c}^T\mathbf{x} - c_v v$, subject the modified "soft" constraint: $\sum_j a_j x_j - v \leq b$ and the other original constraints.

Sometimes, users are more interested in stable than optimal solutions.

Improving a nonoptimal BFS

If the optimality conditions are not satisfied we can attempt to improve the current BFS.

If $\text{type}(x_k) = 1$, $k \in N$, x_k can be at one of its bounds (0, or u_k). Index set of nonbasic variables at upper bound is J .

Given a feasible basis \mathbf{B} and J , from (21), the corresponding basic feasible solution is:

$$\mathbf{x}_B = \mathbf{B}^{-1} \left(\mathbf{b} - \sum_{j \in J} \mathbf{a}_j u_j \right).$$

If the reduced cost of a nonbasic variable x_q violates the optimality condition we can try to include x_q in the basis with the hope of improving the value of the objective function (it cannot be guaranteed, see degeneracy). For simplicity, assume $x_q = 0$ (nonbasic at lower bound). If its value is increased, the main equations (see (20) on page 37) still have to be satisfied. Additionally, we want to increase the value of x_q in such a way that

- (i) x_q itself takes a feasible value and
- (ii) all the basic variables remain feasible.

The main equations at the beginning: $\mathbf{B}\mathbf{x}_B = \mathbf{b} - \sum_{j \in J} \mathbf{a}_j u_j$.

We start moving x_q away from its current value of zero. The displacement of x_q is denoted by t . Now the main equations become: $\mathbf{B}\mathbf{x}_B(t) + t\mathbf{a}_q = \mathbf{b} - \sum_{j \in J} \mathbf{a}_j u_j$, or

$$\mathbf{x}_B(t) = \mathbf{B}^{-1} \left(\mathbf{b} - \sum_{j \in J} \mathbf{a}_j u_j \right) - t\mathbf{B}^{-1}\mathbf{a}_q, \quad (37)$$

where $\mathbf{x}_B(t)$ refers to the dependance on t . This form shows that the value of each basic variable is a linear function of t . Introducing notations

$$\mathbf{x}_B(0) = \mathbf{B}^{-1} \left(\mathbf{b} - \sum_{j \in J} \mathbf{a}_j u_j \right) \text{ and } \boldsymbol{\alpha}_q = \mathbf{B}^{-1}\mathbf{a}_q$$

and indicating the dependence on t , (37) can be rewritten as

$$\mathbf{x}_B(t) = \mathbf{x}_B(0) - t\boldsymbol{\alpha}_q, \quad (38)$$

or for the i -th component

$$x_{Bi}(t) = x_{Bi}(0) - t\alpha_{iq},$$

which is a linear function of t . Now we want to determine the maximum value $t \geq 0$ such that $x_{Bi}(t)$ remains feasible. It is evident that $x_{Bi}(t)$ is decreasing if $\alpha_{iq} > 0$ and increasing if $\alpha_{iq} < 0$. We investigate the two cases separately.

Generalized Ratio Test

$\alpha_{iq} > 0$: Since $x_{Bi}(0)$ is feasible it can decrease until it reaches its lower bound of zero (even if $\text{type}(x_{Bi}) = 0$): $x_{Bi}(t) = x_{Bi}(0) - t\alpha_{iq} = 0$ from where

$$t = \frac{x_{Bi}(0)}{\alpha_{iq}} \text{ (traditional ratio test).}$$

Obviously, $t \geq 0$.

$\alpha_{iq} < 0$: Now $x_{Bi}(t)$ is increasing. This is good unless $\text{type}(x_{Bi})$ is 0 or 1. In these cases x_{Bi} can reach its upper bound u_{Bi} (note, the upper bound for a type-0 variable is 0), $x_{Bi}(t) = x_{Bi}(0) - t\alpha_{iq} = u_{Bi}$, from which

$$t = \frac{x_{Bi}(0) - u_{Bi}}{\alpha_{iq}} \text{ (ratio with upper bound).}$$

Here, $t \geq 0$ holds because $0 \leq x_{Bi}(0) \leq u_{Bi}$ and $\alpha_{iq} < 0$.

From the above it follows that if

- $\text{type}(x_{Bi}) = 3 \Rightarrow x_{Bi}$ can take any value \Rightarrow it does not block $t \Rightarrow$ can be left out of ratio test.
- $\text{type}(x_{Bi}) = 0 \Rightarrow x_{Bi} = 0$ (feasibility). If $\alpha_{iq} = 0$ it is not involved in ratio test, otherwise it will define a zero valued ratio.

The above investigation has to be performed for each basic variable x_{Bi} , $i = 1, \dots, m$. The different values for t are denoted by t_i . If the minimum of them

$$t_p = \min_i \{t_i\}$$

is taken as the displacement of the incoming variable x_q then all basic variables remain feasible. The p -th basic variable (x_{Bp}) that defined the minimum ratio leaves the basis at feasible level. As we see later, the new value of the objective function will be better (if $t_p > 0$) or not worse (if $t_p = 0$). In the latter case degeneracy prevents a change (= zero progress).

If $\text{type}(x_q) = 1$ ($0 \leq x_q \leq u_q$) and for the minimum ratio $t_p > u_q$ then x_q cannot enter the basis at feasible level. The only possibility is to set it to its upper bound (index q joins set J) and leave the basis unchanged. The solution is, however, transformed (updated) according to (38) as

$$\hat{\mathbf{x}}_B = \mathbf{x}_B(u_q) = \mathbf{x}_B(0) - u_q\boldsymbol{\alpha}_q. \quad (39)$$

We summarize our findings for the generalized ratio test in a tabular form. First, the assumptions:

- (i) We are given a nonoptimal basic feasible solution \mathbf{x}_B and
- (ii) a nonbasic improving candidate x_q is moving away from its current value of zero in positive direction.

The following ratios are defined:

type(x_{Bi})	$\alpha_{iq} > 0$ (Type-A)	$\alpha_{iq} < 0$ (Type-B)
0	$t_i = 0$	$t_i = 0$
1	$t_i = \frac{x_{Bi}(0)}{\alpha_{iq}}$	$t_i = \frac{x_{Bi}(0) - u_{Bi}}{\alpha_{iq}}$
2	$t_i = \frac{x_{Bi}(0)}{\alpha_{iq}}$	—
3	—	—

Let $t_p = \min_i \{t_i\}$ with p being the subscript defining the minimum ratio. If $t_p \geq u_q$ the "at bound" status of x_q changes (bound swap) and there are no incoming and outgoing variables (basis remains unchanged).

To find the new value of the selected candidate x_q , first $t_p = \min_i \{t_i\}$ is determined which is then compared with u_q (the upper bound of x_q) that can be finite or infinite. Ultimately, the minimum of t_p and u_q is taken which is denoted by θ : $\theta = \min\{u_q, t_p\} = \min\{u_q, \min_i \{t_i\}\}$. The new value of x_q

$$\hat{x}_q = x_q + \theta \quad (40)$$

and the solution is transformed as

$$\hat{\mathbf{x}}_B = \mathbf{x}_B(\theta) = \mathbf{x}_B - \theta\boldsymbol{\alpha}_q. \quad (41)$$

θ is called the **steplength** of the iteration.

If there is an entering variable x_q (basis change, not a bound swap) then the p -th basic variable (x_{B_p}) leaves the basis. If t_p was a type-B ratio then x_{B_p} becomes nonbasic at upper bound, otherwise at zero.

If the incoming candidate x_q comes in from upper bound its displacement, t , must be negative. Similarly, a type-3 variable can be a candidate with negative value. In these two cases all the above development is still valid with $-\alpha_{iq}$ replacing α_{iq} in the formulae. Note, in this way $-t_i$ is calculated and if the minimum of $-t_i$ s is taken it determines $-t_p$. Finally $-\theta = \min\{-t_p, u_q\}$. Having obtained θ we can use (40) and (41) to compute the value of the incoming variable and the new solution.

The change in the value of the objective function (z) is determined by two factors: the displacement of the incoming variable (θ) and the rate of change (d_q). The value of z after the transformations can be derived from (22) and (23):

$$\hat{z} = z + \theta d_q$$

This formula is valid whether there is a basis change or a bound swap.

A basic solution is: $\mathbf{x}_B = [x_{B_1}, \dots, x_{B_p}, \dots, x_{B_m}]^T$. The ordered set of subscripts of basic variables, $B = \{B_1, \dots, B_p, \dots, B_m\}$, is often referred to as **basis heading**.

E.g., if x_8, x_4, x_7 and x_2 are the basic variables (in this order) then $\mathbf{x}_B = [x_8, x_4, x_7, x_2]^T$ and $B = \{8, 4, 7, 2\}$.

If basis changes: $\hat{B} = \{B_1, \dots, B_{p-1}, q, B_{p+1}, \dots, B_m\}$, new BFS: $\mathbf{x}_{\hat{B}} = [\hat{x}_{B_1}, \dots, \hat{x}_{B_{p-1}}, \hat{x}_q, \hat{x}_{B_{p+1}}, \dots, \hat{x}_{B_m}]^T$.

Example 6 Assume, we are given a BFS and a type-2 incoming variable ($x_q = 0$) with its transformed column. Determine the ratios, the value of the incoming variable and the new BFS.

i	x_{B_i}	$type(x_{B_i})$	u_{B_i}	α_{iq}	t_i
1	0	0	0	0	—
2	5	1	10	5	5/5 = 1
3	2	1	4	-1	(2 - 4)/(-1) = 2
4	6	2	∞	2	6/2 = 3
5	2	2	∞	-3	—
6	4	3	∞	1	—

$t_p = \min\{1, 2, 3\} = 1$ and $p = 2$ (the subscript defining the minimum ratio). Since the incoming variable is of type-2, its upper bound is $+\infty \Rightarrow \theta = t_p = 1$. (Note, despite degeneracy of the solution, the minimum ratio is positive.) The new solution:

$$\hat{\mathbf{x}}_B = \mathbf{x}_B(0) - 1 \times \boldsymbol{\alpha}_q = \begin{bmatrix} 0 \\ 5 \\ 2 \\ 6 \\ 2 \\ 4 \end{bmatrix} - \begin{bmatrix} 0 \\ 5 \\ -1 \\ 2 \\ -3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \\ 4 \\ 5 \\ 3 \end{bmatrix}.$$

Variable x_{B_2} has become 0 and leaves the basis. It is replaced by $\hat{x}_q = x_q + \theta = 0 + 1 = 1$ as basic variable x_{B_2} .

$$\hat{B} = B \setminus \{B_2\} \cup \{q\} \text{ and } \mathbf{x}_{\hat{B}} = [0, 1, 3, 4, 5, 3]^T.$$

Example 7 Assume again, we are given a BFS, a type-1 incoming variable, $0 \leq x_q \leq 2$ ($= u_q$) which is nonbasic at lower bound and its transformed column. Determine the ratios, the value of the incoming variable and the new BFS.

i	x_{B_i}	$type(x_{B_i})$	u_{B_i}	α_{iq}	t_i
1	0	0	0	0	—
2	6	1	10	1	6/1 = 6
3	1	1	4	-1	(1 - 4)/(-1) = 3
4	8	2	∞	2	8/2 = 4

$t_p = \min\{6, 3, 4\} = 3$ and $p = 3$.
Now, $\theta = \min\{t_p, u_q\} = \min\{3, 2\} = 2 \Rightarrow t_p > u_q \Rightarrow$ bound swap $\Rightarrow \hat{x}_q = x_q + \theta = 0 + 2 = 2$, the subscript of the incoming variable, q , is added to J and

$$\hat{\mathbf{x}}_B = \mathbf{x}_B(0) - 2\boldsymbol{\alpha}_q = \begin{bmatrix} 0 \\ 6 \\ 1 \\ 8 \end{bmatrix} - 2 \begin{bmatrix} 0 \\ 1 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 3 \\ 4 \end{bmatrix}.$$

It is easy to verify that $\hat{\mathbf{x}}_B$ is feasible.

Since there is no basis change, $\hat{B} = B$ and $\mathbf{x}_{\hat{B}} = \hat{\mathbf{x}}_B$.

Example 8 Assumptions are the same as in Example 7 but the incoming variable (x_q) comes in from its upper bound of 2 ($x_q = 2$).

i	x_{B_i}	$type(x_{B_i})$	u_{B_i}	α_{iq}	$-t_i$
1	0	0	0	0	—
2	6	1	10	1	(6 - 10)/(-1) = 4
3	1	1	4	-1	1/1 = 1
4	8	2	∞	2	—

$-t_p = \min\{4, 1\} = 1$ and $p = 3$.
Now, $-\theta = \min\{-t_p, u_q\} = \min\{1, 2\} = 1, \Rightarrow \theta = -1$.

$$\hat{\mathbf{x}}_B = \mathbf{x}_B(0) - (-1)\boldsymbol{\alpha}_q = \begin{bmatrix} 0 \\ 6 \\ 1 \\ 8 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \\ 0 \\ 10 \end{bmatrix}.$$

Variable x_{B_3} has become 0 and leaves the basis. It is replaced by x_q with its new value of $\hat{x}_q = x_q + \theta = 2 + (-1) = 1$. It is easy to verify that $\hat{\mathbf{x}}_B$ is feasible.

$$\hat{B} = B \setminus \{B_3\} \cup \{q\} \text{ and } \mathbf{x}_{\hat{B}} = [0, 7, 1, 10]^T.$$

In case we have a BFS (with B and J) to the LP problem, the **simplex method with all types of variables (SMA)** can be described (verbally) as follows:

1. Check optimality conditions (see definition of d_k and (24)).
2. If satisfied \Rightarrow Solution is optimal, terminate.
3. Select a nonbasic variable x_q violating the optimality condition to enter the basis and compute $\alpha_q = \mathbf{B}^{-1}\mathbf{a}_q$.
4. Perform **generalized ratio test** (to account for all types of variables in the basis) with \mathbf{x}_B and α_q to determine the outgoing basic variable x_{Bp} or a bound swap.
5. If no ratio has been defined (the ratio is technically $+\infty$) then

$$\text{if type}(x_q) \begin{cases} \neq 1, & \Rightarrow \text{Solution is unbounded, terminate,} \\ = 1, & \Rightarrow \text{Swap bound status of } x_q, \text{ go to 7.} \end{cases}$$

6. If p is defined (no bound swap) then replace the p -th basic variable x_{Bp} by x_q , compute new \mathbf{B}^{-1} and \mathbf{x}_B , go to 1, else swap bound status of x_q , go to step 7.
7. Update solution according to (39), go to step 1.

The way SMA was presented corresponds to the framework of the revised simplex method (RSM). Therefore, we will refer to it as **RSMA** (revised simplex method with all types of variables).

Obtaining a first feasible solution for the general LP problem requires a somewhat sophisticated Phase-I procedure that is beyond the scope of this course.

Brief computational analysis of the steps of RSMA

- Step 1: Requires the computation of d_k by \mathbf{B}^{-1} ; it may involve heavy or moderate computing depending on the way \mathbf{B}^{-1} is stored and computed/updated.
- Step 2: Negligible.
- Step 3: Heavy computing.
- Step 4: Heavy logic, some computing.
- Step 5: Negligible.
- Step 6: Simple or heavy, depending on the way \mathbf{B}^{-1} is stored and computed/updated.
- Step 7: Moderate.

Note: OSM can be adjusted to handle simple bounds algorithmically rather than as extra constraints.

Comparison of OSM and RSMA

	OSM	RSMA
d_k	Available	To be computed
α_q	Available	To be computed
\mathbf{B}^{-1}	Not needed	To be computed/updated
Transformed tableau	To be updated	Not needed

Which is better for solving large scale problems?

Simplex example

$$\begin{aligned} \min z &= -4x_1 - 2x_2 - 12x_3 \\ \text{s.t.} \quad &x_1 + 3x_2 - 2x_3 \leq 12 \\ &-3x_1 + x_2 + 2x_3 = 0 \\ &-2 \leq 4x_1 - x_2 + x_3 \leq 6 \\ &x_1, x_2 \geq 0, 0 \leq x_3 \leq 1 \end{aligned}$$

First, convert every constraint into an equality and add an appropriate logical variable to it.

$$\begin{aligned} \min z &= -4x_1 - 2x_2 - 12x_3 - 2x_4 - 2x_5 + y_1 = 12 \\ \text{s.t.} \quad &x_1 + 3x_2 - 2x_3 + x_4 + y_1 = 12 \\ &-3x_1 + x_2 + 2x_3 + x_5 + y_2 = 0 \\ &4x_1 - x_2 + x_3 + y_3 = 6 \\ &x_1, x_2 \geq 0, 0 \leq x_3 \leq 1, y_1 \geq 0, y_2 = 0, 0 \leq y_3 \leq 8 \end{aligned}$$

Starting basic feasible solution: $y_1 = 12, y_2 = 0$ and $y_3 = 6$. The problem in tabular form (UB is the column of upper bounds of basic variables):

	x_1	x_2	x_3	y_1	y_2	y_3	\mathbf{x}_B	UB
\mathbf{u}	∞	∞	1	∞	0	8		
y_1	1	3	-2	1			12	∞
y_2	-3	1	2		1		0	0
y_3	4	-1	1			1	6	8
\mathbf{d}^T	-4	-2	-12	0	0	0	0	
incoming: $x_3, \min\{\frac{6}{2}, \frac{6}{1}\} = 0, p = 2$								
y_1	-2	4	0	1	1	0	12	∞
x_3	$-\frac{3}{2}$	$\frac{1}{2}$	1	0	$\frac{1}{2}$	0	0	1
y_3	$\frac{11}{2}$	$-\frac{3}{2}$	0	0	$-\frac{1}{2}$	1	6	8
\mathbf{d}^T	-22	4	0	0	6	0	0	
incoming: $x_1, \min\{\frac{0-1}{-3/2}, \frac{6}{11/2}\} = \frac{2}{3}, p = 2, x_3$ leaves at ub of 1								

	x_1	x_2	x_3	y_1	y_2	y_3	x_B	UB
u	∞	∞	1	∞	0	8		
y_1	0	$\frac{10}{3}$	$-\frac{4}{3}$	1	$\frac{1}{3}$	0	$\frac{40}{3}$	∞
x_1	1	$-\frac{1}{3}$	$-\frac{2}{3}$	0	$-\frac{1}{3}$	0	$\frac{2}{3}$	∞
y_3	0	$\frac{1}{3}$	$\frac{11}{3}$	0	$\frac{4}{3}$	1	$\frac{7}{3}$	8
d^T	0	$-\frac{10}{3}$	$-\frac{44}{3}$	0	$-\frac{4}{3}$	0	$\frac{44}{3}$	
incoming: $x_2, \min\{\frac{40/3}{10/3}, \frac{7/3}{1/3}\} = 4, p = 1$								
x_2	0	1	$-\frac{2}{3}$	$\frac{3}{10}$	$\frac{1}{10}$	0	4	∞
x_1	1	0	$-\frac{4}{5}$	$\frac{1}{10}$	$-\frac{3}{10}$	0	2	∞
y_3	0	0	$\frac{19}{5}$	$-\frac{1}{10}$	$\frac{13}{10}$	1	1	8
d^T	0	0	-16	1	-1	0	28	

Optimal solution: $z = -28, x_1 = 2, x_2 = 4, x_3 = 1, y_1 = 1, y_2 = 0, y_3 = 0$.

Product form of the inverse (PFI)

The product form of the inverse is the key to the success of large scale optimization. In the simplex method if we perform a basis change then the new basis differs from the old one just in one column. By knowing the inverse of the old basis it is easy to determine the inverse of the new one (there is no need to recompute it from scratch).

Basis (now the basic vectors are denoted by b_1, \dots, b_m):

$$B = (b_1, \dots, b_{p-1}, b_p, b_{p+1}, \dots, b_m)$$

Neighbouring basis after b_p is replaced by a :

$$B_a = (b_1, \dots, b_{p-1}, a, b_{p+1}, \dots, b_m)$$

Since a is an m dimensional vector it can be written as a linear combination of the basic vectors: $a = \sum_{i=1}^m t_i b_i$ (or $a = Bt \Rightarrow t = B^{-1}a$) from which b_p can be expressed:

$$b_p = \frac{1}{t_p} a - \sum_{i \neq p} \frac{t_i}{t_p} b_i \tag{42}$$

Vectors on the RHS of (42) form B_a . B_a is nonsingular if and only if $t_p \neq 0$.

b_p in the new basis is $b_p = B_a \eta$, where

$$\eta = \left[-\frac{t_1}{t_p}, \dots, -\frac{t_{p-1}}{t_p}, \frac{1}{t_p}, -\frac{t_{p+1}}{t_p}, \dots, -\frac{t_m}{t_p} \right]^T$$

If we define the following matrix

$$E = [e_1, \dots, e_{p-1}, \eta, e_{p+1}, \dots, e_m]$$

then all the original basis vectors can be expressed in terms of the new basis as

$$B = B_a E. \tag{43}$$

E is a very simple matrix. It differs from the unit matrix in just one column. This column is often referred to as η (eta) vector while the matrix is called Elementary Transformation Matrix (ETM):

$$E = \begin{bmatrix} 1 & & & \eta_1 & & \\ & \ddots & & \vdots & & \\ & & & \eta_p & & \\ & & & \vdots & \ddots & \\ & & & \eta_m & & 1 \end{bmatrix}$$

To fully represent E only the eta vector and its position in the unit matrix are to be recorded.

Looking at the structure of E it is easy to see why in (43) it reproduces all but the p -th columns of B while for p it gives b_p by virtue of (42).

Taking the inverse of both sides of (43) we get $B^{-1} = E^{-1} B_a^{-1}$ and multiplying by E we obtain:

$$B_a^{-1} = EB^{-1}. \tag{44}$$

Verbally: the inverse of the new basis can be obtained from the old inverse by premultiplying it by an ETM that is derived from the basis change: the incoming vector a determines $t = B^{-1}a$ and the outgoing vector (the p -th basic vector b_p) determines η by defining the position of the eta vector in the unit matrix

$$\eta_i = -\frac{t_i}{t_p} \text{ for } i \neq p \text{ and } \eta_p = \frac{1}{t_p},$$

where p is **pivot position** and t_p is **pivot element**.

Operation defined in (44) is called **updating** B^{-1} . If rows of B^{-1} are denoted by $\rho^i, i = 1, \dots, m$, then updating B^{-1} means the following. Compute new row p : $\bar{\rho}^p = (1/v_p)\rho^p$, determine the other new rows: $\bar{\rho}^i = \rho^i - v_i \bar{\rho}^p$ for $i = 1, \dots, m, i \neq p$. B_a^{-1} consists of rows $\bar{\rho}^1, \dots, \bar{\rho}^m$.

Operations of the simplex method are described in terms of B^{-1} . If B^{-1} is kept in **product form** (PFI, see soon) factors grow in number as a new ETM is created at each basis change. Therefore, operations with PFI become slow and increasingly inaccurate. Thus, the form has to be recomputed regularly or occasionally. This operation is called **reversion**. Some desirable features:

- Speedy operation to cause minimum delay in optimization.
- Generation of sparse PFI to speed up subsequent operations in the simplex.
- High numerical accuracy to restore the accuracy of the iterations and the LP solution.

Computational aspects of PFI

While B^{-1} is unique, the product form is not. There is a freedom in the order the ETMs are created. This freedom can be utilized to create sparse eta vectors. The importance of this observation can be illustrated by the following example:

$$B = [b_1 \ b_2 \ b_3 \ b_4 \ b_5] = \begin{bmatrix} 1 & & & & 1 \\ 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \end{bmatrix}$$

If we form the eta vectors by pivoting down the main diagonal (i.e., taking $b_1, b_2 \dots$ in this order), we obtain

η_1	η_2	η_3	η_4	η_5
1				-0.5
-1	1			0.5
	-1	1		-0.5
		-1	1	0.5
			-1	0.5

The pivot positions are boxed. Note, altogether 13 nonzeros represent the inverse.

Obtaining sparse representation of B^{-1}

Typical nonzero pattern of a lower triangular matrix:

x					
x	x				
	x	x			
x		x	x		
		x		x	
			x		x

Top left position: the only nonzero in the row.
 Bottom right position: the only nonzero in the column.

Symbolic triangularization:

Set up row (column) counts of nonzeros in rows (columns)

2				x	x	
2	x		x			
2		x	x			
1		x				
2	x			x		
3	x	x			x	
	3	3	2	1	1	2

The first eta vector is formed from column 2 pivoting on position 4. In this way the remaining columns need not be transformed (updated) as they have zeros in the pivot position of the ETM created from column 2.

If we choose the reverse order of pivoting (up the main diagonal), the outcome is

η_1	η_2	η_3	η_4	η_5
-1	1	-1	1	0.5
			1	-0.5
		1	-1	0.5
	1	-1	1	-0.5
1	-1	1	-1	0.5

This time 19 nonzeros represent the inverse. The explicit form of B^{-1} is fully dense with 25 nonzeros:

$$B^{-1} = \begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix}$$

Operations with B^{-1} are faster if the representation is shorter.

For real large and sparse problems the difference in representation can be far more dramatic. Calculating and storing the explicit inverse of a 10,000 x 10,000 matrix is practically impossible, while its product form is probably very modest in terms of computation, storage and operations with it.

Deleting column 2 and row 4 the row counts are updated by decrementing the row counts of rows where column 2 had a nonzero. The deletion is formally indicated by a "-" (dash) for the counts and 'o' replacing 'x' in the matrix. The pivot position is denoted by \otimes_1 (the first pivot).

2					x	x
2	x		x			
1		o	x			
-		\otimes_1				
2	x			x		
2	x	o				x
	3	-	2	1	1	2

Now, there is a new count value of 1 in row 3. Taking the (row3, column3) position as pivot we obtain the next ETM for the product form. After deleting row 3 and column 3 we update the counts:

2					x	x
1	x		o			
-		o	\otimes_2			
-		\otimes_1				
2	x			x		
2	x	o				x
	3	-	-	1	1	2

Continuing in a similar fashion, finally we obtain:

-				\otimes_6	o
-	\otimes_3		o		
-		o	\otimes_2		
-		\otimes_1			
-	o			\otimes_4	
-	o	o			\otimes_5

If at some step several row counts are equal to 1 then any of them can be chosen.

Additional remarks:

- If a matrix is fully triangularizable the above procedure will find one lower triangular form (which may not be unique).
- If a row count becomes zero other than for a deleted row then this row contains no more nonzero elements. In this case we say that the matrix is **structurally singular**.
- If all row counts of free (unused) rows are > 1 then the matrix cannot be triangularized symbolically.
- Triangularization can be done by using column counts. This version proceeds from the bottom right corner upwards on the main diagonal (see figure of a triangular matrix).
- If rowwise triangularization gets stuck we can turn to finding lower triangular factors using column counts.
- If a matrix cannot be triangularized it is still important to identify the largest possible triangular part(s).

DUALITY IN LINEAR PROGRAMMING

Duality is a very general principle that has relevance also to LP. To any LP problem (which we call **primal LP**)

$$(P2) \quad \min \quad \mathbf{c}^T \mathbf{x} \tag{45}$$

$$\text{s.t.} \quad \mathbf{Ax} = \mathbf{b}, \tag{46}$$

$$\mathbf{x} \geq \mathbf{0}, \tag{47}$$

there is an associated LP problem, called **dual**, in the form of

$$(D2) \quad \max \quad \mathbf{b}^T \mathbf{y}$$

$$\text{s.t.} \quad \mathbf{A}^T \mathbf{y} \leq \mathbf{c}.$$

Note, dual variables in \mathbf{y} are unrestricted in sign (free variables). The dual of the dual is the primal.

Weak duality theorem. Given an arbitrary primal feasible solution \mathbf{x} to (P2) and any dual feasible solution to (D2), then for the corresponding objective values relation $\mathbf{b}^T \mathbf{y} \leq \mathbf{c}^T \mathbf{x}$ holds.

Proof. As \mathbf{x} and \mathbf{y} are solutions of their respective problems and $\mathbf{x} \geq \mathbf{0}$, we have

$$\mathbf{b} = \mathbf{Ax} \quad \text{and} \quad \mathbf{y}^T \mathbf{A} \leq \mathbf{c}^T.$$

Multiplying the left hand side equation by \mathbf{y}^T , the right-hand side inequality by \mathbf{x} and comparing the two, we obtain

$$\mathbf{y}^T \mathbf{b} = \mathbf{y}^T \mathbf{Ax} \quad \text{and} \quad \mathbf{y}^T \mathbf{Ax} \leq \mathbf{c}^T \mathbf{x}.$$

Meaning: as dual objective is maximized, it is bounded from above by any primal feasible solution. Equality holds when both are optimal.

Strong duality theorem. If any problem of the primal-dual pair (P2) and (D2) has a feasible solution and a finite optimum then the same holds for the other problem and the two optimum values are equal.

Instead of a rigorous proof, consider:

If \mathbf{B} is an optimal basis for the primal then the optimal solution is $\mathbf{x}_B = \mathbf{B}^{-1} \mathbf{b}$ and the optimal value is

$$z = \mathbf{c}_B^T \mathbf{x}_B = \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{b} = \boldsymbol{\pi}^T \mathbf{b}.$$

The solution of the dual is $\mathbf{y}^T = \mathbf{c}_B^T \mathbf{B}^{-1} = \boldsymbol{\pi}^T$ and the optimal value is

$$\mathbf{b}^T \mathbf{y} = \mathbf{b}^T \boldsymbol{\pi}$$

which is the same as the primal.

Theorem. If any of the primal-dual pair has an unbounded (feasible!) solution then the other has no feasible solution.

The converse is not true. Namely, if one of the primal-dual pair is infeasible then the other is either unbounded or infeasible.

Example 9 Both primal and dual are infeasible.

Primal:

$$\begin{aligned} \min \quad & 2x_1 + 5x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 = 3 \\ & 2x_1 + 4x_2 = 1 \\ & x_1, x_2 \text{ free.} \end{aligned}$$

Dual:

$$\begin{aligned} \max \quad & 3y_1 + y_2 \\ \text{s.t.} \quad & y_1 + 2y_2 = 2 \\ & 2y_1 + 4y_2 = 5 \\ & y_1, y_2 \text{ free.} \end{aligned}$$

Both constraint sets are obviously contradictory.

Clark's Theorem says that unless both primal and dual are infeasible at least one of them has an unbounded feasibility set.

In general, there is a relationship between the types of primal variables and the corresponding dual constraints and, similarly, between types of primal constraints and the corresponding dual variables. This is summarized below. We assume the primal is a minimization so the dual is a maximization problem.

Primal variable	Dual constraint
$x_j = 0$	$\mathbf{y}^T \mathbf{a}_j \ll c_j$
$x_j \geq 0$	$\mathbf{y}^T \mathbf{a}_j \leq c_j$
$x_j \leq 0$	$\mathbf{y}^T \mathbf{a}_j \geq c_j$
x_j free	$\mathbf{y}^T \mathbf{a}_j = c_j$

Primal constraint	Dual variable
$\mathbf{a}^i \mathbf{x} = b_i$	y_i free
$\mathbf{a}^i \mathbf{x} \geq b_i$	$y_i \geq 0$
$\mathbf{a}^i \mathbf{x} \leq b_i$	$y_i \leq 0$
$\mathbf{a}^i \mathbf{x} \ll b_i$	$y_i = 0$

Reminder: \mathbf{a}_j is the j th column of \mathbf{A} and \mathbf{a}^i is the i th row (row vector!).

If there are bounded variables present in the primal problem the situation is a bit more complicated since a bounded variable implies an explicit primal constraint which generates an additional dual variable. Details are omitted here.

Dual feasibility

Consider the (P2)–(D2) pair. As (P2) is in CF-1 with type-2 variables only, \mathbf{A} contains a unit matrix and $m < n$. With the introduction of vector $\mathbf{w} = [w_1, \dots, w_n]^T$ of dual logical variables (D2) can be rewritten as

$$\max \quad \mathbf{b}^T \mathbf{y} \tag{48}$$

$$\text{s.t.} \quad \mathbf{A}^T \mathbf{y} + \mathbf{w} = \mathbf{c}, \tag{49}$$

$$\mathbf{w} \geq \mathbf{0}. \tag{50}$$

Let \mathbf{B} be a basis to \mathbf{A} . It need not be primal feasible. Rearranging (49), we get $\mathbf{w}^T = \mathbf{c}^T - \mathbf{y}^T \mathbf{A}$, or in partitioned form

$$\mathbf{w}_B^T = \mathbf{c}_B^T - \mathbf{y}^T \mathbf{B}, \tag{51}$$

$$\mathbf{w}_N^T = \mathbf{c}_N^T - \mathbf{y}^T \mathbf{N}. \tag{52}$$

The nonnegativity (and, in this case, the feasibility) requirement (50) of \mathbf{w} in partitioned form is $[\mathbf{w}_B^T, \mathbf{w}_N^T]^T \geq \mathbf{0}$. Choosing $\mathbf{y}^T = \mathbf{c}_B^T \mathbf{B}^{-1}$ we obtain

$$\mathbf{w}_B^T = \mathbf{c}_B^T - \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{B} = \mathbf{0}, \tag{53}$$

$$\mathbf{w}_N^T = \mathbf{c}_N^T - \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{N} = \mathbf{d}_N^T \geq \mathbf{0}, \tag{54}$$

where \mathbf{d}_N denotes the vector formed by the primal reduced costs of the nonbasic variables. Since (53) is satisfied with any basis and \mathbf{y} is unrestricted in sign, a basis \mathbf{B} is dual feasible if it satisfies (54). This is, however, nothing but the primal optimality condition.

Conclusion: the **dual feasibility conditions are the same as the primal optimality conditions**. Additionally, the dual logicals are equal to the primal reduced costs. Therefore, w_j and d_j can be used interchangeably.

This observation suggests the following idea. Assume we have a dual feasible basis to (48)–(50). If it is also primal feasible then it is optimal for both. If not then make basis changes that lead to primal feasibility while maintaining dual feasibility.

The Dual Simplex Method (DSM)

One possibility to **solve the dual** is simply to formulate the dual explicitly. Since it is an 'ordinary' LP problem it can be solved **directly by the primal simplex method**.

Disadvantages of this approach:

- If $m \ll n$ then the row size of the dual, and thus the work per iteration, may become prohibitively large (unless some other sophisticated techniques are used).
- For large scale problems this formulation is cumbersome.
- Upper bounded variables cause added complications.

There is a need for the dual simplex.

DSM effectively works on the primal problem but uses different rules to perform basis changes (pivot steps). At successful termination it provides a solution to both the primal and dual problem.

The symmetry between the primal and dual is obvious for the (P2)–(D2) pair (CF-1 with type-2 variables). The level of symmetry is higher and holds even for CF-2 with MI variables. (Reminder: primal is minimization, dual is maximization.)

The dual feasibility (primal optimality) conditions for the dual of CF-1 in case of a given basis:

Type(x_j)	Value	d_j	Remark
0	$x_j = 0$	Immaterial	
1	$x_j = 0$	≥ 0	$j \in J$
1	$x_j = u_j$	≤ 0	
2	$x_j = 0$	≥ 0	
3	$x_j = 0$	$= 0$	

(55)

It is immediately obvious that dual logical variables corresponding to type-0 primal variables are always at feasible level, therefore, they can be left out from further investigations of the dual.

Assume J of nonbasic variables at upper bound and a basis \mathcal{B} are given so that dual feasibility is satisfied. The dual solution is $\mathbf{y}^T = \mathbf{c}_B^T \mathbf{B}^{-1}$, $\mathbf{d}_B^T = \mathbf{0}^T$ and $\mathbf{d}_N^T = \mathbf{c}_N^T - \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{N}$. If the corresponding

primal solution, $\boldsymbol{\beta} = \mathbf{x}_B = \mathbf{B}^{-1} \left(\mathbf{b} - \sum_{j \in J} \mathbf{a}_j u_j \right)$, is feasible

then it is optimal. Otherwise, the current solution is primal infeasible. Thus, there exists a neighboring dual feasible basis with better (or not worse) dual objective value unless dual unboundedness is detected. This is the idea of the **dual phase-2 algorithm**.

Dual simplex for CF-1, all variables of type-2

Verbal description of DSM.

The dual algorithm starts with a dual basic feasible solution (DBFS) and maximizes the dual objective function. If the DBFS is also primal feasible, it is optimal. Otherwise, any infeasible primal basic variable can be made feasible by removing it from the basis at zero level (selection of the outgoing variable). The incoming variable is determined in such a way that dual feasibility is maintained (in the meantime, dual objective gets better or remains unchanged). This pattern is repeated until all primal variables become feasible.

Note: While the leaving infeasible basic variable becomes feasible (nonbasic at zero) other feasible basic variables may become infeasible. However, since no dual basis is repeated (assuming dual non-degeneracy) and the dual objective keeps improving the algorithm will eventually terminate.

While this verbal outline of DSM applies to any LP problem (including problems with all types of variables), the algorithmic description, for simplicity, covers only the standard case, see (45)–(47) and (48)–(50).

Consider (D2) and add dual logical variables to make

$$(D3) \quad \max \quad Z = \mathbf{b}^T \mathbf{y} \quad (56)$$

$$\text{s.t.} \quad \mathbf{A}^T \mathbf{y} + \mathbf{d} = \mathbf{c}, \quad (57)$$

$$\mathbf{d} \geq \mathbf{0}. \quad (58)$$

\mathbf{A} is assumed to contain a unit matrix in accordance with CF-1. Assume \mathbf{B} is a feasible basis to this problem. If it is not primal feasible then it is not optimal \Rightarrow exists a neighboring dual feasible basis with a better (or not worse) dual objective value unless the dual problem is unbounded.

Algorithmic description of the dual simplex method

Problem: $\min z = \mathbf{c}^T \mathbf{x}$, $\mathbf{A} \mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$.

Assumption: \mathbf{B} is a DBFS, i.e., $\mathbf{d}_N \equiv \mathbf{w}_N \geq \mathbf{0}$, $\mathbf{x}_B = \mathbf{B}^{-1} \mathbf{b}$

Step 1. Select a primal infeasible basic variable, say x_{Bp} . If none exists, \mathbf{B} is optimal, stop.

Step 2. Do **dual ratio test** with \mathbf{w}_N and $\boldsymbol{\alpha}^p$ (row p of the transformed tableau) to determine position q of the incoming variable:

$$\theta_D = \frac{w_q}{\alpha_{pq}} = \max \left\{ \frac{w_j}{\alpha_{pj}} : \alpha_{pj} < 0, j \in N \right\}.$$

If such a q does not exist, dual is unbounded \Rightarrow primal is infeasible, stop.

Step 3. **Update** solution and tableau. Assume, \mathbf{w} is row 0 of \mathbf{A} , $a_{0j} = w_j$, $\forall j$. It gets transformed the same way as any other row of \mathbf{A} . x_{B0} is the negative of the objective value.

Solution: Let $\theta_P = x_{Bp} / \alpha_{pq}$ (\equiv primal ratio.)

$$\hat{x}_{Bi} = x_{Bi} - \theta_P \alpha_{iq}, \quad i \neq p,$$

$$\hat{x}_{Bp} = \theta_P.$$

Tableau: Let $t_i = \alpha_{iq} / \alpha_{pq}$, $i \neq p$,

$$\hat{\alpha}_{ij} = \alpha_{ij} - t_i \alpha_{pj}, \quad i \neq p, \forall j,$$

$$\hat{\alpha}_{pj} = \alpha_{pj} / \alpha_{pq}.$$

Go to Step 1.

Example 10 (Dual simplex) Consider the following (primal) LP problem:

$$\begin{aligned} \min \quad & 2x_1 + x_2 + 4x_3 + 2x_4 \\ \text{s. t.} \quad & x_1 - 2x_2 + x_3 + x_4 \leq 3 \\ & -x_1 + 2x_2 - x_3 \leq -1 \\ & -x_1 + 4x_3 - 2x_4 \leq -2 \\ & x_j \geq 0, \quad j = 1, \dots, 4 \end{aligned}$$

Since all constraints are ' \leq ' the added logical variables (denoted by s_1, s_2, s_3) are of type-2. If we take the all-logical basis as a starting basis then the corresponding solution is dual feasible but primal infeasible \Rightarrow we can apply the dual simplex method.

Using the tableau form, initially we have:

B	x_1	x_2	x_3	x_4	s_1	s_2	s_3	x_B
s_1	1	-2	1	1	1			3
s_2	-1	2	-1	0		1		-1
s_3	-1	0	4	-2			1	-2
w	2	1	4	2	0	0	0	0

There are two infeasible basic variables, s_2 and s_3 . We choose s_3 to leave the basis \Rightarrow pivot row is $p = 3$. Eligible candidates for the dual ratio test are in columns 1 and 4 giving ratios of -2 and -1 , respectively. The maximum of them is -1 in column 4 $\Rightarrow x_4$ will enter the basis in position 3, pivot element is $\alpha_{34} = -2$.

The transformed tableau is obtained by applying the update formulae of Step 3 of DSM:

B	x_1	x_2	x_3	x_4	s_1	s_2	s_3	x_B
s_1	$\frac{1}{2}$	-2	3	0	1		$\frac{1}{2}$	2
s_2	-1	2	-1	0		1	0	-1
x_4	$\frac{1}{2}$	0	-2	1	0	0	$-\frac{1}{2}$	1
w	1	1	8	0	0	0	1	-2

It can be seen that the solution remained dual feasible. Now we have only one infeasible primal variable, s_2 . Choosing it to leave the basis gives $p = 2$. Eligible ratios are in columns 1 and 3 giving ratios -1 and -8 , respectively. Taking the maximum (-1), the pivot element is $\alpha_{21} = -1$, x_1 enters the basis in position 2. The transformed tableau:

B	x_1	x_2	x_3	x_4	s_1	s_2	s_3	x_B
s_1	0	-1	$\frac{5}{2}$	0	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$
x_1	1	-2	1	0		-1	0	1
x_4	0	1	$-\frac{5}{2}$	1	0	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
w	0	3	7	0	0	1	1	-3

Now, the solution is both primal and dual feasible, thus optimal.

We are interested in the primal variables:

$$x_1 = x_{B2} = 1, \quad x_2 = 0, \quad x_3 = 0, \quad x_4 = x_{B3} = \frac{1}{2},$$

$$s_1 = x_{B1} = \frac{3}{2}, \quad s_2 = s_3 = 0.$$

$$\text{The dual solution from } \mathbf{y}^T = \mathbf{c}_B^T \mathbf{B}^{-1} = [0, -1, -1].$$

Interior Point Methods (IPM) in LP

The SM is an iterative procedure that moves along the edges of the convex polyhedron of the feasible solutions until optimality (unboundedness) is detected. If the network of edges is very large, the number of iterations can also be very large. If, additionally, we have many explicit constraints (m is large) then despite the use of advanced sparsity techniques both the work per iteration and the overall computational effort can grow considerably.

A result of recent development in LP theory is the introduction of **interior point methods (IPMs)** for solving LP problems. Their main feature is that starting from a point in the interior of the convex polyhedron they move inside all the way in the direction of the optimality. Interestingly enough, they reach an ϵ optimal solution in 40–80 iterations, independent of the size of the problem. However, the computational work per iteration grows polynomially by the size. Additionally, the **solution will not be basic** and postoptimal analysis cannot be directly performed. Furthermore, detection of infeasibility or unboundedness of an LP problem is more complicated and less reliable than with the SM.

Still, the excellent performance of IPMs in case of very large scale problems makes them an important tool that complements the SM very well. In practice, we need both of them in the LP toolbox.

State-of-the-art in Linear Programming

State-of-the-art keeps changing in time. What is coming is true for LP at the turn of the millennium.

First of all, the SM and IPMs are inherently serial algorithms with relatively little scope for parallelization.

Today's serial LP solvers are still very capable. They are the result of **algorithmic developments** in LP, inclusion of **achievements of computer science and software engineering** and taking advantage of **developments in computer technology**.

Now even on PCs with 128MB of free memory and a Pentium processor ($> 1\text{GHz}$) we can solve problems in the magnitude of 100,000 rows, 200,000 columns and 1,000,000 nonzeros. Problems in this size are usually rather degenerate. Therefore, the recommended solution algorithm in such cases is an IPM (IPMs are known to be very insensitive to degeneracy).

The use of the SM is well justified for problems with up to 50,000 rows. Additionally, the SM seems to be the only suitable solution algorithm in case of integer programming problems (see later).

There are special structured LP models representing **multisector** or **multiperiod** decision **problems** of industry and finance when the size of a problem can be a million or more constraints. In such a case the application of a solution technique, called **decomposition** can be used that solves the problem through a series of smaller subproblems which can be solved independently of each other (very suitable for parallel computing).

INTEGER PROGRAMMING (IP)

We talk about **integer programming (IP)** if in an LP model some variables are restricted to take integer values. More precisely, if some variables are continuous and others are integer then we have a **mixed integer programming (MIP)** problem. The standard form of MIP:

$$\min \quad \mathbf{c}^T \mathbf{x} + \mathbf{h}^T \mathbf{y} \quad (59)$$

$$\text{s.t.} \quad \mathbf{A} \mathbf{x} + \mathbf{D} \mathbf{y} = \mathbf{b} \quad (60)$$

$$\mathbf{x}, \mathbf{y} \geq \mathbf{0} \quad (61)$$

$$\mathbf{x} \text{ integer } (\mathbf{x} \in \mathbb{Z}^n). \quad (62)$$

Very often it is reasonable to assume that the integer variables are bounded, i.e., $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$.

If all variables are integer (no y variables in the model) then it is a **pure integer programming (PIP)** problem. In the sequel, (59)–(62) will be referred to also as (MIP).

IP is a rather general modeling tool that goes far beyond what the original IP formulation may suggest.

IP has traditionally been used in situations where it is only meaningful to make integral quantities of certain goods, like aeroplanes, houses, power stations, or use integral quantities of some resources like men.

Typical application areas of MIP

1. Capital investment: Choice between alternatives with respect to budget limit and cash flow constraints.
2. Capacity expansion: Increasing production capacity in discrete steps subject to demand/supply constraints and cost considerations.
3. Planning of activities when integral number of products are to be produced using integral units of resources.
4. Production planning with logical conditions, e.g. if certain products are produced then some other products also must (must not) be produced, or there is an upper limit on the number of ingredients in a blend or on the number of tools in a process or the like.
5. Knapsack problem: A storage with given volume and/or weight capacity must be filled with goods of given volume and/or weight such that the total value of stored items is maximal. This problem can appear in disguise in many other situations.
6. Job shop scheduling, sequencing problems: Optimal ordering of operations on different machines.
7. Allocation and assignment problems: Assigning discrete resources to activities (people to jobs, vehicles to deliveries; balancing assembly lines, etc.).
8. Airline and air crew scheduling problems to assign air crews to sets of flights and/or aircraft to flights in order to meet some regulatory and other requirements.
9. Facility location problem: To decide where to locate depots (warehouses or factories) in order to supply customers.

10. Fixed charge problems: Planning activities with set-up costs. The cost of such activities has two components: a fixed cost arising if the level of the activity is positive (set-up) and a component that is proportional to the level of the activity (linear term).
11. Network problems: Finding the minimum cost flow of some commodity in a network (also transportation, transshipment, shortest path, maximum flow, minimum spanning tree).
12. Travelling salesman problem: To make a round trip visiting m cities with minimum cost/time/distance. Model can also be used for manufacturing (and even designing) VLSI chips.
13. Problems with discrete (not necessarily integer) variables.
14. Problems with disjunctive constraints: Either one or another set of constraints has to be satisfied (can be both).
15. Cutting stock problem: To cut paper/steel from pieces of wide rolls into rolls of given widths in given quantities while minimizing trimming loss.
16. And many more . . .

Solution of MIP problems

In general, MIP problems are very much harder to solve than LP problems. Even a few dozens of integer variables can cause a solver to run for days until successful termination.

Solution algorithms usually start with solving the problem without the (62) integrality requirement (**LP relaxation** of the MIP problem):

$$\min \quad \mathbf{c}^T \mathbf{x} + \mathbf{h}^T \mathbf{y} \quad (63)$$

$$\text{s.t.} \quad \mathbf{A} \mathbf{x} + \mathbf{D} \mathbf{y} = \mathbf{b} \quad (64)$$

$$\mathbf{x}, \mathbf{y} \geq \mathbf{0} \quad (65)$$

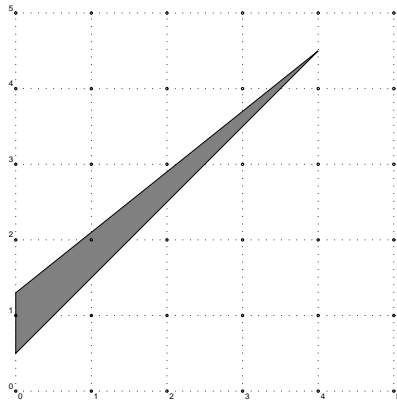
If in the optimal solution of the relaxed problem all integer variables take integer values then it is also an optimal solution to the MIP problem. Though this is the most fortunate case it is not rare at all! Certain characteristics of a problem can guarantee that this will happen. Knowing them, we can try to formulate the model in such a way that it has the necessary features.

If not all integer variables have integer values in the relaxed problem we can round the solution values to the nearest integers. This can be acceptable in cases when the solution values are large enough (say, greater than 8). In other cases rounding may result in unacceptable infeasible solutions as in the following example.

Example 11 (from H.P. Williams):

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.t.} \quad & -2x_1 + 2x_2 \geq 1 \\ & -8x_1 + 10x_2 \leq 13 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned}$$

The optimal solution of the relaxed problem is $x_1 = 4$ and $x_2 = 4.5$. If we round x_2 to either of the nearest integers (4 or 5) the solution becomes infeasible. The true integer optimal solution is $x_1 = 1$ and $x_2 = 2$ which is quite far away from the continuous optimum and it is not trivial how it can be obtained from the relaxed solution. The diagram of the problem demonstrates the situation.



For MIP there is no general purpose solution algorithm like the simplex method for LP. We have to use one of the several MIP algorithms. The most successful ones are based on the **Branch and Bound (B&B)** principle. B&B is a systematic way of testing (explicitly or implicitly) all possible integer combinations and identifying a/the best one (tree search). Knowing the model can help substantially in directing the search so that infeasible and non-optimal (hopefully large) branches of the tree are easily identified and ruled out.

Many practical IP models restrict the integer variables to just two values, 0 or 1. Such variables are often referred to as **binary variables** and the problems involving them **0-1** (or **0/1**) **programming** problems. Binary variables usually represent 'yes' (1) or 'no' (0) decisions. They also can be used to represent logical relationships among variables and constraints. As such, they are indispensable modeling tools.

The LP relaxation of a 0/1 MIP problem is

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{x} + \mathbf{h}^T \mathbf{y} \\ \text{s.t.} \quad & \mathbf{A} \mathbf{x} + \mathbf{D} \mathbf{y} = \mathbf{b} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}, \quad \mathbf{y} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{1}$ is a vector with unit components, $\mathbf{1} = [1, \dots, 1]^T$.

In practice, the most important MIP problems are the ones with 0/1 variables. Additionally, nearly any sensible MIP problem can be converted into 0/1.

Outline of Branch and Bound methods

Assume, we are trying to solve (MIP) with all the integer variables \mathbf{x} having finite upper bounds. This problem is called P_0 . First, we solve the LP relaxation, $LP(P_0)$, of it. Note, since $LP(P_0)$ is less constrained than P_0 the optimum of $LP(P_0)$ is better than or equal to that of P_0 .

If in the optimal solution of $LP(P_0)$ one or more components of \mathbf{x} are non-integer, one such variable, say x_j is chosen and used to split the problem into two subproblems, P_1 and P_2 . Assume, x_j takes a non-integer value of $\beta_j > 0$ in the optimum of $LP(P_0)$. P_1 and P_2 are defined as follows:

$$P_1 := P_0 \text{ and } x_j \leq \lfloor \beta_j \rfloor \quad (66)$$

$$P_2 := P_0 \text{ and } x_j \geq \lfloor \beta_j \rfloor + 1. \quad (67)$$

Now, any solution to P_0 is either a solution of P_1 or P_2 meaning that P_0 can be solved by solving P_1 and P_2 .

Constraints added to P_0 to create P_1 and P_2 are individual bounds on x_j that can be handled algorithmically rather than as explicit constraints (see LP with all types of variables).

Note, if x_j is a binary variable (66) and (67) reduce to adding constraints $x_j = 0$ and $x_j = 1$, respectively, i.e., the variable becomes fixed at one of its feasible values.

We carry on and solve $LP(P_1)$ or $LP(P_2)$. Assume P_1 was chosen. The solution of $LP(P_1)$ is evaluated in the same way as that of $LP(P_0)$. If \mathbf{x} is still not all integral, we split P_1 and create P_3 and P_4 in a similar spirit. And so on . . .

This process can be viewed as the construction of a **binary tree** of subproblems whose terminal nodes, called **waiting nodes**, correspond to problems still to be solved and evaluated.

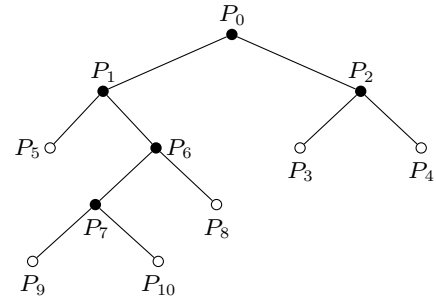
At a certain stage we encounter an all-integer solution. It may or may not be an optimal solution to P_0 . If there are still waiting nodes all of them have to be examined as they may lead to a better integer solution. However, knowing an integer solution (with optimal value Z) can dramatically reduce the generation of new nodes. Namely,

if for the LP optimum z of a node $z > Z$ holds then even the relaxed problems in any successor of this node will have the same or worse optimum value, therefore, they all can be excluded from further investigations.

For the formal description of the B&B algorithm in the context of MIP, let Z denote the objective value of the best intermediate integer solution found until a certain stage of the solution process and W the set of problems in the waiting nodes.

Branch and Bound Algorithm for MIP

- Step 0: Solve $LP(P_0)$. If \mathbf{x} is all-integral, it is an optimal solution to P_0 , terminate.
 Otherwise, initialize $Z = +\infty$. Set $k = 0$, put P_0 in the root of the tree as a waiting node, set $W := \{P_0\}$ and denote the optimal value of $LP(P_0)$ by z_0 .
- Step 1: Choose a problem $P_t \in W$ such that $z_t < Z$ if one exists, otherwise terminate. The best integer solution found so far is optimal. If none has been found the problem has no feasible integer solution (integer infeasibility).
- Step 2: Choose an integer variable x_j with a fractional value β_j . Create two new LP problems P_{k+1} and P_{k+2} (successor nodes) from the current problem, one with the added extra constraint $x_j \leq \lfloor \beta_j \rfloor$ and the other with $x_j \geq \lfloor \beta_j \rfloor + 1$.
- Step 3: Solve the LP relaxations $LP(P_{k+1})$ and $LP(P_{k+2})$. If a new and improved integer solution is found, store it and update Z . If either z_{k+1} or z_{k+2} is greater than Z the corresponding problem can be abandoned together with all its possible successors (see remark earlier). Similar action is made if any of the problems turns out to be infeasible. Add P_{k+1} and/or P_{k+2} to W if the corresponding problem is feasible and the LP optimum (z_{k+1} and/or z_{k+2}) is better than Z .
 Increment k by the number of nodes (0, 1, or 2) added to W and go to Step 1.



Note (again), the newly created subproblems are more restricted than any of their predecessors, therefore, the LP relaxations of these problems have worse (not better) optimum values.

Since the integer variables are all bounded the algorithm must terminate eventually because as one proceeds further down the tree the bounds on the variables become ever tighter and they finally become exact if the LP solutions were never integer before.

Important to note that the newly created subproblems need not be solved from scratch but starting from the optimal solution of the predecessor problem. To this end, DSM can be used with great success because dual feasibility of the solution is not effected by changing the bounds of a basic variable. Therefore, the optimal basis of the parent node is a dual feasible basis for its child nodes. In such cases DSM usually requires very few iterations to obtain an optimal solution to the modified problem (process called **reoptimization**).

Some additional notes on B&B

There is a certain flexibility in B&B. At some points a number of choices are available. In this way the algorithm can be tuned. The actual selection of an alternative makes a huge difference in the performance of B&B. There is no known selection strategy that is the best for all problems. The good choice is quite problem and situation dependent. The latter means that a good strategy of choosing one possibility can change even during solution.

Choosing a waiting node. Some of the successful strategies: (i) select node with best objective value, (ii) do a breadth-first type search of the tree, or (iii) a depth-first search.

Choosing a branching variable. This gives rise to even more variations, like fractional variable (i) with best objective coefficient, (ii) that is about half way between two integer values, (iii) that is recommended by some utility/penalty function, (iv) that is an 'important' variable in the model (e.g., 0/1 decision). Usually the latter is the most efficient and it assumes the knowledge of the model. Therefore, the owner of the model has the best chance to solve it efficiently or at all. Priority ordering of variables is useful to assist the choice of the branching variable.

Feasible set of MIP solutions

In any feasible MIP solution integer variables can take only integer values. Simplex method provides a vertex solution.

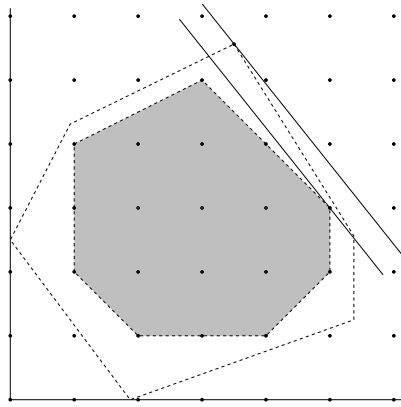
Ambitious goal: Add constraints to ensure that no integer feasible point is cut off and in the optimal LP vertex integrality is satisfied. It is usually very difficult to achieve.

Less ambitious goal: (Re)formulate the MIP problem such that it has a **tight** LP relaxation.

The importance of tightness can be highlighted as follows. For easier demonstration, we consider pure IP problems with all a_{ij} and b_i coefficients integer and finite bounds on integer variables. In this way the set of all feasible integer solutions is finite, $F = \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$. The **convex hull** of F is defined as the convex linear combination of all \mathbf{x} points:

$$H_c(F) = \left\{ \sum_{j=1}^k \lambda_j \mathbf{x}_j : \sum_{j=1}^k \lambda_j = 1, \lambda_j \geq 0, \mathbf{x}_j \in F, \forall j \right\}.$$

Obviously, $F \subseteq H_c(F)$ and $H_c(F)$ has integer vertices. ($H_c(F)$ is the smallest convex set that contains F .) Furthermore, the feasible set P of any LP relaxation contains $H_c(F)$, i.e., $H_c(F) \subseteq P$. Therefore, we have $F \subseteq H_c(F) \subseteq P$.



If we knew $H_c(F)$ explicitly like $H_c(F) = \{x : Gx \leq g\}$, we could solve the IP problem

$$\min\{c^T x : x \in F\}$$

by finding a vertex solution to the LP problem

$$\min\{c^T x : x \in H_c(F)\}. \tag{68}$$

Using the simplex method to solve (68) we can easily achieve it.

Solution algorithms like B&B increasingly tighten the problem so that no integer solution is left out of the tighter set of constraints.

The **quality of a formulation** of an IP problem with a feasible set F can be judged by the closeness of the feasible set of its LP relaxation to $H_c(F)$, the convex hull of F . One formulation of the same IP problem is said to be stronger than another if its feasible set is contained in that of the other.

While, in general, there is no easy way of determining $H_c(F)$, there is a property of an IP problem that ensures all vertex solutions are integer. This property is called **total unimodularity** of the matrix of constraints.

Formally, a matrix A is totally unimodular if the determinant of every square sub-matrix of A is equal to 0 or ± 1 .

Checking unimodularity of a matrix is not easy. There is, however, a property (called property K) which is easier to verify and which guarantees total unimodularity (sufficient condition), and thus an easy way of solving the problem by the simplex method.

Property K (after H.P. Williams)

1. Each element of A is 0 or ± 1 .
2. No more than two nonzeros appear in each column.
3. The rows can be partitioned into two subsets K_1 and K_2 such that (i) if a column contains two nonzeros of the same sign, one element is in each of the subsets; (ii) if a column contains two nonzeros of the opposite sign, both elements are in the same subset.

Note, either set can be empty.

There are several types of IP problem that satisfy property K . They include transportation, transshipment, maximum flow, assignment and shortest path problems. For them it is sufficient to solve the LP relaxation and the solution is guaranteed to be integer, thus a solution to the IP problem.

The Knapsack Problem

It is a PIP problem of special importance:

$$\begin{aligned} \max \quad & c_1x_1 + c_2x_2 + \dots + c_nx_n \\ \text{subject to} \quad & a_1x_1 + a_2x_2 + \dots + a_nx_n \leq b \\ & x_j, j = 1, \dots, n \text{ nonnegative integers.} \end{aligned}$$

Note: only one constraint! In general, x_j variables have finite (usually small) upper bound. In most cases x_j s are 0/1 variables.

The knapsack model is widely used either in its own right (*project selection, capital budget allocation, warehouse stocking etc.*) or, much more frequently, as a subproblem of other (M)IP problems like the *cutting stock* problem.

Knapsack Problem: Examples

Race car upgrade

Problem

XY racing team wants to improve top speed of its cars. Choice can be made from 6 different features that can be added. Estimated costs and speed enhancements are as follows:

	Features					
	1	2	3	4	5	6
Cost (1000 £'s)	10.2	6.0	23.0	11.1	9.8	31.6
Speed gain (kmh)	8	3	15	7	10	12

Team has a **budget** of £35,000 that cannot be exceeded. Within this limit they want to achieve the **maximum speed improvement**.

Model

Decision variables:

$$x_j = \begin{cases} 1, & \text{if feature } j \text{ is added;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \max \quad & 8x_1 + 3x_2 + 15x_3 + 7x_4 + 10x_5 + 12x_6 \\ \text{s.t.} \quad & 10.2x_1 + 6x_2 + 23x_3 + 11.1x_4 + 9.8x_5 + 31.6x_6 \leq 35 \\ & x_j \in \{0, 1\}, \quad j = 1, \dots, 6. \end{aligned}$$

An optimal solution is $x_1 = x_4 = x_5 = 1, x_2 = x_3 = x_6 = 0$.

Total gain is **25 kmh** at cost of **£31,100**.

Variation of the upgrade

Speed must be **increased** by at least 30 kmh at **minimum cost**.

Model

Decision variables: as above.

$$\begin{aligned} \min \quad & 10.2x_1 + 6x_2 + 23x_3 + 11.1x_4 + 9.8x_5 + 31.6x_6 \\ \text{s.t.} \quad & 8x_1 + 3x_2 + 15x_3 + 7x_4 + 10x_5 + 12x_6 \geq 30 \\ & x_j \in \{0, 1\}, \quad j = 1, \dots, 6. \end{aligned}$$

An optimal solution is $x_1 = x_3 = x_5 = 1, x_2 = x_4 = x_6 = 0$.

Total cost is **£43,000** while speed gain is **33 kmh**.

Set Covering

Given: **set of objects**. For simplicity, they are **referred to by numbers**:

$$S = \{1, 2, \dots, m\}$$

Also given: set of subsets of S denoted by $\mathbb{S}, |\mathbb{S}| = n$. A **cost** is assigned to each of these **subsets**.

Example 12

$$S = \{1, 2, 3, 4, 5\}, \tag{69}$$

$\mathbb{S} = \{\{1\}, \{1, 2\}, \{1, 3, 5\}, \{2, 4, 5\}, \{3\}, \{4, 5\}\}$, and the corresponding costs $\{1, 2, 3, 3, 1, 2\}$. Note: in this example $m = 5$ and $n = 6$.

The **Set Covering Problem** is to select a subset of \mathbb{S} that **covers all elements** of S by members of \mathbb{S} at **minimum cost**.

Example 13 A possible cover for S of (69) is $\{1, 3, 5\}, \{1, 2\}, \{2, 4, 5\}$ with cost $3 + 2 + 3 = 8$.

Mathematical model:

Introduce indicator variables $\delta_j, j = 1, 2, \dots, 6$

$$\delta_j = \begin{cases} 1 & \text{if the } j\text{th member of } \mathbb{S} \text{ is in the cover,} \\ 0 & \text{otherwise.} \end{cases}$$

Introduce constraints that ensure that each member of S is covered (appears in at least one of the chosen subsets). For instance, in order to cover element $\{1\}$ of S we must choose at least one of the subsets that contains $\{1\}$, i.e., $\{1\}, \{1, 2\}, \{1, 3, 5\}$. It can be expressed by $\delta_1 + \delta_2 + \delta_3 \geq 1$.

The other constraints are set up in a similar way resulting in

$$\begin{aligned} \text{minimize} \quad & 1\delta_1 + 2\delta_2 + 3\delta_3 + 3\delta_4 + 1\delta_5 + 2\delta_6 \\ \text{subject to} \quad & \delta_1 + \delta_2 + \delta_3 \geq 1 \\ & \delta_2 + \delta_4 \geq 1 \\ & \delta_3 + \delta_5 \geq 1 \\ & \delta_4 + \delta_6 \geq 1 \\ & \delta_3 + \delta_4 + \delta_6 \geq 1 \end{aligned}$$

$$\forall j : \delta_j \in \{0, 1\}$$

Using matrix notation:

$$\begin{aligned} \min \quad & \mathbf{c}^T \boldsymbol{\delta} \\ \text{s.t.} \quad & \mathbf{A} \boldsymbol{\delta} \geq \mathbf{1}, \\ & \delta_j \in \{0, 1\}, \quad j = 1, \dots, n, \end{aligned}$$

where $a_{ij} = \begin{cases} 1 & \text{if member } j \text{ of } \mathbb{S} \text{ covers object } i \text{ of } S, \\ 0 & \text{otherwise.} \end{cases}$

If the costs are all 1, then we require a cover that minimizes the number of members of \mathbb{S} used to cover S .

Typical application areas: Location of service centers (fire, ambulance, etc.) to cover regions.

Example 14 Minimum set of keys

There are 5 keys and 6 locks. Every key opens one or more locks as shown in the following table:

	Key1	Key2	Key3	Key4	Key5
Lock1	x	x	x		
Lock2			x	x	
Lock3	x				x
Lock4	x	x			x
Lock5		x		x	
Lock6		x		x	x

Write an optimization model that chooses the minimum number of keys such that any of the locks can be opened.

The situation can be modeled as a set covering problem.

Decision (indicator) variables: $\delta_j, j = 1, 2, \dots, 5$.

$$\delta_j = \begin{cases} 1, & \text{if key } j \text{ is in the selection,} \\ 0, & \text{otherwise.} \end{cases}$$

The model

$$\begin{aligned}
& \text{minimize} && \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 \\
& \text{subject to} && \delta_1 + \delta_2 + \delta_3 \geq 1 \\
& && \delta_3 + \delta_4 \geq 1 \\
& && \delta_1 + \delta_2 + \delta_5 \geq 1 \\
& && \delta_2 + \delta_4 \geq 1 \\
& && \delta_2 + \delta_4 + \delta_5 \geq 1
\end{aligned}$$

and $\delta_j \in \{0, 1\}$, $j = 1, \dots, 5$.

Set Packing

Given: **set of objects**. For simplicity, they are referred to by **numbers**:

$$\mathcal{S} = \{1, 2, \dots, m\}$$

Also given: set of subsets of \mathcal{S} denoted by \mathbb{S} , $|\mathbb{S}| = n$. A **value** is assigned to each of these **subsets**.

Example 15

$$\mathcal{S} = \{1, 2, 3, 4, 5, 6\}, \quad (70)$$

$\mathbb{S} = \{\{1, 2, 5\}, \{1, 3\}, \{2, 4\}, \{3, 6\}, \{2, 3, 6\}\}$, and the corresponding values $\{3, 2, 2, 2, 3\}$. Note: in this example $m = 6$ and $n = 5$.

The **Set Packing Problem** is to 'pack' as many members of \mathbb{S} into \mathcal{S} as possible to maximize total value such that there is **no overlap**.

The latter means at most one element of \mathcal{S} can appear in the chosen subsets.

Example 16 A possible packing for \mathcal{S} of (70) is $\{1, 2, 5\}$, $\{3, 6\}$ with value $3 + 2 = 5$.

Mathematical model:

Introduce indicator variables δ_j , $j = 1, 2, \dots, 5$

$$\delta_j = \begin{cases} 1 & \text{if the } j\text{th member of } \mathbb{S} \text{ is in the pack,} \\ 0 & \text{otherwise.} \end{cases}$$

Constraints are set up to ensure that no element of \mathcal{S} is included in more than one member of \mathbb{S} in the pack (to avoid overlap). For instance, to guarantee that element $\{1\}$ of \mathcal{S} is included at most once, we can write $\delta_1 + \delta_2 \leq 1$. If we want to **maximize the number of members** of \mathbb{S} in the pack the following model is obtained:

$$\begin{aligned}
& \text{maximize} && \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 \\
& \text{subject to} && \delta_1 + \delta_2 \leq 1 \\
& && \delta_1 + \delta_3 + \delta_5 \leq 1 \\
& && \delta_2 + \delta_4 + \delta_5 \leq 1 \\
& && \delta_3 \leq 1 \\
& && \delta_1 \leq 1 \\
& && \delta_4 + \delta_5 \leq 1
\end{aligned}$$

$$\forall j : \delta_j \in \{0, 1\}$$

Using matrix notation, the **set packing problem** is:

$$\begin{aligned}
& \max && \mathbf{1}^T \boldsymbol{\delta} \\
& \text{s.t.} && \mathbf{A} \boldsymbol{\delta} \leq \mathbf{1}, \\
& && \delta_j \in \{0, 1\}, j = 1, \dots, n,
\end{aligned}$$

where $a_{ij} = \begin{cases} 1 & \text{if member } j \text{ of } \mathbb{S} \text{ includes object } i \text{ of } \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$

Objective coefficients can be different from 1. In that case the total value of the pack is maximized.

Set Partitioning

Again, given a **set of objects**. They are referred to by **numbers**:

$$\mathcal{S} = \{1, 2, \dots, m\}$$

Also given a set of subsets of \mathcal{S} denoted by \mathbb{S} , $|\mathbb{S}| = n$. A **value** is assigned to each of these **subsets**.

Example 17 The same as for set covering.

$$\mathcal{S} = \{1, 2, 3, 4, 5\}, \quad (71)$$

$\mathbb{S} = \{\{1\}, \{1, 2\}, \{1, 3, 5\}, \{2, 4, 5\}, \{3\}, \{4, 5\}\}$, and the corresponding costs $\{1, 2, 3, 3, 1, 2\}$. Note: in this example $m = 5$ and $n = 6$.

The **Set Partitioning Problem** is to **cover** all elements of \mathcal{S} using members of \mathbb{S} **with no overlap** (i.e., cover and pack) while minimizing/maximizing an objective.

Example 18 A possible partitioning for \mathcal{S} of (71) is $\{1, 2\}$, $\{3\}$, and $\{4, 5\}$ with value $2 + 1 + 2 = 5$.

Mathematical model:

Introduce indicator variables δ_j , $j = 1, 2, \dots, 6$

$$\delta_j = \begin{cases} 1 & \text{if the } j\text{th member of } \mathbb{S} \text{ is included,} \\ 0 & \text{otherwise.} \end{cases}$$

Constraints are the same as for set covering except \geq is replaced by $=$. Now we are interested in the minimum number of subsets needed for partitioning.

$$\begin{aligned} \text{minimize} \quad & \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 + \delta_6 \\ \text{subject to} \quad & \delta_1 + \delta_2 + \delta_3 = 1 \\ & \delta_2 + \delta_4 = 1 \\ & \delta_3 + \delta_5 = 1 \\ & \delta_4 + \delta_6 = 1 \\ & \delta_3 + \delta_4 + \delta_6 = 1 \end{aligned}$$

$$\forall j : \delta_j \in \{0, 1\}$$

Using matrix notation, the **set partitioning problem** is:

$$\begin{aligned} \text{min or max} \quad & \mathbf{1}^T \boldsymbol{\delta} \\ \text{s.t.} \quad & \mathbf{A} \boldsymbol{\delta} = \mathbf{1}, \\ & \delta_j \in \{0, 1\}, j = 1, \dots, n, \end{aligned}$$

where $a_{ij} = \begin{cases} 1 & \text{if member } j \text{ of } \mathbb{S} \text{ includes object } i \text{ of } \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$

Objective coefficients can be different from 1. In that case the total value of the partitioning is minimized or maximized.

Application areas include airline crew scheduling, rosters of all kinds, political districting (defining constituencies).

Comparison of Set Covering, Packing and Partitioning

Set Covering

$$\begin{aligned} \text{min} \quad & \mathbf{1}^T \boldsymbol{\delta} \\ \text{s.t.} \quad & \mathbf{A} \boldsymbol{\delta} \geq \mathbf{1}, \\ & \delta_j \in \{0, 1\}, j = 1, \dots, n, \end{aligned}$$

Set Packing

$$\begin{aligned} \text{max} \quad & \mathbf{1}^T \boldsymbol{\delta} \\ \text{s.t.} \quad & \mathbf{A} \boldsymbol{\delta} \leq \mathbf{1}, \\ & \delta_j \in \{0, 1\}, j = 1, \dots, n, \end{aligned}$$

Set Partitioning

$$\begin{aligned} \text{min or max} \quad & \mathbf{1}^T \boldsymbol{\delta} \\ \text{s.t.} \quad & \mathbf{A} \boldsymbol{\delta} = \mathbf{1}, \\ & \delta_j \in \{0, 1\}, j = 1, \dots, n, \end{aligned}$$

In all cases \mathbf{A} is a 0-1 matrix.

Plant Location

Problem: Company wants to build plants to supply customers. There are m customers and n potential locations for plants.

Problem data:

- n potential locations for plants
- m number of customers
- c_{ij} cost of supplying one unit of demand i from plant j
- f_j fixed cost of opening (building) a plant in location j
- d_i demand of customer i
- s_j supply available at plant j (if open)

Decision variables:

- x_{ij} units of product delivered from plant j to customer i
- δ_j binary variable: = 1 if plant j is to be built, 0 otherwise.

This model results in a mixed integer LP problem that is often called **capacitated plant location** problem.

$$\begin{aligned} \text{min} \quad & \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + \sum_{j=1}^n f_j \delta_j \\ \text{s.t.} \quad & \sum_{j=1}^n x_{ij} = d_i, i = 1, \dots, m \\ & \sum_{i=1}^m x_{ij} - s_j \delta_j \leq 0, j = 1, \dots, n \\ & x_{ij} \geq 0, i = 1, \dots, m, j = 1, \dots, n \\ & \delta_j \in \{0, 1\}, j = 1, \dots, n. \end{aligned}$$

Remark: impossible deliveries can be excluded by assigning high c_{ij} unit cost.

Importance of (M)IP formulation

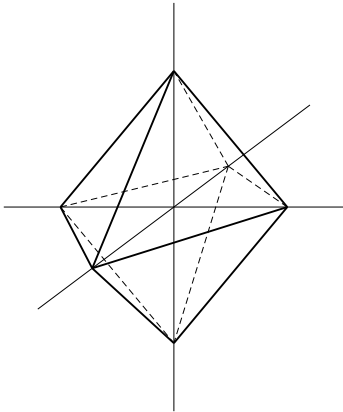
The chances of success in solving (M)IP problems heavily depend on the problem formulation.

For example, the feasible set defined by

$$|x_1| + |x_2| + \dots + |x_n| \leq 1$$

is a generalized octahedron.

For $n = 3$ it is the traditional octahedron:



Equivalent LP formulations:

Formulation 1:

$$\begin{aligned} x_1 + x_2 + \dots + x_n &\leq 1 \\ -x_1 + x_2 + \dots + x_n &\leq 1 \\ &\dots \\ -x_1 - x_2 - \dots - x_n &\leq 1 \end{aligned}$$

i.e., all possible combinations of ± 1 coefficients, altogether 2^n dense constraints with n variables.

Formulation 2:

$$\begin{aligned} x_1 &= y_1^+ - y_1^- \\ &\dots \\ x_n &= y_n^+ - y_n^- \\ y_1^+ + y_1^- + \dots + y_n^+ + y_n^- &\leq 1 \\ y_j^+, y_j^- &\geq 0, \quad j = 1, \dots, n, \end{aligned}$$

i.e., $2n$ nonnegative variables and $n + 1$ constraints.

Contrast between LP and MIP: LP is sensitive to the # of constraints m while the # of variables is less important. MIP is very sensitive to the # of integer variables and can benefit from a larger number of constraints (that make the formulation tighter).

NETWORK OPTIMIZATION

Networks: Popular and successful modelling tools.

- Many real life decision problems can be modelled by networks.
- Networks can be visualized, results well understood.
- Efficient solution algorithms exist for a number of network type problems.

Physical flow :

- Electricity (generation and distribution)
- Information (communication)
- Goods, vehicles, passengers
- Money
- ...

Logical flow :

- Assignment
- Determining path lengths
- ...

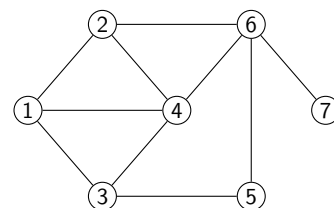
Most of the network problems are 'just' LP problems \Rightarrow LP solution technology can be used (simplex or interior point methods). **But:** special network algorithms are 50-200 times more efficient and also have some additional benefits.

Sources of efficiency

1. Algorithmic advancements exploiting the special structure of the problems.
2. Use of achievements of computer science:
 - advanced data structures,
 - efficient search, sort and merge algorithms,
 - tree and list processing.

NETWORKS are best represented by GRAPHS

Undirected graph: set of nodes and edges, $G = (N, E)$, where N is the set of nodes, E is the set of edges. (i, j) is an undirected edge connecting distinct nodes i and j . (i, i) self-loop is not allowed.

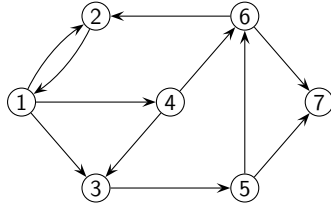


Path: a sequence of distinct nodes i_1, \dots, i_q such that edge $(i_k, i_{k+1}) \in E$ for $k = 1, \dots, q - 1$.

Cycle: a path with $i_1 = i_q$.

G is **connected** if for every distinct $i, j \in N$ there exists a path from i to j .

Directed graph: set of nodes and directed arcs, $G = (N, A)$, where N is the set of nodes, A is the set of directed arcs. (i, j) is an ordered pair denoting an arc from i to j ; outgoing from node i , incoming to node j .
 $O(i)$ = set of end nodes of outgoing arcs from node i ,
 $O(i) = \{j \in N : (i, j) \in A\}$.
 $I(i)$ = set of starting nodes of incoming arcs to node i ,
 $I(i) = \{j \in N : (j, i) \in A\}$.



Path: a sequence i_1, \dots, i_q of distinct nodes and the associated sequence a_1, \dots, a_{q-1} of arcs.

a_k is a $\begin{cases} \text{forward arc if } a_k = (i_k, i_{k+1}), \\ \text{backward arc if } a_k = (i_{k+1}, i_k). \end{cases}$

E.g., 1, (1, 3), 3, (4, 3), 4, (4, 6), 6 is a path with one backward arc (4, 3) in it.

Directed path: a path containing only forward arcs.

Cycle: a path with $i_1 = i_q$.

A directed graph is **connected** if the graph obtained by ignoring the direction of the arcs is connected.

The general network problem

Now: **network is a directed graph:** $G = (N, A)$.

$n = |N|$ # of nodes

$m = |A|$ # of arcs

b_i signed external supply at node i , $i \in N$.

For nodes : $i \in N$

$b_i > 0$: supply node (source),

$b_i < 0$: demand node (sink),

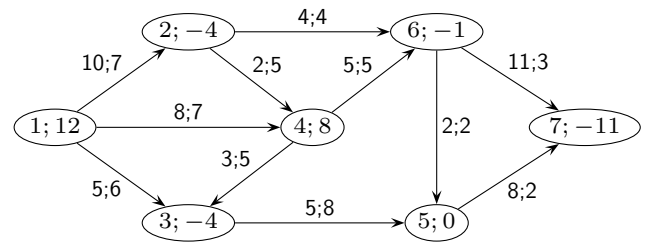
$b_i = 0$: transshipment node.

For arcs : $(i, j) \in A$

x_{ij} : flow from node i to node j (arc variable),

u_{ij} : capacity of arc (i, j) ,

c_{ij} : cost of moving one unit along arc (i, j)



Node labels $(i; b_i)$
 Arc labels $(u_{ij}; c_{ij})$

Formal statement of the problem

Constraints :

Conservation of flow

$$b_i + \sum_{j \in I(i)} x_{ji} = \sum_{j \in O(i)} x_{ij}, \quad \forall i \in N$$

Bounds on flows

$$0 \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A$$

Objective :

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

Minimal cost network flow (MCNF) problem

$$\begin{aligned} & \min \sum_{(i,j) \in A} c_{ij} x_{ij} \\ & \text{subject to} \\ & \sum_{j \in O(i)} x_{ij} - \sum_{j \in I(i)} x_{ji} = b_i, \quad \forall i \in N \\ & 0 \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A \end{aligned}$$

This is an LP problem of the form:

$$\begin{aligned} \min & \quad \mathbf{c}^T \mathbf{x} \\ \text{s.t.} & \quad \mathbf{A} \mathbf{x} = \mathbf{b} \\ & \quad \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{aligned}$$

Here, \mathbf{A} is an **arc-node incidence matrix**

Column representing arc (i, j) :

+1 in position i (from)
 -1 in position j (to)
 0 elsewhere

Necessary condition for the feasibility of the problem: $\sum_{i \in N} b_i = 0$.

MCNF: matrix form of the example

		A r c s									
Nodes	(1, 2)	(1, 3)	(1, 4)	(2, 4)	(2, 6)	(3, 5)	(4, 3)	(4, 6)	(5, 7)	(6, 7)	b_i
1	1	1	1	1	1	1					12
2	-1										-4
3		-1				1	-1				-4
4			-1	-1			1	1			8
5					-1	-1		-1	1	-1	0
6								-1	1	1	-1
7									-1	-1	-11
u_{ij}	10	5	8	2	4	5	3	5	8	11	
c_{ij}	7	6	7	5	4	8	5	5	2	3	

Integrity property: If all b_i and u_{ij} are integer then every basic feasible solution (including optimal BFS) is integer valued (because **A** satisfies property K).

Additional properties of MCNF

- If all data (including c_{ij}) are integer then both primal and dual BFSs are integer.
- Every basis **B** to **A** is triangular \Rightarrow no need for explicit \mathbf{B}^{-1} .
- The subgraph corresponding to **B** is a tree T .
- Simplex operations involving \mathbf{B}^{-1} are performed using the tree representation.
- Most of the (or the entire) algorithm can work with integer arithmetic.

Transportation problem

Transporting a single commodity.

- m # of supply nodes
- n # of demand nodes
- s_i supply at node i , $i = 1, \dots, m$
- d_j demand at node j , $j = 1, \dots, n$
- c_{ij} cost of transporting one unit from supply node i to demand node j ,
- x_{ij} (unknown) quantity transported from node i to node j .

Assumption:

Total supply is equal to total demand: $\sum_{i=1}^m s_i = \sum_{j=1}^n d_j$.

Objective:

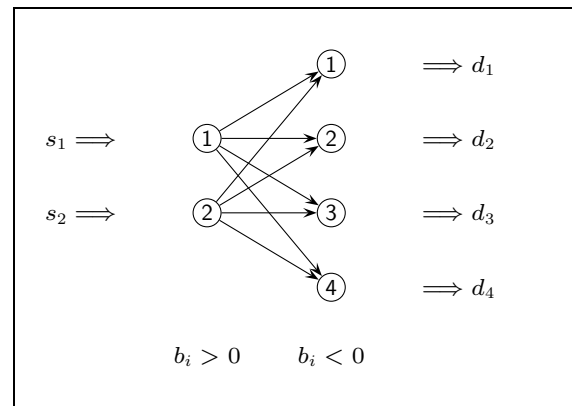
To satisfy every demand with the least possible total cost.

$$\begin{aligned} & \min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\ & \text{subject to} \sum_{i=1}^m x_{ij} = d_j, \quad j = 1, \dots, n \\ & \sum_{j=1}^n x_{ij} = s_i, \quad i = 1, \dots, m \\ & \sum_{i=1}^m s_i = \sum_{j=1}^n d_j \\ & x_{ij} \geq 0, \quad \forall(i, j) \end{aligned}$$

Capacitated transportation problem: all the above plus bounds on arc variables:

$$0 \leq x_{ij} \leq u_{ij}, \quad \text{for some or all } (i, j)$$

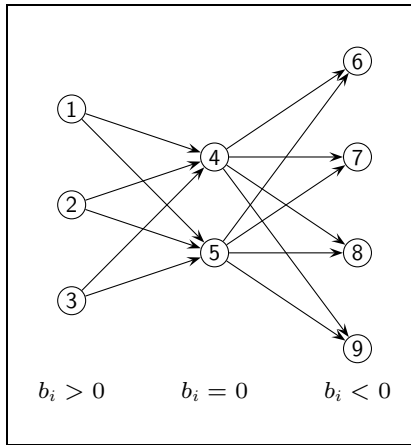
The equivalent MCNF form (example):



Bipartite graph: Nodes are divided into two groups (supply and demand nodes, in this case); there is no connection between nodes within a group.

Transshipment problem

It is a generalization of the transportation problem. It is assumed that a given commodity is produced at different sources. It is then delivered to intermediate storages which supply the customers.



The diagram clearly shows how this transshipment problem can be formulated as an MCNF problem with 9 nodes and 14 arcs.

Arc capacities can also be imposed leading to the **capacitated transshipment problem**.

Assignment problem

m # of assignees (workers, vehicles, machines)

m # of tasks

Assumptions:

Each assignee must be assigned to exactly one task.
Each task is to be performed by one assignee.

c_{ij} cost associated with assignee i performing task j .

Objective:

Find an optimal (min cost) assignment.

$$x_{ij} = \begin{cases} 1, & \text{if assignee } i \Rightarrow \text{task } j \\ 0, & \text{otherwise.} \end{cases}$$

$$\min \sum_{i=1}^m \sum_{j=1}^m c_{ij}x_{ij}$$

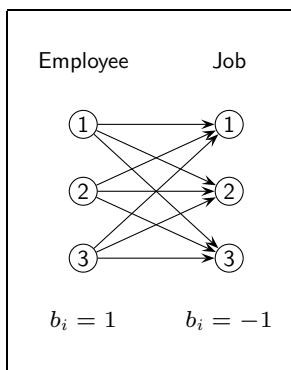
subject to

$$\sum_{i=1}^m x_{ij} = 1, \quad j = 1, \dots, m$$

$$\sum_{j=1}^m x_{ij} = 1, \quad i = 1, \dots, m$$

$$x_{ij} \in \{0, 1\}, \quad \forall(i, j)$$

The assignment problem is a **special case** of the transportation problem, and thus the MCNF problem, with the following interpretation.



Maximum flow problem

Given a directed graph $G = (N, A)$ with two special nodes

s : source node,

t : sink node.

Flow on arc (i, j) is bounded: $0 \leq x_{ij} \leq u_{ij}$.

N.B. Some (but not all) u_{ij} can be $+\infty$.

Find the **maximum possible flow** from source to sink, subject to the usual constraints (conservation of flow, arc capacities, nonnegativity of the flow) with zero external supply at all nodes different from s and t .

$$\max b_s$$

subject to

$$\mathbf{Ax} = \mathbf{b}$$

$$b_t = -b_s$$

$$b_i = 0, \quad i \neq s, t$$

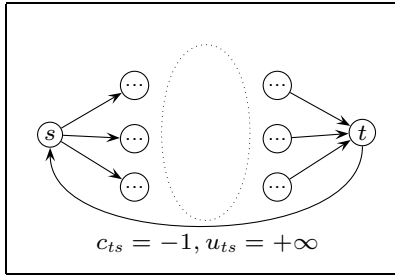
$$0 \leq x_{ij} \leq u_{ij}, \quad \forall(i, j) \in A.$$

Note: b_s is a variable now (so is b_t).

Reformulation of maximum flow as a minimum cost network flow problem:

Let $c_{ij} = 0$ for all (i, j) .
 Connect sink and source with an arc (t, s) of infinite capacity and a negative (say, -1) cost.
 Let $\bar{A} = A \cup \{(t, s)\}$ and let x_{ts} denote the corresponding arc variable.

If we minimize $\sum_{(i,j) \in \bar{A}} c_{ij}x_{ij}$ (which is equal to $-x_{ts}$) then the negative cost coefficient of arc (t, s) will force the flow through t (and also s) to increase as much as possible. Other constraints are the same as in the maxflow formulation.



Shortest path problem

Given a directed graph $G = (N, A)$, $|N| = n$, $|A| = m$.

c_{ij} is the length (or cost) of traversing arc (i, j) for each $(i, j) \in A$.

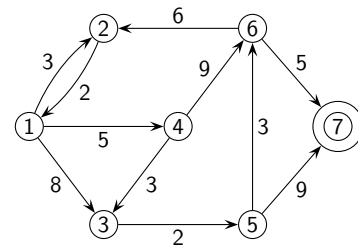
Length of a path is the sum of c_{ij} along the path.

Shortest path from i to k : one with minimum length among all possible paths from i to k .

Instead of finding the shortest path between two nodes in A , we can determine the shortest path **from all nodes to a designated node** (say, n) in the network.

Assumptions:

- From every $i \neq n$ there exists a path to n .
- $O(n) = \emptyset$ (no outgoing arc from n).



MCNF formulation:

Every node $i \neq n$ is given a unit (+1) supply, node n has a demand $-(n - 1)$.
 Arc capacities are infinite.

Formally, $u_{ij} = \infty, \quad \forall (i, j) \in A,$
 $b_i = 1, \quad i \neq n,$
 $b_n = -(n - 1).$

The MCNF problem to be solved is:

$$\min \sum_{(i,j) \in A} c_{ij}x_{ij}$$

subject to

$$\sum_{j \in O(i)} x_{ij} - \sum_{j \in I(i)} x_{ji} = b_i, \quad \forall i \in N$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in A$$

The arcs in the optimal basis of this problem form a tree rooted at node n . In this tree there is a unique path from every node $i \neq n$ to the root and this is the shortest path from i to n .

LOGIC CONSTRAINTS AND (M)IP

Indicator variables

Sometimes it is important to indicate that a variable is in a certain state, like $x > 0$ (see fixed charge problem). It can be achieved by the introduction of a 0/1 variable δ . We actually want to represent the condition

$$x > 0 \rightarrow \delta = 1, \tag{72}$$

which reads 'x greater than zero implies $\delta = 1$ '. Assuming that x has a known upper bound, S , (72) can be expressed as

$$x - S\delta \leq 0, \tag{73}$$

because for any $x > 0$ this constraint can only be satisfied if the maximum possible value of x , S , is subtracted from it. As such, $x > 0$ forces δ to be 1.

We may also wish to impose the condition

$$x = 0 \rightarrow \delta = 0, \tag{74}$$

which is equivalent to

$$\delta = 1 \rightarrow x > 0 \tag{75}$$

Combining (72) and (74) (or (75)) we get ' $\delta = 1$ if and only if $x > 0$ ' or

$$\delta = 1 \leftrightarrow x > 0. \tag{76}$$

Representing (75) or (76) by a constraint is not possible. If a tolerance level $\epsilon > 0$ is given to x for being different from zero then (75) can be rewritten as

$$\delta = 1 \rightarrow x \geq \epsilon \tag{77}$$

which can be imposed by the constraint

$$x - \epsilon\delta \geq 0.$$

Summarizing the above, if we want to express x is positive if and only if $\delta = 1$ (see (76)) it can be achieved by the following two inequalities and an indicator variable $\delta \in \{0, 1\}$:

$$x - S\delta \leq 0$$

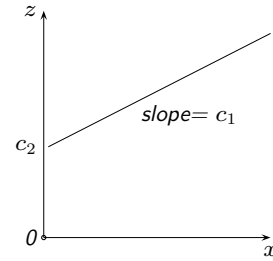
and

$$x - \epsilon\delta \geq 0.$$

Example 19 (The fixed charge problem) Let x represent the quantity of a product with unit production cost of c_1 . If the product is manufactured at all, there is a setup cost of c_2 associated with it. Thus, the cost of producing x units is

$$z = \begin{cases} 0, & \text{if } x = 0 \\ c_1x + c_2, & \text{if } x > 0. \end{cases}$$

Clearly, z is not linear, and not even continuous, see diagram.



We can introduce indicator variable δ such that $x > 0 \rightarrow \delta = 1$. Now the total cost z can be expressed as

$$z = c_1x + c_2\delta,$$

i.e., by the introduction of a 0/1 variable δ and an extra constraint for the logical condition $x > 0 \rightarrow \delta = 1$, we could model the situation as a MIP problem. **Note**, if cost is minimized condition $x = 0 \rightarrow \delta = 0$ need not be modelled.

Consider inequality

$$\sum_{j=1}^n a_jx_j \leq b. \tag{78}$$

We can use an indicator variable to show whether it is satisfied or not. More precisely, we want to model the following:

$$\delta = 1 \rightarrow \sum_{j=1}^n a_jx_j - b \leq 0. \tag{79}$$

Let S denote the largest value the expression $\sum a_jx_j - b$ can take. Now (78) can be imposed by $\sum_{j=1}^n a_jx_j - b \leq S(1 - \delta)$, or placing the variables on the LHS:

$$\sum_{j=1}^n a_jx_j + S\delta \leq b + S. \tag{80}$$

It is easy to see that if $\delta = 1$ the desired relation is enforced and if $\delta = 0$ no additional constraint is imposed as $\sum_{j=1}^n a_jx_j - b \leq S$ always holds by the definition of S .

To model the reverse of (79), namely,

$$\sum_{j=1}^n a_jx_j - b \leq 0 \rightarrow \delta = 1,$$

we consider the equivalent

$$\delta = 0 \rightarrow \sum_{j=1}^n a_jx_j - b > 0 \tag{81}$$

which can be modeled only if we allow for a tolerance $\epsilon > 0$ as in (77): $\sum_{j=1}^n a_jx_j \geq b + \epsilon$. Now we can rewrite (81) as

$$\delta = 0 \rightarrow -\sum_{j=1}^n a_jx_j + b + \epsilon \leq 0,$$

which can be represented by the constraint

$$\sum_{j=1}^n a_jx_j - (s - \epsilon)\delta \geq b + \epsilon, \tag{82}$$

where s is a lower bound for expression $\sum_j a_jx_j - b$.

To verify this expression, first substitute $\delta = 0$ which gives the desired relation, next take $\delta = 1$ and observe that (82) now reduces to $\sum_j a_jx_j - b \geq s$ which always holds by the definition of s .

To monitor whether constraint

$$\sum_{j=1}^n a_j x_j \geq b \tag{83}$$

is satisfied or not an indicator variable δ should be introduced. If (83) is transformed into a ' \leq ' the developed technique (79)–(82) can be used. First, we express

$$\delta = 1 \rightarrow \sum_{j=1}^n a_j x_j \geq b \tag{84}$$

(or the equivalent $\delta = 1 \rightarrow -\sum_j a_j x_j \leq -b$). If s and S denote the smallest and largest values expression $\sum_j a_j x_j - b$ can take, (84) is equivalent to $-\sum_j a_j x_j + b \leq -s(1 - \delta)$, or

$$\sum_{j=1}^n a_j x_j + s\delta \geq b + s. \tag{85}$$

Next, to express

$$\sum_{j=1}^n a_j x_j \geq b \rightarrow \delta = 1,$$

the quoted procedure results in

$$\sum_{j=1}^n a_j x_j - (S + \epsilon)\delta \leq b - \epsilon. \tag{86}$$

Example 20 Use a 0/1 variable to indicate whether or not constraint

$$4x_1 + x_2 \leq 2$$

is satisfied, where $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$ are continuous variables. The two relations we want to impose are

$$\delta = 1 \rightarrow 4x_1 + x_2 \leq 2 \tag{87}$$

$$4x_1 + x_2 \leq 2 \rightarrow \delta = 1. \tag{88}$$

First, determining s and S for $4x_1 + x_2 - 2$, we obtain $s = 0 + 0 - 2 = -2$, and $S = 4 + 1 - 2 = 3$. Applying (80), (87) can be expressed as

$$4x_1 + x_2 + 3\delta \leq 5 \quad (\text{being } 2 + 3).$$

Applying (82) with $\epsilon = 0.1$, (88) results in

$$4x_1 + x_2 + 2.1\delta \geq 2.1$$

What remains to be investigated is how an indicator variable can be used to force an equality constraint

$$\sum_j a_j x_j = b$$

to be satisfied or violated. Equality means \leq and \geq hold simultaneously. On the other hand, \neq means only one of the inequalities holds.

To express $\delta = 1 \rightarrow \sum_j a_j x_j = b$ we can use $\delta = 1$ to indicate that both inequalities hold simultaneously, i.e., we apply this δ in both (80) and (85).

With $\delta = 0$ we want to force one of the derived \leq and \geq constraints to be violated. To achieve it, $\delta', \delta'' \in \{0, 1\}$ are introduced and used to impose (82) and (86):

$$\begin{aligned} \sum_{j=1}^n a_j x_j - (s - \epsilon)\delta' &\geq b + \epsilon, \\ \sum_{j=1}^n a_j x_j - (S + \epsilon)\delta'' &\leq b - \epsilon. \end{aligned}$$

One more constraint is to be added to force the required condition

$$\delta' + \delta'' - \delta \leq 1.$$

Logical conditions and 0/1 variables

Many different types of logical conditions can be modelled by 0/1 variables (examples to come).

Notation

Propositions: P, Q, R, \dots, X, Y .

Example: P can stand for $x > 0$.

Connectives:

\vee inclusive 'or'

\wedge 'and'

\neg 'not'

\rightarrow 'implies' (or 'if ... then ...')

\leftrightarrow 'if and only if' ('iff')

The precedence order of the connectives from highest to lowest is ' \neg ', ' \wedge ', ' \vee ' and ' \rightarrow '.

Example 21 If X_i represents proposition 'Ingredient i is in the blend', $i \in \{A, B, C\}$, then proposition 'If ingredient A is in the blend then ingredient B or C (or both) must also be in the blend' can be expressed by $X_A \rightarrow (X_B \vee X_C)$ which is equivalent to $(X_A \rightarrow X_B) \vee (X_A \rightarrow X_C)$.

Correspondence can be established between propositions and indicator variables. If δ_i is a 0/1 indicator variable then X_i can stand for the proposition $\delta_i = 1$. In this way the following propositions and constraints are equivalent:

$X_1 \vee X_2$	\leftrightarrow	$\delta_1 + \delta_2 \geq 1$
$X_1 \wedge X_2$	\leftrightarrow	$\delta_1 = 1, \delta_2 = 1$
$\neg X_1$	\leftrightarrow	$\delta_1 = 0$
$X_1 \rightarrow X_2$	\leftrightarrow	$\delta_1 - \delta_2 \leq 0$
$X_1 \leftrightarrow X_2$	\leftrightarrow	$\delta_1 - \delta_2 = 0$

Example 22 Express the logical relationship of Example 21 with constraints containing 0/1 variables. Reminder: X_i represents proposition 'Ingredient i is in the blend', $i \in \{A, B, C\}$. We are interested in proposition 'If ingredient A is in the blend then ingredient B or C (or both) must also be in the blend', or $X_A \rightarrow (X_B \vee X_C)$.

Solution: Let indicator variables δ_i correspond to X_i , $i \in \{A, B, C\}$. Since $X_B \vee X_C$ can be expressed as $\delta_B + \delta_C \geq 1$, we want to impose $\delta_A = 1 \rightarrow \delta_B + \delta_C \geq 1$. This is the situation shown in (84). Now we have to compute s for $\delta_B + \delta_C - 1$ which is -1 . The required implication can be ensured by (85) which gives $\delta_B + \delta_C + (-1)\delta_A \geq 1 - 1$ or

$$-\delta_B - \delta_C + \delta_A \leq 0.$$

If there are other constraints in the model we have to add the above variables and the constraint to them.

Utilizing equivalent formulations

The following logical condition is often part of an optimization model (assuming all δ s are indicator variables):

$$\delta_1 = 1 \vee \delta_2 = 1 \vee \dots \vee \delta_n = 1 \rightarrow \delta = 1. \quad (89)$$

This condition can be imposed by the constraint

$$\delta_1 + \delta_2 + \dots + \delta_n - n\delta \leq 0. \quad (90)$$

(89) can also be written in an equivalent form:

$$\delta = 0 \rightarrow \delta_1 = 0 \wedge \delta_2 = 0 \wedge \dots \wedge \delta_n = 0. \quad (91)$$

(91) can be imposed by n constraints:

$$\begin{aligned} \delta_1 - \delta &\leq 0 \\ &\vdots \\ \delta_n - \delta &\leq 0. \end{aligned} \quad (92)$$

If all other LP constraints are similar to the ones in (92) then A^T satisfies property K. Taking the dual of the problem, we have a totally unimodular matrix that guarantees the all integer solution of the problem. Therefore, it is sufficient to solve the LP relaxation of the problem which will be integer. In other words, in this way constraints define the convex hull of integer solutions.

If not all LP constraints are in the form of (92) then this formulation is still preferred to (90) because it results in a tighter formulation of the problem.

Example 23 (Tighter formulation) We can illustrate why a series of constraints like (92) is preferable to a single constraint (90). While the two formulations are equivalent in logical and also in IP sense (actually, (90) is the sum of constraints in (92)) they are not equivalent in an LP sense. By adding together constraints in LP we generally get a larger feasible region, i.e., a looser formulation.

E.g., for $n \geq 3$ the solution

$$\delta_1 = \frac{1}{2}, \delta_2 = \delta_3 = \frac{1}{4}, \delta = \frac{1}{n}, \text{ all other } \delta_i = 0$$

satisfies (90) but does not satisfy (92).

This example shows that (92) is more effective in eliminating useless fractional solutions.

When linking a continuous x and an indicator δ variable in a constraint we used some upper bound on x . For instance,

$$x > 0 \rightarrow \delta = 1,$$

was represented by the constraint

$$x - S\delta \leq 0$$

where S is a true upper bound on x . In the light of the preceding discussion, it is quite important to keep S as small as possible and thus tightening the LP relaxation. It is usually the owner of the model who can determine the smallest realistic value for S .

Disjunctive Constraints in LP

In several real-life problems we do not require all constraints to hold but **at least one subset** of constraints. This can be stated as

$$R_1 \vee R_2 \vee \dots \vee R_M, \quad (93)$$

where R_i stands for the proposition 'Constraints in subset i are satisfied', $1 \leq i \leq M$. (93) is known as **disjunction of constraints**.

Introduce indicator variables δ_i to indicate whether constraints in R_i are satisfied. To fulfil (93), we impose

$$\delta_i = 1 \rightarrow R_i$$

for all subsets. Keeping in mind that R_i represents a (sub)set of constraints (that can be of any type), we have to apply the techniques of section 'Logic Constraints and (M)IP' to each constraint, all with the single indicator variable δ_i in the subset. Having done it for all $1 \leq i \leq M$, we can impose condition (93) by the constraint

$$\delta_1 + \delta_2 + \dots + \delta_M \geq 1.$$

If the condition to be modelled is 'at least k subsets have to be satisfied out of M ' then we can write

$$\delta_1 + \delta_2 + \dots + \delta_M \geq k.$$

Condition 'at most k of the subsets must be satisfied' can also be modelled. Now, indicator variables are used to express conditions

$$R_i \rightarrow \delta_i = 1.$$

This, again, involves handling all constraints individually in R_i according to their relation (\leq , $=$, \geq) and then imposing

$$\delta_1 + \delta_2 + \dots + \delta_M \leq k.$$

Interesting to note that the connective ' \wedge ' ('and') is usually not needed explicitly in LP since a conjunction of constraints simply involves a series of constraints holding simultaneously.

Example 24 (Limiting the number of active variables) *In the standard LP problem there are no more than m variables different from zero in an optimal solution. Sometimes this number has to be further restricted to $k < m$ for some (usually technological) reasons. Even if the original problem was an LP this restriction can be modelled by IP. We associate an indicator variable δ_j with each continuous variable x_j to indicate the condition $x_j > 0 \rightarrow \delta_j = 1$. As shown earlier, this can be imposed by the constraint $x_j - S_j \delta_j \leq 0$, where S_j is an upper bound on x_j . Now, we can impose the condition that no more than k continuous x_j variables can be nonzero in any feasible solution:*

$$\delta_1 + \delta_2 + \dots + \delta_n \leq k.$$

Some typical situations for this type of a condition are limiting the number of (i) different stocks in a portfolio, (ii) ingredients in a blend, (iii) cutting knives in cutting stock problems.

Example 25 (Discrete capacity expansion) *If a constraint represents a capacity limitation it may not be possible to expand it continuously but rather in steps, i.e., by installing a new machine or building a new storage.*

Suppose the initial RHS value of the constraint is b_0 which may be successively increased to $b_1 < b_2 < \dots < b_t$. The actual LP constraint will be one of the following:

$$\sum_{j=1}^n a_j x_j \leq b_i, \quad i = 0, 1, \dots, t,$$

where $i = 0$ represents the original constraint. The expansions incur costs $0 < c_1 < c_2 < \dots < c_t$, respectively. To model the situation, we introduce indicator variables δ_i , $i = 0, 1, \dots, t$ to represent the successive possible RHS values. We then have

$$\sum_{j=1}^n a_j x_j - b_0 \delta_0 - b_1 \delta_1 - \dots - b_t \delta_t \leq 0$$

$$\delta_0 + \delta_1 + \dots + \delta_t = 1, \tag{94}$$

and the following expression is added to the objective function

$$c_1 \delta_1 + c_2 \delta_2 + \dots + c_t \delta_t.$$

(94) ensures that exactly one of the alternatives is selected. It is interesting to note that in the LP relaxation (0/1 variables $\delta_0, \delta_1, \dots, \delta_t$ are relaxed to $0 \leq \delta_i \leq 1$, $i = 0, \dots, t$) the integrality requirement is automatically satisfied.

Example 26 (Maximax objectives) *(Not to be mistaken for minimax.) Assume we are given some linear expressions*

$$\sum_{j=1}^n a_{ij} x_j, \quad i = 1, \dots, m,$$

that are evaluated for different sets of x_j values. We may be interested in the following problem. Find x_1, \dots, x_n such that

$$\max \left(\max_i \left(\sum_{j=1}^n a_{ij} x_j \right) \right)$$

subject to conventional LP constraints.

While this problem shows similarities to the minimax objective (that can be modelled by LP), we have to treat it as a case of a disjunctive constraint and use IP. Introducing a continuous variable z , the model can be expressed as

$$\max \quad z$$

subject to

$$\sum_{j=1}^n a_{1j} x_j - z = 0$$

or

$$\dots$$

or

$$\sum_{j=1}^n a_{mj} x_j - z = 0$$

and the technique of handling disjunctive constraints can be used.

State-of-the-art in MIP

Currently, there is no single best general purpose (M)IP solver. The most widely used solution algorithms are based on the **Branch and Bound** principle. First LP relaxation is solved by IPM or the SM and re-optimizations are done by the dual SM.

As mentioned earlier, MIP is sensitive to the number of integer variables. E.g., if there are 100 0/1 variables in a model the number of all possible combinations of them is $2^{100} \approx 10^{30}$. A B&B method will test all of them (mostly implicitly). Depending on the **branching strategy**, the computational effort can vary enormously. Defining a good strategy often lies in the hands of the model owner who knows which are the 'important' integer variables that are useful to branch on.

Model formulation also heavily influences the efficiency of solution. Tighter formulations are more easily solved than less tight ones. Some algorithmic procedures can help tighten constraints but the modeller can do the most in this respect.

There are very few general purpose MIP solvers available (commercial or academic). They are all built around a state-of-the-art LP solver. Dual SM is a 'must' for MIP since a very large proportion of the solution time is spent on re-optimization.

Problems with several hundreds of 0/1 (and many continuous) variables and tens of thousands of constraints can be solved if properly formulated. The variation is big, anything can happen from efficient solution to unsolvability. However, skilled users have good chances to solve large real-life MIP problems.

MODELLING LANGUAGES

Need for modelling languages

Real-world LP and (M)IP problems tend to be very large. They have to be formulated, edited, verified and maintained. All these functions call for a **model management system**. Optimization modelling languages have been created to make the above activities possible in a convenient and efficient way. They also serve as a documentation of the work done.

A key requirement is to allow formulation in a similar way as humans naturally do it. This enables better understanding of the model.

Several optimization modelling languages have been developed and implemented since the early 1990s. They all share certain common features. Here are some of the most important ones.

- Model building is supported by a **model editor**.
- **Separation** of model and data.
- Variables and constraints are identified by **meaningful names**.
- Sophisticated **indexing** of variables is possible.
- **Variables** can be placed **on both sides** of constraints.
- **Algebraic expressions** can be used for defining constraints.
- **Logical relationships** can be defined.
- **Data** can be **imported**.
- **Macros** can be defined for repetitive parts of the model.
- **Comments** can be inserted anywhere in the model description.

Typical structure of a model file

Definition Part:

TITLE	Name of the problem
INDEX	Dimensions of the problem
DATA	Scalars, vectors, data files
VARIABLES	Vector variables
MACROS	Macros for repetitive parts

Model Part:

MODEL	
MAX or MIN	The objective function.
SUBJECT TO	The constraints.
BOUNDS	Individual bounds on variables.
FREE	Free variables.
INTEGER	Integer variables.
BINARY	Binary variables.
END	

The output of the model editor is a file that constitutes the input of an LP/MIP solver. Several formats for the output can be chosen. The most important one is the widely used industrial standard **MPS input format**.

Example of MPS format

```

NAME          Return maximization (simplified)
ROWS
  N Return
  L Row1
  L Row2
  L Row3
  L Row4
COLUMNS
  Var1  Return    5.4000  Row1    0.5000
  Var1  Row2      0.2500  Row3   -1.0000
  Var1  Row4      1.0000
  Var2  Return    7.3000  Row2    0.5000
  Var2  Row3      1.0000
  Var3  Return   12.9600  Row1    0.6000
  Var3  Row2      0.6000
  Var4  Return   -6.0000  Row1   -1.0000
  Var5  Return   -9.0000  Row2   -1.0000
RHS
  Constr Return    800.0000 Row1     80.0000
  Constr Row2     40.0000 Row4    150.0000
RANGES
  Range1 Row4     50.0000
BOUNDS
  UP Bound1 Var4    20.0000
  UP Bound1 Var5    10.0000
ENDATA

```

Note, the fields have fixed positions and names cannot be longer than 8 characters. The direction of optimization (min or max) cannot be defined in MPS format.

The following example highlights the use of the different components of a modelling language.

Aggregate production planning

A company wants to

- distribute production capacity among products,
- determine production and inventory levels in order to meet demand for each product

in every month while maximizing its profit.

The company has three products.

Planning is done for twelve months.

Prices, demands, production capacity and cost are fixed for the whole period.

The decision variables are monthly production, sales and inventory.

```

TITLE
  Production Planning;
INDEX
  product = 1..3;
  month   = {Jan, Feb, Mar, Apr, May, Jun,
            Jul, Aug, Sep, Oct, Nov, Dec};
DATA
  Price[product]      := (99.00, 240.00, 750.00);
  Demand[month,product] := DATAFILE(demand.dat);
  ProdCapacity[product] := (1000.0, 550.0, 220.0);
  ProdCost[product]    := (55.60, 176.40, 585.30);
  InvCost              := 6.55; \\ Inventory unit cost
VARIABLES
  Inventory[product,month] -> Invt \\ Short name
  Prod[product,month]      \\ Production
  Sales[product,month]     -> Sale \\ Short name
MACRO
  Revenues := SUM(product,month: Price * Sales);
  TotalCost := SUM(product,month: InvCost * Inventory
                  + ProdCost * Prod);
MODEL
MAX Profit = Revenues - TotalCost;
SUBJECT TO
  InventoryBalance[product,month]: \\ Constraint name
  Inventory[month] = Inventory[month-1] + Prod - Sales;
BOUNDS
  Sales < Demand;
  Prod < ProdCapacity;
  Inventory[product,month=Jan..Nov] < 8000;
  Inventory[product,Dec] = 1800;
END

```

If short names are given they are used in the generated MPS file (where names are limited to 8 characters).

The model editor will generate decision variables for Inventory, Production and Sales for each product for each month of the year (108 variables). Using the short names, the finally generated variable names as used in the MPS format may look as follows:

```

Invt1Jan  Invt2Jan  Invt3Jan
Invt1Feb  Invt2Feb  Invt3Feb
...
Invt1Dec  Invt2Dec  Invt3Dec

Prod1Jan  Prod2Jan  Prod3Jan
...
Prod1Dec  Prod2Dec  Prod3Dec

Sale1Jan  Sale2Jan  Sale3Jan
...
Sale1Dec  Sale2Dec  Sale3Dec

```

There will be an inventory constraint generated for each product for months Feb to Dec, altogether 33 of them. Starting inventory is assumed to be zero. Inventory levels are further restricted by individual bounds.

Other bounds restrict sales (no more can be sold than the demand is) and production (production capacity cannot be exceeded).

While the above model as described in this language is quite understandable, the corresponding MPS file is much less intuitive. In case of large scale problems the advantages of a modelling system are even more obvious. Actually, such problems simply cannot be handled without a proper modelling system.