# Linear Programming (LP)

Linear programming is the most commonly used optimisation technique in various applications. There are many reasons for this. Linear programs are relatively easy to formulate, use and understand. The LP optimisation techniques are also efficient and well developed. A surprisingly large set of real-life problems can be represented as linear programs, or approximated sufficiently well with linear programs. Finally, several more advanced modelling and solution techniques are based on linear programming, such as quadratic programming, fractional programming, integer linear programming, mixed integer linear programming, constraint logic programming, multi-objective linear programming, linear goal programming, etc.

## *Different formulations of LP problems*

The standard formulation of an LP problem is minimization of a linear objective function subject to linear inequality constraints:

$$\min \sum_{j=1}^{n} c_j x_j$$

s.t. (1)

$$\sum_{j=1}^{n} A_{ij} x_j \le b_i \qquad i=1,\ldots,m$$

$$x_j \ge 0 \qquad j=1,\ldots,n$$

Instead of writing the problem using sums and iteration constructs, it is often more convenient to use the vector and matrix notation:

$$\min z = cx$$

s.t. (2)

$$Ax \le b$$

$$x \ge 0$$

$z$ is the linear objective function to be minimized, $x = [x_1, \ldots, x_n]^T$ is a column-vector of $n$ non-negative decision variables, $c = [c_1, \ldots, c_n]$ is a row-vector of cost coefficients, $A$ is an $n \times m$ matrix of constraint coefficients, and $b = [b_1, \ldots, b_m]^T$ is a column vector of right hand side (RHS) coefficients (sometimes called the *resource vector*).

Other equivalent formulations can also be used:

- Maximization: max $cx$ is equivalent to –min –$cx$.
- Greater than inequalities: $Ax \ge b$ is equivalent to $-Ax \le -b$.
- Equality constraints: $Ax = b$ is equivalent to double inequalities $Ax \ge b$ and $Ax \le b$.
- Non-positive variables: $y \le 0$ can be substituted by $-x$ where $x \ge 0$.
- Unconstrained variables: unconstrained y can be substituted by difference $x_1$-$x_2$ where $x_1,x_2 \ge 0$.
- Nonzero lower bound for variable: $y \ge y^{min}$ can be substituted by $x=y-y^{min}$ where $x \ge 0$.

When formulating an LP problem it is convenient to allow a more general formulation

$$\min (\max)\ cx$$

s.t. (3)

$$b^{min} \leq Ax \leq b^{max} \quad \text{(two sided constraints)}$$

$$x^{min} \leq x \leq x^{max} \quad \text{(lower and upper bounds)}$$

In this form each constraint may be two-sided and decision variables may have non-zero lower bounds and upper bounds. Choosing some $b^{min} = b^{max}$ yields equality constraints. Upper bounds can be disabled by choosing $\infty$ as upper bound.

## *Graphical solution of LP problems*

Small problems with two decision variables can be visualized and solved graphically. Each linear inequality constraint divides the plane into two half-planes: the feasible and infeasible side. The feasible region is a convex polygon formed as the intersection of these half-planes. The objective function is a direction in the plane. The optimum solution is always in some corner of this polygon. (In rare cases, two corners may give the same optimal solution. Then also all points on their connecting edge are optimal).

The intersections between constraints are so-called *basic solutions* of the LP problem (see later).

Example: Consider the problem

$$\min z = -2\ x_1 - 3\ x_2$$

s.t.

$$3\ x_1 + 2\ x_2 \leq 24$$

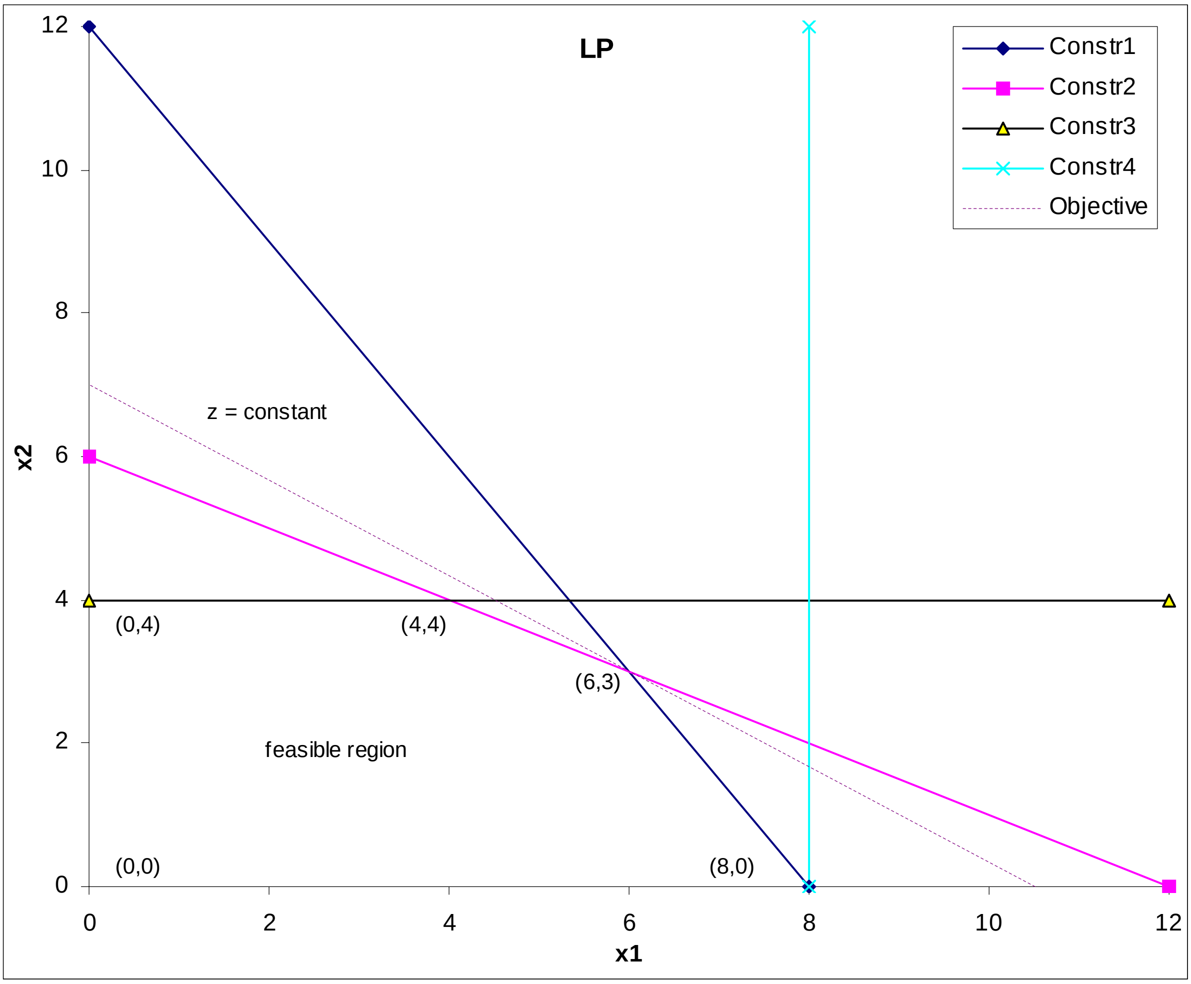
$$x_1 + 2\ x_2 \leq 12$$

$$x_2 \leq 4$$

$$x_1 \leq 8$$

$$x_1, x_2 \geq 0$$

This problem is illustrated graphically in Figure 1. The feasible region is a convex polygon with corners (0,0), (0,4), (4,4), (6,3), and (8,0). The dotted line shows a level where $z$ is constant. The minimal feasible solution $z = -21$ is found by shifting the level line as far to the Northeast as possible while still touching the feasible region. As seen in the figure, this happens at point (6,3). Similarly, the maximal feasible solution $z = 0$ is found at (0,0) by shifting the line as far to Southwest as possible.




## *Transforming an LP problem into the canonical form*

Prior to solving an LP problem numerically, it is normally converted into a format, where each constraint is an equality that includes a *unique slack variable*. This format may either contain or not contain upper bounds for variables. The LP solution software will normally do this automatically. However, it is useful to understand the transformation because it makes it possible to understand the optimization process and results.

### Canonical form with upper bounds

The following transformations are applied:

- Inequality constraints $Ax \leq b$ are converted to equality constraints by adding non-negative slack variables into $Ax + s = b$, where $s \geq 0$.
- Inequality constraints $Ax \geq b$ are multiplied by –1 and then converted by adding non-negative slack variables into $-Ax + s = -b$, where $s \geq 0$.
- The equality constraints $Ax = b$ are augmented with so called artificial variables into $Ax + s = b$, where $0 \leq s \leq 0$.

- Two-sided constraints $b^{min} \le Ax \le b^{max}$ can be efficiently handled by using a combined slack/surplus variable: $Ax + s = b^{max}$, where $0 \le s \le b^{max}-b^{min}$

After these transformations, the *canonical form* of the LP problem with upper bounds is obtained:

$$\min z = cx$$

s.t. (4)

$$Ax + s = b$$

$$0 \le x \le x^{max}$$

$$0 \le s \le s^{max}$$

The x-variables are called *structural variables*. The *s*-variables are called *slacks* for short. The only difference between the *s*-variables and *x*-variables is that the objective function coefficients of *s*-variables are zeroes and the constraint coefficients of the *s*-variables form an identity matrix. If we do not want to highlight these differences, we can extend the *x*-vector to include the *s*-variables and augment the *c* and $x^{max}$ vectors and the *A*-matrix correspondingly. Then the LP problem can be written as

$$\min z = c'x'$$

s.t.

$$A'x' = b,$$

$$0 \le x' \le x'^{max}.$$

In this form the matrix A' consists of the original A corresponding to the original *x*-variables and an $m \times m$ identity matrix I corresponding to the *s*-variables. Non-zero lower bounds and negative variables can be modelled easily through substitution of variables, as described earlier.

For example the canonical form with upper bounds of the sample problem is

$$\min z = -2\ x_1 - 3\ x_2$$

s.t.

$$3\ x_1 + 2\ x_2 + s_1 \quad = 24$$

$$x_1 + 2\ x_2 \quad + s_2 = 12$$

$$0 \le x_1 \le 8$$

$$0 \le x_2 \le 4$$

$$0 \le s_1,\ s_2 \le \infty$$

## Canonical form without upper bounds

The following transformations are applied:

- Upper bounds for variables are treated as inequalities.

- Inequality constraints $Ax \leq b$ are converted to equality constraints by adding non-negative slack variables into $Ax + s = b$, where $s \geq 0$.

- Inequality constraints $Ax \geq b$ are multiplied by –1 and then converted by adding non-negative slack variables into $-Ax + s = -b$, where $s \geq 0$.

- Equality constraints $Ax = b$ are treated as two separate inequalities $Ax \leq b$ and $Ax \geq b$.

- Two-sided constraints $b^{min} \leq Ax \leq b^{max}$ are treated as two separate inequalities.

After these transformations, the *canonical form* of the LP problem without upper bounds is obtained:

$$\min z = cx$$

s.t. (5)

$$Ax + s = b$$

$$x, s \geq 0$$

In the canonical form without upper bounds, the problem is typically larger (has more constraints and variables) than in the upper bounds formulation. The previous sample problem can be written in canonical format without upper bounds as

$$\min z = -2\, x_1 - 3\, x_2$$

s.t.

$$\begin{aligned} 3\, x_1 + 2\, x_2 + s_1 \qquad\qquad &= 24 \\ x_1 + 2\, x_2 \quad + s_2 \qquad &= 12 \\ x_2 \qquad + s_3 \quad &= 4 \\ x_1 \qquad\qquad + s_4 &= 8 \\ x_1, x_2, s_1, s_2, s_3, s_4 &\geq 0 \end{aligned}$$

## *The Tabular Simplex algorithm*

In the tabular Simplex algorithm, the optimisation problem is organized as a table of numbers corresponding to the equality constraints. Equations can, of course, be reordered, multiplied by factors and summed together without affecting their validity. The Simplex algorithm performs such operations in order to transform the equations into a format where the optimal solution is obvious.

Consider an LP problem in the canonical format. Writing also the objective function as an equality, yields the form

$$\min z$$

s.t. (6)

$$z - c_1x_1 - c_2x_2 - \ldots - c_nx_n = 0$$

$$a_{11}\, x_1 + a_{12}\, x_2 + \ldots + a_{1n}\, x_n + s_1 = b_1$$

$$a_{21}\, x_1 + a_{22}\, x_2 + \ldots + a_{2n}\, x_n + s_2 = b_2$$

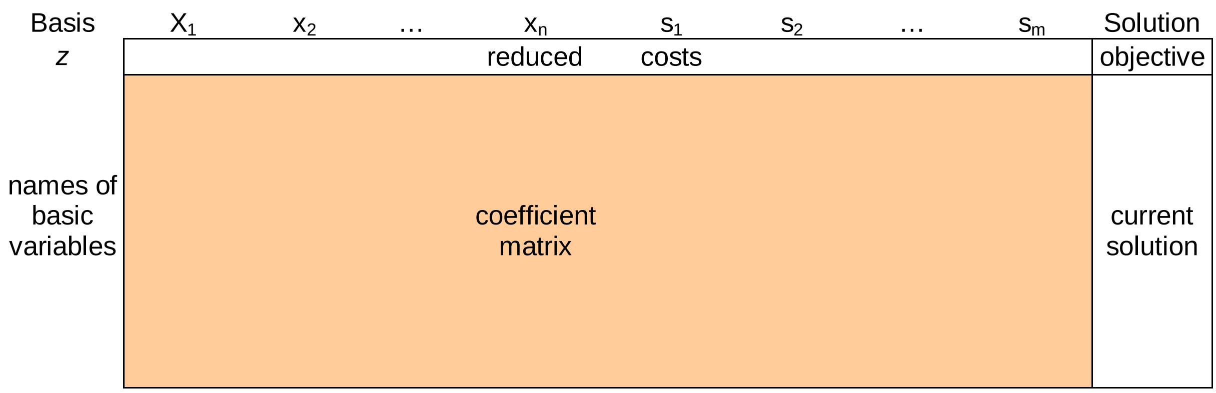
$$\vdots$$

$$a_{m1}\, x_1 + a_{m2}\, x_2 + \ldots + a_{mn}\, x_n + s_m = b_m$$

$$0 \le x \le x^{max}$$

$$0 \le s \le s^{max}$$

The Simplex algorithm is based on exploring *basic solutions* of the problem. A *basis* is a set of linearly independent column vectors that in a linear combination can represent every vector. The basic solutions correspond to intersections between constraints in graphical representation of the problem. During the Simplex algorithm, exactly *m* variables $x^B$ are *basic*. The remaining *n* variables $x^N$ are *non-basic*. The non-basic variables are set to their lower (or upper) bounds. The *m* basic variables are then solved from the system of *m* linear equalities.




The simplex table for performing the necessary computations is organized as follows:

- To the left of the simplex table is a column showing the names of the variables that form the current basis $x^B$. The current basis may contain any selection of *m* variables out of the *n* *x*-variables and *m* *s*-variables. The order in which the basic variables are listed identifies from which equation that variable has been solved.
- The names of each variable are listed on top of the simplex table.
- The z-row shows the so-called *reduced costs* for each variable.
- The current objective function value appears in the upper right hand corner.
- Below the reduced costs is the coefficient matrix.

- The last column shows the current solution, i.e. the values of the basic variables $x^B$.

During the algorithm, the simplex table is maintained in a format where the sub-matrix corresponding to the basic variables is an identity matrix.

In the initial simplex table, the basis consists of all the slacks and all *x*-variables are non-basic. This is convenient, because the sub-matrix corresponding to the slacks is an identity matrix. The non-basic variables are set to zero (their lower bound). The initial solution vector $x^B$ is then equal to the *b*-vector. For the moment, we are not concerned about the feasibility of the solution.

| Basis | $x_1$ | $x_2$ | … | $x_n$ | $s_1$ | $s_2$ | … | $s_m$ | Solution |
|---|---|---|---|---|---|---|---|---|---|
| *z* | $-c_1$ | $-c_2$ | … | $-c_n$ | 0 | 0 | … | 0 | 0 |
| $s_1$ | $a_{11}$ | $a_{12}$ | … | $a_{1n}$ | 1 | 0 | … | 0 | $b_1$ |
| $s_2$ | $a_{21}$ | $a_{22}$ | … | $a_{2n}$ | 0 | 1 | | | $b_2$ |
| | | | ⋮ | | | | ⋮ | | ⋮ |
| $s_m$ | $a_{m1}$ | $a_{m2}$ | … | $a_{mn}$ | | | … | 1 | $b_m$ |

The initial simplex table corresponding previous sample problem in canonical format without upper bounds is

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | Solution |
|---|---|---|---|---|---|---|---|
| *z* | 2 | 3 | 0 | 0 | 0 | 0 | 0 |
| $s_1$ | 3 | 2 | 1 | 0 | 0 | 0 | 24 |
| $s_2$ | 1 | 2 | 0 | 1 | 0 | 0 | 12 |
| $s_3$ | 0 | 1 | 0 | 0 | 1 | 0 | 4 |
| $s_4$ | 1 | 0 | 0 | 0 | 0 | 1 | 8 |

During each iteration of the Simplex algorithm, we want to enter a new variable to the basis and remove an old variable from the basis so that the value of *z* improves (decreases) and the feasibility of all feasible variables is maintained.

The entering variable is determined by examining the **reduced costs** of the non-basic variables. The reduced cost of a variable represents the net decrease in *z* when the non-basic variable is increased (and the basic variables are adjusted to satisfy the constraints). The reduced costs of basic variables are by definition zero.

For example, in the sample table, increasing the value for $x_2$ by +1 would (in order to maintain the equality $z\text{-}cx=0$) decrease *z* by $c_2 = 3$. Thus, *z* decreases if the reduced cost for the entering variable is positive. When minimizing, the variable with the largest reduced cost is typically chosen to enter the basis. When all reduced costs are non-positive, the solution is *optimal* and the algorithm terminates.

The column corresponding to the entering variable is called the *pivot column y* and it is highlighted in the sample table. The pivot column shows what happens to the basic variables when the (non-basic) entering variable is moved from its bound. When the entering variable increases from its lower bound, a positive *y*-element indicates that the corresponding basic variable decreases (to maintain the equality), a negative element indicates that the basic variable increases, and a zero indicates that the basic variable is unaffected. For example in the initial table, increasing $x_2$ by +1 decreases $s_1$ by 2, $s_2$ by 2, and $s_3$ by 1, and leaves $s_4$ unaffected.

If the variables are bounded only from below, only the positive *y*-elements corresponding to decreasing basic variables are of interest. To protect any basic variable from becoming infeasible (negative), the variable that first reaches zero value must leave the basis. This is the variable that corresponds to the smallest ratio $\Theta = \min_i x^B_i/y_i$ for $y_i>0$ and $x^B_i \geq 0$. Negative $x^B_i$ indicate infeasible variables. Protecting an already infeasible variable from becoming even more infeasible is not a concern at this phase. The smallest ratio $\Theta$ indentifies the *pivot row*, i.e. the equation from which the entering variable must be solved. If no leaving variable is found (all *y*-elements are non-positive), that indicates that the problem is *unbounded* and the algorithm terminates.

The following table shows the ratios for determining $\Theta$ in the sample problem. The smallest ratio is 4 and it appears on the third equation row. Thus, $s_3$ is the variable that must leave the basis.

| Basis | $X_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | Solution | $x^B_i/y_i$ |
|---|---|---|---|---|---|---|---|---|
| *z* | 2 | 3 | 0 | 0 | 0 | 0 | 0 | |
| $s_1$ | 3 | 2 | 1 | 0 | 0 | 0 | 24 | 12 |
| $s_2$ | 1 | 2 | 0 | 1 | 0 | 0 | 12 | 6 |
| $s_3$ | 0 | **1** | 0 | 0 | 1 | 0 | 4 | 4 |
| $s_4$ | 1 | 0 | 0 | 0 | 0 | 1 | 8 | - |

After the entering and leaving variables have been determined, the so-called pivot step is performed. This step will return the table to a format where the basic variables correspond to an identity matrix. This is done by applying the Gauss-Jordan elimination method on the simplex table in such a manner that the *pivot element* at the intersection of the highlighted pivot row and column becomes one and the remaining parts of the pivot column become zero.

The Gauss-Jordan method is applied as follows. The pivot row is divided by the pivot element to make the pivot element equal to 1. In the table above, the pivot element was already 1 so the division was redundant. The remaining elements $y_i$ of the pivot column are eliminated through row-operations by subtracting $y_i$ times the transformed pivot row from each row i, including the reduced cost row (row zero).

The following table shows the necessary Gauss-Jordan elimination operations on our sample problem. The last column indicates what row operations have been applied. Each cell shows the formula for computing the new value. Observe that the pivot row needs no transformation in this case, because the pivot element $y_3$ is already one. Also, the last row needs no processing, because $y_4$ is already zero. Also, because the pivot row contains zeroes on columns $x_1$, $s_1$, $s_2$ and $s_4$, no operations on these columns are needed.

| Basis | $X_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | Solution | operation |
|---|---|---|---|---|---|---|---|---|
| *z* | 2 | 3-3×1 | 0 | 0 | 0-3×1 | 0 | 0-3×4 | -= 3×r |
| $s_1$ | 3 | 2-2×1 | 1 | 0 | 0-2×1 | 0 | 24-2×4 | -= 2×r |
| $s_2$ | 1 | 2-2×1 | 0 | 1 | 0-2×1 | 0 | 12-2×4 | -= 2×r |
| $x_2$ | 0 | 1/1 | 0 | 0 | 1/1 | 0 | 4/1 | /= 1 |
| $s_4$ | 1 | 0 | 0 | 0 | 0 | 1 | 8 | -= 0×r |

The following table shows the result after the row-operations. The *z*-value has improved from 0 to –12. The variable $x_2$ is now in the basis on row 3, which is indicated in the basis-column. The reduced cost of $x_2$ is zero and the column of $x_2$ is part of the identity matrix formed by all basic variables. To see the identity matrix, the basic columns would have to be sorted into the order specified in the basis column. Finally, observe that the column of $s_3$ that has been removed from the basis, is no longer a part of the identity matrix.

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | Solution |
|---|---|---|---|---|---|---|---|
| $z$ | 2 | 0 | 0 | 0 | -3 | 0 | -12 |
| $s_1$ | 3 | 0 | 1 | 0 | -2 | 0 | 16 |
| $s_2$ | 1 | 0 | 0 | 1 | -2 | 0 | 4 |
| $x_2$ | 0 | 1 | 0 | 0 | 1 | 0 | 4 |
| $s_4$ | 1 | 0 | 0 | 0 | 0 | 1 | 8 |

This table is not yet optimal, because the reduced cost for $x_1$ is positive. Choosing x1 as the entering variable and computing the $x^B/y$ ratios identifies row 2 as the pivot row and $s_2$ as the leaving variable.

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | Solution | $x^B{}_i/y_i$ |
|---|---|---|---|---|---|---|---|---|
| $z$ | 2 | 0 | 0 | 0 | -3 | 0 | -12 | |
| $s_1$ | 3 | 0 | 1 | 0 | -2 | 0 | 16 | 5.333 |
| $s_2$ | **1** | 0 | 0 | 1 | -2 | 0 | 4 | 4 |
| $x_2$ | 0 | 1 | 0 | 0 | 1 | 0 | 4 | - |
| $s_4$ | 1 | 0 | 0 | 0 | 0 | 1 | 8 | 8 |

The new table after pivoting is still not optimal, because now the reduced cost of $s_3$ is positive. This time the pivot row is row 1 with leaving variable $s_1$.

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | Solution | $x^B{}_i/y_i$ |
|---|---|---|---|---|---|---|---|---|
| $z$ | 0 | 0 | 0 | -2 | 1 | 0 | -20 | |
| $s_1$ | 0 | 0 | 1 | -3 | **4** | 0 | 4 | 1 |
| $x_1$ | 1 | 0 | 0 | 1 | -2 | 0 | 4 | - |
| $x_2$ | 0 | 1 | 0 | 0 | 1 | 0 | 4 | 4 |
| $s_4$ | 0 | 0 | 0 | -1 | 2 | 1 | 4 | 2 |

After the row-operations we obtain an optimal table with $z = -21$.

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | Solution |
|---|---|---|---|---|---|---|---|
| $z$ | 0 | 0 | -0.25 | -1.25 | 0 | 0 | -21 |
| $s_3$ | 0 | 0 | 0.25 | -0.75 | 1 | 0 | 1 |
| $x_1$ | 1 | 0 | 0.5 | -0.5 | 0 | 0 | 6 |
| $x_2$ | 0 | 1 | -0.25 | 0.75 | 0 | 0 | 3 |
| $s_4$ | 0 | 0 | -0.5 | 0.5 | 0 | 1 | 2 |

## Shadow prices

The reduced costs for slack variables in the optimal simplex table are called *shadow prices* for constraints. The shadow price indicates how the objective function changes when the right-hand side (resource) of corresponding constraints is modified a little. The shadow price is zero for inactive (non-binding) constraints, and non-zero for active (binding) constraints.

In optimal table of the previous example, constraints 1 and 2 are active, which is reflected by negative shadow prices (reduced costs for slack variables $s_1$ and $s_2$). This indicates that reducing the right hand side (resource) of the two first vectors disproves (increases) the objective function value, and increasing the resource improves (decreases) the objective function value by 0.25 and 1.25 units, correspondingly. The shadow prices for constraints 3 and 4 are zero, which means that adjusting their right-hand side does not affect the optimum. When using shadow prices for sensitivity analysis, it is necessary to understand that the shadow price may be valid only subject to a small (even infinitely small) change in the resource. A more robust method for sensitivity analysis is to modify the model parameters and re-optimize the model.

## *Tabular Simplex with upper bounds*

Upper bounds of structural variables can be handled as separate inequality constraints, as was done in the previous example. This, however, increases the size of the problem unnecessarily. A much more efficient technique is to handle the upper bounds directly in the algorithm.

Let us consider a variable with lower bound zero and non-negative upper bound $0 \le x_j \le x_j^{max}$. We define a new variable $x_j^* = x_j^{max} - x_j$. Both variables have the same bounds, but when $x_j$ is on its upper bound, $x_j^*$ is zero and when $x_j$ is zero, $x_j^* = x_j^{max}$. The idea is to substitute $x_j$ with $x_j^*$ whenever $x_j$ reaches its upper bound, and substitute $x_j^*$ with $x_j$ when $x_j^*$ reaches its upper bound.

Determining the pivot row is slightly more complicated when upper bounds are present. It is necessary to check if some variable reaches its upper bound when the entering variable is moved. This is done by checking the ratios $(x^B - x^{B,max}_i)/y_i$ also for $y_i < 0$. Prior to removing a variable from the basis based on the upper bound check, the variable substitution should be applied. The third possibility is that the entering variable $x_j$ itself reaches its upper bound before any basic variable reaches its either bound. To allow the algorithm to work also when the current solution is infeasible, the test is omitted for variables that are already on the wrong side of their bound. Thus, the maximum step to make is

$$\Theta = \min \left\{ \begin{array}{ll} x_i^B / y_i & \text{when } y_i > 0 \wedge x_i^B \ge 0 \\ \left(x_i^B - x_i^{B,\max}\right) / y_i & \text{when } y_i < 0 \wedge x_i^B \le x_i^{B,\max} \\ x_j^{\max} & \end{array} \right\} \tag{7}$$

Consider the sample problem with upper bounds $0 \le x_1 \le 8$, and $0 \le x_2 \le 4$. The simplex table is

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | Solution | $\Theta$ |
|---|---|---|---|---|---|---|
| *z* | 2 | 3 | 0 | 0 | 0 | |
| $s_1$ | 3 | 2 | 1 | 0 | 24 | 12 |
| $s_2$ | 1 | 2 | 0 | 1 | 12 | 6 |

The $x^B/y_i$ –ratios would allow an increase of 6 in $x_2$. However, in this case the minimum $\Theta$ = 4 corresponding to the entering variable itself.

The upper bound substitution can be done very easily on the simplex table. A non-basic variable is substituted by subtracting $x_j^{max}$ times the variable column from the solution vector and negating then the column. Substituting $x_2 = 4 - x_2^*$ in the previous table gives

| Basis | $x_1$ | $x_2$* | $s_1$ | $s_2$ | Solution | Θ |
|---|---|---|---|---|---|---|
| *z* | 2 | -3 | 0 | 0 | -12 | |
| $s_1$ | 3 | -2 | 1 | 0 | 16 | 5.333 |
| $s_2$ | 1 | -2 | 0 | 1 | 4 | 4 |

Comparing this solution with the previous example after the first iteration we observe that the basis is different, but the solution is essentially the same (same values for *z*, $x_1$, $x_2$, $s_1$ and $s_2$). Next we enter $x_1$ and remove $s_2$ from row 2 with minimum Θ = 4.

| Basis | $x_1$ | $x_2$* | $s_1$ | $s_2$ | Solution | Θ |
|---|---|---|---|---|---|---|
| *z* | 0 | 1 | 0 | -2 | -20 | |
| $s_1$ | 0 | 4 | 1 | -3 | 4 | 1 |
| $x_1$ | 1 | -2 | 0 | 1 | 4 | 2 |

This table is not optimal because the reduced cost of $x_2$* is 1. Because $y_2$ is negative and the corresponding basic variable $x_1$ has a finite upper bound of 8, we must compute the ratio as (4-8)/-2 = 2. This time the ratio for the first row is smallest. Thus, we enter $x_2$* and remove $s_1$ from row 1 with minimum Θ = 1.

| Basis | $x_1$ | $x_2$* | $s_1$ | $s_2$ | Solution | operation |
|---|---|---|---|---|---|---|
| *z* | 0 | 0 | -0.25 | -1.25 | -21 | |
| $x_2$* | 0 | 1 | 0.25 | -0.75 | 1 | $x_2$* = 4-$x_2$ |
| $x_1$ | 1 | 0 | 0.5 | -0.5 | 6 | |

This table is optimal. To obtain the solution in terms of the original variables, we can substitute the $x_2$* with 4-$x_2$. Because $x_2$* is basic, this substitution affects only the $x_2$ row.

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | Solution | operation |
|---|---|---|---|---|---|---|
| *z* | 0 | 0 | -0.25 | -1.25 | -21 | |
| $x_2$ | 0 | -1 | 0.25 | -0.75 | -3 | *= -1 |
| $x_1$ | 1 | 0 | 0.5 | -0.5 | 6 | |

However, negating the column of $x_2$ has made the identity matrix element –1. To restore the identity matrix, the row must yet be multiplied by –1.

| Basis | $x_1$ | $x_2$ | $s_1$ | $s_2$ | Solution |
|---|---|---|---|---|---|
| *z* | 0 | 0 | -0.25 | -1.25 | -21 |
| $x_2$ | 0 | 1 | -0.25 | 0.75 | 3 |
| $x_1$ | 1 | 0 | 0.5 | -0.5 | 6 |

This same solution was found in the previous example without the upper bounds technique.

## *Handling Infeasibility*

So far, we have assumed that the initial solution to the LP problem is feasible, i.e., all non-basic variables are within their bounds. This situation is true for example in problems without upper bounds, where all constraints are of less or equal type with non-negative *b*. The Simplex algorithm will maintain the feasibility, while improving the objective function. If the initial solution is infeasible for some variables, the algorithm will preserve the feasibility of any feasible variables. The algorithm may also accidentally make some or all of the infeasible variables feasible, but cannot guarantee that all variable eventually become feasible.

To guarantee that the algorithm always finds a feasible optimal solution, it is necessary to handle the infeasibilities somehow. The so-called *two-phase technique* is based on first solving a related LP problem whose optimal solution provides a feasible solution for the original problem. In the second phase, the original problem is solved starting from the found feasible solution. If the problem is in the canonical form with upper bounds (4) and we start with a slack basis, the infeasibilities are due to some $b_i<0$ or some $b_i>s_i^{max}$. The objective in the first phase is then to minimize these infeasibilities:

$$\min z' = \sum_{i:b_i^B > s_i^{max}} s_i - \sum_{i:b_i^B < 0} s_i$$

s.t. (8)

$$Ax + s = b,$$

$$0 \le x \le x^{max}$$

$$0 \le s \le s^{max}.$$

A more efficient single-pass technique is to use an objective function that is a linear combination of the original objective function and penalty terms:

$$\min (f \cdot c + F)\, x \tag{9}$$

Here $x$ is the extended variable vector (including the slacks), $f$ is a non-negative scaling factor (typically in the range [0,1], ideal value depends on problem type), and $F$ is a row-vector of penalty terms

$$F_j = \begin{cases} -1 \text{ when } x_j < 0 \\ +1 \text{ when } x_j > x_j^{max} \\ \ 0 \text{ when } x_j \text{ is feasible} \end{cases} \tag{10}$$

The objective function is modified during the algorithm. When a variable becomes feasible, the corresponding $F$-coefficient is set to zero. When the overall solution becomes feasible, the original objective function is restored by assigning $f$ = 1. If the algorithm stops with an optimal but infeasible solution, this can mean that $f$ is too large. In such cases $f$ is decreased or set to zero and the algorithm proceeds. If no feasible solution is found even with zero $f$, this means that the problem is infeasible (has no feasible solutions).

The penalty method is more efficient than the two-phase method, because it allows advancing simultaneously towards a feasible and optimal solution. There are additional benefits with the penalty method when solving large or numerically difficult LP problems. Sometimes the feasibility of the problem may be lost during the iterations due to numerical inaccuracy in the computations. The penalty function can be easily reapplied whenever this happens.

## *The Revised Simplex algorithm*

The tabular Simplex algorithm is suitable only for rather small problems, because the complete Simplex table has to be recomputed and maintained during the iterations. Large problems are usually solved using the so-called *revised Simplex* algorithm. During each iteration the Simplex algorithm needs only access to one row and two columns of the simplex table: the z-row, pivot-column, and solution-column. The idea of revised Simplex is to maintain information about the current basis inverse revising (updating) it as the basis changes and to compute only necessary parts of the current simplex table.

We describe a generic variant of the Revised Simplex algorithm, which is commonly used for solving large LP problems (Taha, 1982, Flannery et al., 1988, Aittoniemi, 1988). The Simplex algorithm improves $z$ iteratively by moving from one basic solution to another. In a basic solution the ($n$-$m$) non-basic variables are set to their lower or upper bounds and the values for the $m$ basic variables are determined so that the constraints are satisfied. $A$, $x$ and $c$ are partitioned as

$$A = [B \mid N],\ \ x = \begin{bmatrix} x^B \\ \hline x^N \end{bmatrix},\ c = [c^B \mid c^N], \tag{11}$$

where $B$ is the non-singular basis matrix, $N$ is the non-basic matrix, $x^B$ is the vector of basic variables, $x^N$ is the vector of non-basic variables, and $c^B$ and $c^N$ are the cost coefficients for the basic and non-basic variables. The constraints become then

$$Bx^B + Nx^N = b. \tag{12}$$

To satisfy the constraints, $x^B$ is solved in terms of $x^N$:

$$x^B = B^{-1}(b - Nx^N). \tag{13}$$

The basic solution is *feasible* when the basic variables are within their bounds $0 \le x^B \le x^{B,max}$ (any non-basic variable is by definition on its either bound and therefore feasible). Introducing row vector $\pi$ such that

$$\pi = c^B B^{-1}, \tag{14}$$

and substituting $x^B$ and $\pi$ into the objective function gives

$$z = c^B x^B + c^N x^N = \pi b + (c^N - \pi N)x^N = \pi b - dx^N, \tag{15}$$

where

$$d = \pi N - c^N \tag{16}$$

is the row vector of the *reduced costs* for non-basic variables. The reduced costs indicate how $z$ changes when non-basic variables are moved away from their bounds. The solution is optimal, if the *optimality condition* holds:

$d_j \leq 0$ for all non-basic variables that are at their lower bound, and
$d_j \geq 0$ for all non-basic variables that are at their upper bound.

Any non-basic variable that does not satisfy the optimality condition will improve the objective function value when entering the basis. Entering the basis means increasing the variable from its lower bound or decreasing it from its upper bound. However, this movement will affect the values of the basic variables. If the value of a non-basic variable $x_j^N$ is changed by $\Delta x_j^N$, then, according to (13), the vector of basic variable values changes by $-y\Delta x_j^N$, where we have introduced the so-called *pivot column*

$$y = B^{-1} N_{\cdot j}. \tag{17}$$

The maximum allowed change $\Delta x_j^N$ for the entering variable must be chosen such that the feasibility of all basic variables is maintained, and one of the basic variables reaches its either bound, i.e., leaves the basis. This means finding the largest (in magnitude) change $\Delta x_j^N$ such that the following inequalities are still satisfied (and the entering variable itself does not exceed it's bound):

$$0 \leq x^B - y\Delta x_j^N \leq x^{B,\max}. \tag{18}$$

The following procedure implements the generic Revised Simplex algorithm.

## *Algorithm summary: Generic Revised Simplex with upper bounds*

1. Start from some feasible basic solution (11).
2. Compute the basic variables $x^B$ from (13), $\pi$ from (14), and the reduced costs $d$ from (16).
3. Find a variable $x_j^N$ to enter the basis so that $z$ improves. For variables at lower bound this condition is $d_j>0$, and at upper bound $d_j<0$. If there is no such variable, stop with optimal $x^B$.
4. Compute the pivot column $y$ from (17).
5. Find the variable to leave the basis so that feasibility is maintained. The leaving variable is the one that according to (18) reaches its upper or lower bound first when the entering is moved away from its bound. If there is no such variable, stop with an unbounded solution.

6. Update the basis and go back to 2.

Different variants of the Revised Simplex algorithm use different techniques for representing and maintaining the basis and/or its inverse. For example, in the Product Form of Inverse (PFI) the basis inverse is represented as a product of elimination matrices, and each iteration contributes to one additional factor ($B^{-1} = E_k E_{k-1} \ldots E_1$). In the Elimination Form of Inverse (EFI) a triangular factorisation of the non-inverted basis is maintained instead ($B = LU$, where L is lower triangular and U is upper triangular).