



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)

[Contents](#)

[Featured content](#)

[Current events](#)

[Random article](#)

[Donate to Wikipedia](#)

[Wikipedia store](#)

Interaction

[Help](#)

[About Wikipedia](#)

[Community portal](#)

[Recent changes](#)

[Contact page](#)

Tools

[What links here](#)

[Related changes](#)

[Upload file](#)

[Special pages](#)

[Permanent link](#)

[Page information](#)

[Wikidata item](#)

[Cite this page](#)

Print/export

[Create a book](#)

[Download as PDF](#)

[Printable version](#)

Languages

[Afrikaans](#)

[العربية](#)

[Беларуская](#)

[Bosanski](#)

[Català](#)

[Čeština](#)

[Deutsch](#)

[Español](#)

[Euskara](#)

[فارسی](#)

[Français](#)

[한국어](#)

[हिन्दी](#)

[Bahasa Indonesia](#)

[Italiano](#)

[עברית](#)

[Magyar](#)

[Nederlands](#)

[日本語](#)

[Norsk nynorsk](#)

[Polski](#)

[Português](#)

[Română](#)

[Русский](#)

[Sicilianu](#)

[Simple English](#)

[Slovenščina](#)

[Српски / srpski](#)

[Srpskohrvatski /](#)

[српскохрватски](#)

Article [Talk](#)

[Read](#)

[Edit](#)

[More](#) ▾



# Linear programming

From Wikipedia, the free encyclopedia

**Linear programming** (**LP**; also called **linear optimization**) is a method to achieve the best outcome (such as maximum profit or lowest cost) in a **mathematical model** whose requirements are represented by **linear relationships**. Linear programming is a special case of mathematical programming (**mathematical optimization**).

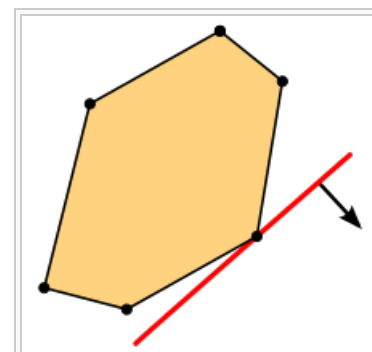
More formally, linear programming is a technique for the **optimization** of a **linear objective function**, subject to **linear equality** and **linear inequality constraints**. Its **feasible region** is a **convex polytope**, which is a set defined as the **intersection** of finitely many **half spaces**, each of which is defined by a linear inequality. Its objective function is a **real-valued affine (linear) function** defined on this polyhedron. A linear programming **algorithm** finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.

Linear programs are problems that can be expressed in **canonical form** as

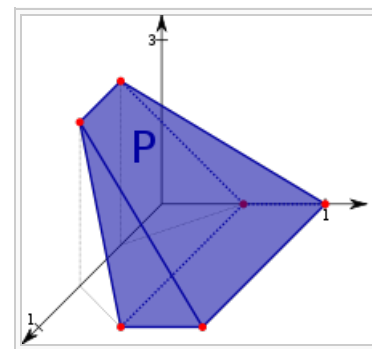
$$\begin{aligned} &\text{maximize} && \mathbf{c}^T \mathbf{x} \\ &\text{subject to} && \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ &\text{and} && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

where **x** represents the vector of variables (to be determined), **c** and **b** are **vectors** of (known) coefficients, **A** is a (known) **matrix** of coefficients, and **(.)<sup>T</sup>** is the **matrix transpose**. The expression to be maximized or minimized is called the *objective function* (**c<sup>T</sup>x** in this case). The inequalities **Ax ≤ b** and **x ≥ 0** are the constraints which specify a **convex polytope** over which the objective function is to be optimized. In this context, two vectors are **comparable** when they have the same dimensions. If every entry in the first is less-than or equal-to the corresponding entry in the second then we can say the first vector is less-than or equal-to the second vector.

Linear programming can be applied to various fields of study. It is widely used in business and **economics**, and is also utilized for some engineering problems. Industries that use linear programming models include transportation, energy, telecommunications, and manufacturing. It has proved useful in modeling diverse types of problems in planning, **routing**, **scheduling**, **assignment**, and design.



A pictorial representation of a simple linear program with two variables and six inequalities. The set of feasible solutions is depicted in yellow and forms a **polygon**, a 2-dimensional **polytope**. The linear cost function is represented by the red line and the arrow: The red line is a **level set** of the cost function, and the arrow indicates the direction in which we are optimizing.



A closed feasible region of a problem with three variables is a convex **polyhedron**. The surfaces giving a fixed value of the objective function are **planes** (not shown). The linear programming problem is to find a point on the polyhedron that is on the plane with the highest possible value.

## Contents [hide]

- History
- Uses
- Standard form
  - Example
- Augmented form (slack form)
  - Example
- Duality
  - Example
  - Another example
- Covering/packing dualities
  - Examples
- Complementary slackness
- Theory
  - Existence of optimal solutions

- 8.2 Optimal vertices (and rays) of polyhedra
- 9 Algorithms
  - 9.1 Basis exchange algorithms
    - 9.1.1 Simplex algorithm of Dantzig
    - 9.1.2 Criss-cross algorithm
  - 9.2 Interior point
    - 9.2.1 Ellipsoid algorithm, following Khachiyan
    - 9.2.2 Projective algorithm of Karmarkar
    - 9.2.3 Path-following algorithms
  - 9.3 Comparison of interior-point methods versus simplex algorithms
  - 9.4 Approximate Algorithms for Covering/Packing LPs
- 10 Open problems and recent work
- 11 Integer unknowns
- 12 Integral linear programs
- 13 Solvers and scripting (programming) languages
- 14 See also
- 15 Notes
- 16 References
- 17 Further reading
- 18 External links

## History [\[edit\]](#)

The problem of solving a system of linear inequalities dates back at least as far as [Fourier](#), who in 1827 published a method for solving them,<sup>[1]</sup> and after whom the method of [Fourier–Motzkin elimination](#) is named.

The first linear programming formulation of a problem that is equivalent to the general linear programming problem was given by [Leonid Kantorovich](#) in 1939, who also proposed a method for solving it.<sup>[2]</sup> He developed it during [World War II](#) as a way to plan expenditures and returns so as to reduce costs to the army and increase losses incurred by the enemy. About the same time as Kantorovich, the Dutch-American economist [T. C. Koopmans](#) formulated classical economic problems as linear programs. Kantorovich and Koopmans later shared the 1975 [Nobel prize in economics](#).<sup>[1]</sup> In 1941, [Frank Lauren Hitchcock](#) also formulated transportation problems as linear programs and gave a solution very similar to the later Simplex method;<sup>[2]</sup> Hitchcock had died in 1957 and the Nobel prize is not awarded posthumously.

During 1946–1947, [George B. Dantzig](#) independently developed general linear programming formulation to use for planning problems in US Air Force. In 1947, Dantzig also invented the [simplex method](#) that for the first time efficiently tackled the linear programming problem in most cases. When Dantzig arranged meeting with [John von Neumann](#) to discuss his Simplex method, Neumann immediately conjectured the theory of [duality](#) by realizing that the problem he had been working in [game theory](#) was equivalent. Dantzig provided formal proof in an unpublished report "A Theorem on Linear Inequalities" on January 5, 1948.<sup>[3]</sup> Postwar, many industries found its use in their daily planning.

Dantzig's original example was to find the best assignment of 70 people to 70 jobs. The computing power required to test all the permutations to select the best assignment is vast; the number of possible configurations exceeds the number of particles in the observable universe. However, it takes only a moment to find the optimum solution by posing the problem as a linear program and applying the [simplex algorithm](#). The theory behind linear programming drastically reduces the number of possible solutions that must be checked.

The linear programming problem was first shown to be solvable in polynomial time by [Leonid Khachiyan](#) in 1979, but a larger theoretical and practical breakthrough in the field came in 1984 when [Narendra Karmarkar](#) introduced a new [interior-point method](#) for solving linear-programming problems.

## Uses [\[edit\]](#)

Linear programming is a widely used field of optimization for several reasons. Many practical problems in [operations research](#) can be expressed as linear programming problems. Certain special cases of linear



Leonid Kantorovich



programming, such as *network flow* problems and *multicommodity flow* problems are considered important enough to have generated much research on specialized algorithms for their solution. A number of algorithms for other types of optimization problems work by solving LP problems as sub-problems. Historically, ideas from linear programming have inspired many of the central concepts of optimization theory, such as *duality*, *decomposition*, and the importance of *convexity* and its generalizations. Likewise, linear programming is heavily used in *microeconomics* and company management, such as planning, production, transportation, technology and other issues. Although the modern management issues are ever-changing, most companies would like to *maximize profits* or minimize costs with limited resources. Therefore, many issues can be characterized as linear programming problems.

## Standard form [\[edit\]](#)

*Standard form* is the usual and most intuitive form of describing a linear programming problem. It consists of the following three parts:

- **A linear function to be maximized**

e.g.  $f(x_1, x_2) = c_1x_1 + c_2x_2$

- **Problem constraints** of the following form

e.g.

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 \leq b_2$$

$$a_{31}x_1 + a_{32}x_2 \leq b_3$$

- **Non-negative variables**

e.g.

$$x_1 \geq 0$$

$$x_2 \geq 0$$

The problem is usually expressed in *matrix form*, and then becomes:

$$\max\{c^T x \mid Ax \leq b \wedge x \geq 0\}$$

Other forms, such as minimization problems, problems with constraints on alternative forms, as well as problems involving negative *variables* can always be rewritten into an equivalent problem in standard form.

## Example [\[edit\]](#)

Suppose that a farmer has a piece of farm land, say  $L \text{ km}^2$ , to be planted with either wheat or barley or some combination of the two. The farmer has a limited amount of fertilizer,  $F$  kilograms, and insecticide,  $P$  kilograms. Every square kilometer of wheat requires  $F_1$  kilograms of fertilizer and  $P_1$  kilograms of insecticide, while every square kilometer of barley requires  $F_2$  kilograms of fertilizer and  $P_2$  kilograms of insecticide. Let  $S_1$  be the selling price of wheat per square kilometer, and  $S_2$  be the selling price of barley. If we denote the area of land planted with wheat and barley by  $x_1$  and  $x_2$  respectively, then profit can be maximized by choosing optimal values for  $x_1$  and  $x_2$ . This problem can be expressed with the following linear programming problem in the standard form:

Maximize:  $S_1 \cdot x_1 + S_2 \cdot x_2$  (maximize the revenue—revenue is the "objective function")

Subject to:  $x_1 + x_2 \leq L$  (limit on total area)

$$F_1 \cdot x_1 + F_2 \cdot x_2 \leq F \text{ (limit on fertilizer)}$$

$$P_1 \cdot x_1 + P_2 \cdot x_2 \leq P \text{ (limit on insecticide)}$$

$$x_1 \geq 0, x_2 \geq 0 \text{ (cannot plant a negative area).}$$

Which in matrix form becomes:

$$\begin{aligned} &\text{maximize } \begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &\text{subject to } \begin{bmatrix} 1 & 1 \\ F_1 & F_2 \\ P_1 & P_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} L \\ F \\ P \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \end{aligned}$$

## Augmented form (slack form) [\[edit\]](#)

Linear programming problems can be converted into an *augmented form* in order to apply the common form of the *simplex algorithm*. This form introduces non-negative *slack variables* to replace inequalities with equalities in

the constraints. The problems can then be written in the following [block matrix](#) form:

Maximize  $Z$ :

$$\begin{bmatrix} 1 & -\mathbf{c}^T & 0 \\ 0 & \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$$

$$\mathbf{x}, \mathbf{x}_s \geq 0$$

where  $\mathbf{x}_s$  are the newly introduced slack variables, and  $Z$  is the variable to be maximized.

### Example [\[edit\]](#)

The example above is converted into the following augmented form:

Maximize:  $S_1 \cdot x_1 + S_2 \cdot x_2$  (objective function)

subject to:  $x_1 + x_2 + x_3 = L$  (augmented constraint)

$F_1 \cdot x_1 + F_2 \cdot x_2 + x_4 = F$  (augmented constraint)

$P_1 \cdot x_1 + P_2 \cdot x_2 + x_5 = P$  (augmented constraint)

$x_1, x_2, x_3, x_4, x_5 \geq 0$

where  $x_3, x_4, x_5$  are (non-negative) slack variables, representing in this example the unused area, the amount of unused fertilizer, and the amount of unused insecticide.

In matrix form this becomes:

Maximize  $Z$ :

$$\begin{bmatrix} 1 & -S_1 & -S_2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & F_1 & F_2 & 0 & 1 & 0 \\ 0 & P_1 & P_2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Z \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ L \\ F \\ P \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq 0.$$

## Duality [\[edit\]](#)

*Main article: [Duality \(optimization\)](#)*

Every linear programming problem, referred to as a *primal* problem, can be converted into a [dual problem](#), which provides an upper bound to the optimal value of the primal problem. In matrix form, we can express the *primal* problem as:

Maximize  $\mathbf{c}^T \mathbf{x}$  subject to  $\mathbf{Ax} \leq \mathbf{b}$ ,  $\mathbf{x} \geq 0$ ;

with the corresponding **symmetric** dual problem,

Minimize  $\mathbf{b}^T \mathbf{y}$  subject to  $\mathbf{A}^T \mathbf{y} \geq \mathbf{c}$ ,  $\mathbf{y} \geq 0$ .

An alternative primal formulation is:

Maximize  $\mathbf{c}^T \mathbf{x}$  subject to  $\mathbf{Ax} \leq \mathbf{b}$ ;

with the corresponding **asymmetric** dual problem,

Minimize  $\mathbf{b}^T \mathbf{y}$  subject to  $\mathbf{A}^T \mathbf{y} = \mathbf{c}$ ,  $\mathbf{y} \geq 0$ .

There are two ideas fundamental to duality theory. One is the fact that (for the symmetric dual) the dual of a dual linear program is the original primal linear program. Additionally, every feasible solution for a linear program gives a bound on the optimal value of the objective function of its dual. The [weak duality](#) theorem states that the objective function value of the dual at any feasible solution is always greater than or equal to the objective function value of the primal at any feasible solution. The [strong duality](#) theorem states that if the primal has an optimal solution,  $\mathbf{x}^*$ , then the dual also has an optimal solution,  $\mathbf{y}^*$ , and  $\mathbf{c}^T \mathbf{x}^* = \mathbf{b}^T \mathbf{y}^*$ .

A linear program can also be unbounded or infeasible. Duality theory tells us that if the primal is unbounded then the dual is infeasible by the weak duality theorem. Likewise, if the dual is unbounded, then the primal must be infeasible. However, it is possible for both the dual and the primal to be infeasible. As an example, consider the linear program:

Maximize:  $2x_1 - x_2$

Subject to:  $x_1 - x_2 \leq 1$

$$-x_1 + x_2 \leq -2$$

$$x_1, x_2 \geq 0.$$

### Example [\[edit\]](#)

Revisit the above example of the farmer who may grow wheat and barley with the set provision of some  $L$  land,  $F$  fertilizer and  $P$  pesticide. Assume now that  $y$  unit prices for each of these means of production (inputs) are set by a planning board. The planning board's job is to minimize the total cost of procuring the set amounts of inputs while providing the farmer with a floor on the unit price of each of his crops (outputs),  $S_1$  for wheat and  $S_2$  for barley. This corresponds to the following linear programming problem:

$$\begin{aligned} \text{Minimize: } & L \cdot y_L + F \cdot y_F + P \cdot y_P && \text{(minimize the total cost of the means of production as the} \\ & && \text{"objective function")} \\ \text{subject} & & & \\ \text{to: } & y_L + F_1 \cdot y_F + P_1 \cdot y_P \geq S_1 && \text{(the farmer must receive no less than } S_1 \text{ for his wheat)} \\ & y_L + F_2 \cdot y_F + P_2 \cdot y_P \geq S_2 && \text{(the farmer must receive no less than } S_2 \text{ for his barley)} \\ & y_L, y_F, y_P \geq 0 && \text{(prices cannot be negative).} \end{aligned}$$

In matrix form this becomes:

$$\begin{aligned} \text{Minimize: } & [L \quad F \quad P] \begin{bmatrix} y_L \\ y_F \\ y_P \end{bmatrix} \\ \text{subject to: } & \begin{bmatrix} 1 & F_1 & P_1 \\ 1 & F_2 & P_2 \end{bmatrix} \begin{bmatrix} y_L \\ y_F \\ y_P \end{bmatrix} \geq \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}, \quad \begin{bmatrix} y_L \\ y_F \\ y_P \end{bmatrix} \geq 0. \end{aligned}$$

The primal problem deals with physical quantities. With all inputs available in limited quantities, and assuming the unit prices of all outputs is known, what quantities of outputs to produce so as to maximize total revenue?

The dual problem deals with economic values. With floor guarantees on all output unit prices, and assuming the available quantity of all inputs is known, what input unit pricing scheme to set so as to minimize total expenditure?

To each variable in the primal space corresponds an inequality to satisfy in the dual space, both indexed by output type. To each inequality to satisfy in the primal space corresponds a variable in the dual space, both indexed by input type.

The coefficients that bound the inequalities in the primal space are used to compute the objective in the dual space, input quantities in this example. The coefficients used to compute the objective in the primal space bound the inequalities in the dual space, output unit prices in this example.

Both the primal and the dual problems make use of the same matrix. In the primal space, this matrix expresses the consumption of physical quantities of inputs necessary to produce set quantities of outputs. In the dual space, it expresses the creation of the economic values associated with the outputs from set input unit prices.

Since each inequality can be replaced by an equality and a slack variable, this means each primal variable corresponds to a dual slack variable, and each dual variable corresponds to a primal slack variable. This relation allows us to speak about complementary slackness.

### Another example [\[edit\]](#)

Sometimes, one may find it more intuitive to obtain the dual program without looking at the program matrix. Consider the following linear program:

$$\begin{aligned} \text{Minimize } & \sum_{i=1}^m c_i x_i + \sum_{j=1}^n d_j t_j \\ \text{subject to } & \sum_{i=1}^m a_{ij} x_i + e_j t_j \geq g_j, \quad 1 \leq j \leq n \\ & f_i x_i + \sum_{j=1}^n b_{ij} t_j \geq h_i, \quad 1 \leq i \leq m \\ & x_i \geq 0, t_j \geq 0, \quad 1 \leq i \leq m, 1 \leq j \leq n \end{aligned}$$

We have  $m + n$  conditions and all variables are non-negative. We shall define  $m + n$  dual variables:  $y_j$  and  $s_i$ . We get:

$$\begin{aligned}
 &\text{Minimize} && \sum_{i=1}^m c_i x_i + \sum_{j=1}^n d_j t_j \\
 &\text{subject to} && \sum_{i=1}^m a_{ij} x_i \cdot y_j + e_j t_j \cdot y_j \geq g_j \cdot y_j, \quad 1 \leq j \leq n \\
 &&& f_i x_i \cdot s_i + \sum_{j=1}^n b_{ij} t_j \cdot s_i \geq h_i \cdot s_i, \quad 1 \leq i \leq m \\
 &&& x_i \geq 0, t_j \geq 0, \quad 1 \leq i \leq m, 1 \leq j \leq n \\
 &&& y_j \geq 0, s_i \geq 0, \quad 1 \leq j \leq n, 1 \leq i \leq m
 \end{aligned}$$

Since this is a minimization problem, we would like to obtain a dual program that is a lower bound of the primal. In other words, we would like the sum of all right hand side of the constraints to be the maximal under the condition that for each primal variable the sum of its [coefficients](#) do not exceed its coefficient in the linear function. For example,  $x_1$  appears in  $n + 1$  constraints. If we sum its constraints' coefficients we get  $a_{1,1}y_1 + a_{1,2}y_2 + \dots + a_{1,n}y_n + f_1s_1$ . This sum must be at most  $c_1$ . As a result, we get:

$$\begin{aligned}
 &\text{Maximize} && \sum_{j=1}^n g_j y_j + \sum_{i=1}^m h_i s_i \\
 &\text{subject to} && \sum_{j=1}^n a_{ij} y_j + f_i s_i \leq c_i, \quad 1 \leq i \leq m \\
 &&& e_j y_j + \sum_{i=1}^m b_{ij} s_i \leq d_j, \quad 1 \leq j \leq n \\
 &&& y_j \geq 0, s_i \geq 0, \quad 1 \leq j \leq n, 1 \leq i \leq m
 \end{aligned}$$

Note that we assume in our calculations steps that the program is in standard form. However, any linear program may be transformed to standard form and it is therefore not a limiting factor.

## Covering/packing dualities [\[edit\]](#)

A [covering LP](#) is a linear program of the form:

$$\begin{aligned}
 &\text{Minimize: } \mathbf{b}^T \mathbf{y}, \\
 &\text{subject to: } A^T \mathbf{y} \geq \mathbf{c}, \mathbf{y} \geq 0,
 \end{aligned}$$

such that the matrix  $A$  and the vectors  $\mathbf{b}$  and  $\mathbf{c}$  are non-negative.

The dual of a covering LP is a [packing LP](#), a linear program of the form:

$$\begin{aligned}
 &\text{Maximize: } \mathbf{c}^T \mathbf{x}, \\
 &\text{subject to: } A \mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0,
 \end{aligned}$$

such that the matrix  $A$  and the vectors  $\mathbf{b}$  and  $\mathbf{c}$  are non-negative.

### Examples [\[edit\]](#)

Covering and packing LPs commonly arise as a [linear programming relaxation](#) of a combinatorial problem and are important in the study of [approximation algorithms](#).<sup>[4]</sup> For example, the LP relaxations of the [set packing problem](#), the [independent set problem](#), and the [matching problem](#) are packing LPs. The LP relaxations of the [set cover problem](#), the [vertex cover problem](#), and the [dominating set problem](#) are also covering LPs.

Finding a [fractional coloring](#) of a [graph](#) is another example of a covering LP. In this case, there is one constraint for each vertex of the graph and one variable for each [independent set](#) of the graph.

## Complementary slackness [\[edit\]](#)

### Covering/packing-problem pairs

Covering problems	Packing problems
Minimum set cover	Maximum set packing
Minimum vertex cover	Maximum matching
Minimum edge cover	Maximum independent set
$v \cdot t \cdot e$	



It is possible to obtain an optimal solution to the dual when only an optimal solution to the primal is known using the complementary slackness theorem. The theorem states:

Suppose that  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$  is primal feasible and that  $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m)$  is dual feasible. Let  $(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_m)$  denote the corresponding primal slack variables, and let  $(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n)$  denote the corresponding dual slack variables. Then  $\mathbf{x}$  and  $\mathbf{y}$  are optimal for their respective problems if and only if

- $\mathbf{x}_j \mathbf{z}_j = 0$ , for  $j = 1, 2, \dots, n$ , and
- $\mathbf{w}_i \mathbf{y}_i = 0$ , for  $i = 1, 2, \dots, m$ .

So if the  $i$ -th slack variable of the primal is not zero, then the  $i$ -th variable of the dual is equal to zero. Likewise, if the  $j$ -th slack variable of the dual is not zero, then the  $j$ -th variable of the primal is equal to zero.

This necessary condition for optimality conveys a fairly simple economic principle. In standard form (when maximizing), if there is slack in a constrained primal resource (i.e., there are "leftovers"), then additional quantities of that resource must have no value. Likewise, if there is slack in the dual (shadow) price non-negativity constraint requirement, i.e., the price is not zero, then there must be scarce supplies (no "leftovers").

## Theory [\[edit\]](#)

---

### Existence of optimal solutions [\[edit\]](#)

Geometrically, the linear constraints define the [feasible region](#), which is a [convex polyhedron](#). A [linear function](#) is a [convex function](#), which implies that every [local minimum](#) is a [global minimum](#); similarly, a linear function is a [concave function](#), which implies that every [local maximum](#) is a [global maximum](#).

An optimal solution need not exist, for two reasons. First, if two constraints are inconsistent, then no feasible solution exists: For instance, the constraints  $\mathbf{x} \geq 2$  and  $\mathbf{x} \leq 1$  cannot be satisfied jointly; in this case, we say that the LP is *infeasible*. Second, when the [polytope](#) is unbounded in the direction of the gradient of the objective function (where the gradient of the objective function is the vector of the coefficients of the objective function), then no optimal value is attained.

### Optimal vertices (and rays) of polyhedra [\[edit\]](#)

Otherwise, if a feasible solution exists and if the (linear) objective function is bounded, then the optimum value is always attained on the boundary of optimal level-set, by the [maximum principle](#) for [convex functions](#) (alternatively, by the *minimum principle* for [concave functions](#)): Recall that linear functions are both convex and concave. However, some problems have distinct optimal solutions: For example, the problem of finding a feasible solution to a system of linear inequalities is a linear programming problem in which the objective function is the zero function (that is, the constant function taking the value zero everywhere): For this feasibility problem with the zero-function for its objective-function, if there are two distinct solutions, then every convex combination of the solutions is a solution.

The vertices of the polytope are also called *basic feasible solutions*. The reason for this choice of name is as follows. Let  $d$  denote the number of variables. Then the fundamental theorem of linear inequalities implies (for feasible problems) that for every vertex  $\mathbf{x}^*$  of the LP feasible region, there exists a set of  $d$  (or fewer) inequality constraints from the LP such that, when we treat those  $d$  constraints as equalities, the unique solution is  $\mathbf{x}^*$ . Thereby we can study these vertices by means of looking at certain subsets of the set of all constraints (a discrete set), rather than the continuum of LP solutions. This principle underlies the [simplex algorithm](#) for solving linear programs.

## Algorithms [\[edit\]](#)

---

See also: [List of numerical analysis topics § Linear programming](#)

### Basis exchange algorithms [\[edit\]](#)

#### Simplex algorithm of Dantzig [\[edit\]](#)

The [simplex algorithm](#), developed by [George Dantzig](#) in 1947, solves LP problems by constructing a feasible solution at a vertex of the [polytope](#) and then walking along a path on the edges of the polytope to vertices with non-decreasing values of the objective function until an optimum is reached for sure. In many practical problems, "stalling" occurs: Many pivots are made with no increase in the objective function.<sup>[5][6]</sup> In rare practical problems, the usual versions of the simplex algorithm may actually "cycle".<sup>[6]</sup> To avoid cycles, researchers developed new pivoting rules.<sup>[7][8][5][6][9][10]</sup>

In practice, the simplex [algorithm](#) is quite efficient and can be guaranteed to find the global optimum if certain

precautions against *cycling* are taken. The simplex algorithm has been proved to solve "random" problems efficiently, i.e. in a cubic number of steps,<sup>[11]</sup> which is similar to its behavior on practical problems.<sup>[5][12]</sup>

However, the simplex algorithm has poor worst-case behavior: Klee and Minty constructed a family of linear programming problems for which the simplex method takes a number of steps exponential in the problem size.<sup>[5][8][9]</sup> In fact, for some time it was not known whether the linear programming problem was solvable in *polynomial time*, i.e. of *complexity class P*.

#### Criss-cross algorithm [\[edit\]](#)

Like the simplex algorithm of Dantzig, the *criss-cross algorithm* is a basis-exchange algorithm that pivots between bases. However, the criss-cross algorithm need not maintain feasibility, but can pivot rather from a feasible basis to an infeasible basis. The criss-cross algorithm does not have *polynomial time-complexity* for linear programming. Both algorithms visit all  $2^D$  corners of a (perturbed) *cube* in dimension  $D$ , the *Klee–Minty cube*, in the *worst case*.<sup>[10][13]</sup>

#### Interior point [\[edit\]](#)

##### Ellipsoid algorithm, following Khachiyan [\[edit\]](#)

This is the first *worst-case polynomial-time* algorithm for linear programming. To solve a problem which has  $n$  variables and can be encoded in  $L$  input bits, this algorithm uses  $O(n^4L)$  pseudo-arithmetic operations on numbers with  $O(L)$  digits. Khachiyan's *algorithm* and his long standing issue was resolved by Leonid Khachiyan in 1979 with the introduction of the *ellipsoid method*. The convergence analysis have (real-number) predecessors, notably the *iterative methods* developed by Naum Z. Shor and the *approximation algorithms* by Arkadi Nemirovski and D. Yudin.

##### Projective algorithm of Karmarkar [\[edit\]](#)

Khachiyan's algorithm was of landmark importance for establishing the polynomial-time solvability of linear programs. The algorithm was not a computational break-through, as the simplex method is more efficient for all but specially constructed families of linear programs.

However, Khachiyan's algorithm inspired new lines of research in linear programming. In 1984, N. Karmarkar proposed a *projective method* for linear programming. *Karmarkar's algorithm* improved on Khachiyan's worst-case polynomial bound (giving  $O(n^{3.5}L)$ ). Karmarkar claimed that his algorithm was much faster in practical LP than the simplex method, a claim that created great interest in interior-point methods.<sup>[14]</sup>

##### Path-following algorithms [\[edit\]](#)

In contrast to the simplex algorithm, which finds an optimal solution by traversing the edges between vertices on a polyhedral set, interior-point methods move through the interior of the feasible region. Since then, many interior-point methods have been proposed and analyzed. Early successful implementations were based on *affine scaling* variants of the method. For both theoretical and practical purposes, *barrier function* or *path-following* methods have been the most popular since the 1990s.<sup>[15]</sup>

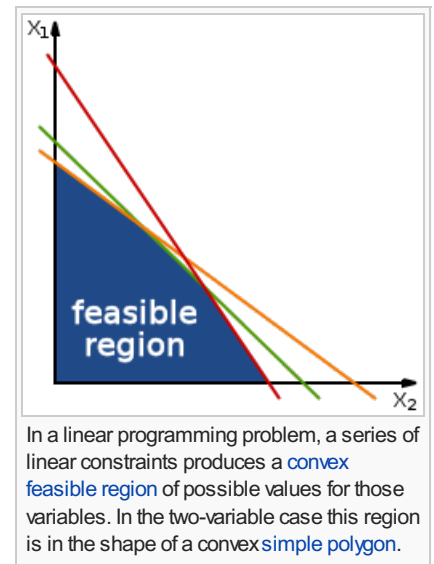
##### Comparison of interior-point methods versus simplex algorithms [\[edit\]](#)

The current opinion is that the efficiency of good implementations of simplex-based methods and interior point methods are similar for routine applications of linear programming.<sup>[15]</sup> However, for specific types of LP problems, it may be that one type of solver is better than another (sometimes much better), and that the structure of the solutions generated by interior point methods versus simplex-based methods are significantly different with the support set of active variables being typically smaller for the later one.<sup>[16]</sup>

LP solvers are in widespread use for optimization of various problems in industry, such as optimization of flow in transportation networks.<sup>[17]</sup>

#### Approximate Algorithms for Covering/Packing LPs [\[edit\]](#)

Covering and packing LPs can be solved approximately in nearly-linear time. That is, if matrix  $A$  is of dimension  $n \times m$  and has  $N$  non-zero entries, then there exist algorithms that run in time  $O(N \cdot (\log N)^{O(1)/\epsilon^{O(1)}})$  and produce





$O(1 \pm \epsilon)$  approximate solutions to given covering and packing LPs. The best known sequential algorithm of this kind runs in time  $O(N + (\log N) \cdot (n+m)/\epsilon^2)$ ,<sup>[18]</sup> and the best known parallel algorithm of this kind runs in  $O((\log N)^2/\epsilon^3)$  iterations, each requiring only a matrix-vector multiplication which is highly parallelizable.<sup>[19]</sup>

## Open problems and recent work [\[edit\]](#)

There are several open problems in the theory of linear programming, the solution of which would represent fundamental breakthroughs in mathematics and potentially major advances in our ability to solve large-scale linear programs.

### List of unsolved problems in computer science

*Does linear programming admit a strongly polynomial-time algorithm?*

- Does LP admit a **strongly polynomial**-time algorithm?
- Does LP admit a strongly polynomial algorithm to find a strictly complementary solution?
- Does LP admit a polynomial algorithm in the real number (unit cost) model of computation?

This closely related set of problems has been cited by [Stephen Smale](#) as among the **18 greatest unsolved problems** of the 21st century. In Smale's words, the third version of the problem "is the main unsolved problem of linear programming theory." While algorithms exist to solve linear programming in weakly polynomial time, such as the [ellipsoid methods](#) and [interior-point techniques](#), no algorithms have yet been found that allow strongly polynomial-time performance in the number of constraints and the number of variables. The development of such algorithms would be of great theoretical interest, and perhaps allow practical gains in solving large LPs as well.

Although the [Hirsch conjecture](#) was recently disproved for higher dimensions, it still leaves the following questions open.

- Are there pivot rules which lead to polynomial-time Simplex variants?
- Do all polytopal graphs have polynomially bounded diameter?

These questions relate to the performance analysis and development of Simplex-like methods. The immense efficiency of the Simplex algorithm in practice despite its exponential-time theoretical performance hints that there may be variations of Simplex that run in polynomial or even strongly polynomial time. It would be of great practical and theoretical significance to know whether any such variants exist, particularly as an approach to deciding if LP can be solved in strongly polynomial time.

The Simplex algorithm and its variants fall in the family of edge-following algorithms, so named because they solve linear programming problems by moving from vertex to vertex along edges of a polytope. This means that their theoretical performance is limited by the maximum number of edges between any two vertices on the LP polytope. As a result, we are interested in knowing the maximum **graph-theoretical diameter** of polytopal **graphs**. It has been proved that all polytopes have subexponential diameter. The recent disproof of the Hirsch conjecture is the first step to prove whether any polytope has superpolynomial diameter. If any such polytopes exist, then no edge-following variant can run in polynomial time. Questions about polytope diameter are of independent mathematical interest.

Simplex pivot methods preserve primal (or dual) feasibility. On the other hand, criss-cross pivot methods do not preserve (primal or dual) feasibility—they may visit primal feasible, dual feasible or primal-and-dual infeasible bases in any order. Pivot methods of this type have been studied since the 1970s. Essentially, these methods attempt to find the shortest pivot path on the **arrangement polytope** under the linear programming problem. In contrast to polytopal graphs, graphs of arrangement polytopes are known to have small diameter, allowing the possibility of strongly polynomial-time criss-cross pivot algorithm without resolving questions about the diameter of general polytopes.<sup>[10]</sup>

## Integer unknowns [\[edit\]](#)

If all of the unknown variables are required to be integers, then the problem is called an **integer programming** (IP) or **integer linear programming** (ILP) problem. In contrast to linear programming, which can be solved efficiently in the worst case, integer programming problems are in many practical situations (those with bounded variables) **NP-hard**. **0-1 integer programming** or **binary integer programming** (BIP) is the special case of integer programming where variables are required to be 0 or 1 (rather than arbitrary integers). This problem is also classified as NP-hard, and in fact the decision version was one of [Karp's 21 NP-complete problems](#).

If only some of the unknown variables are required to be integers, then the problem is called a **mixed integer programming** (MIP) problem. These are generally also NP-hard because they are even more general than ILP programs.

There are however some important subclasses of IP and MIP problems that are efficiently solvable, most notably problems where the constraint matrix is [totally unimodular](#) and the right-hand sides of the constraints are integers or - more general - where the system has the [total dual integrality](#) (TDI) property.

Advanced algorithms for solving integer linear programs include:

- [cutting-plane method](#)
- [branch and bound](#)
- [branch and cut](#)
- [branch and price](#)
- if the problem has some extra structure, it may be possible to apply [delayed column generation](#).

Such integer-programming algorithms are discussed by Padberg and in Beasley.

## Integral linear programs [\[edit\]](#)

A linear program in real variables is said to be **integral** if it has at least one optimal solution which is integral. Likewise, a polyhedron  $P = \{x \mid Ax \geq 0\}$  is said to be **integral** if for all bounded feasible objective functions  $c$ , the linear program  $\{\max cx \mid x \in P\}$  has an optimum  $x^*$  with integer coordinates. As observed by Edmonds and Giles in 1977, one can equivalently say that the polyhedron  $P$  is integral if for every bounded feasible integral objective function  $c$ , the optimal *value* of the linear program  $\{\max cx \mid x \in P\}$  is an integer.

Integral linear programs are of central importance in the polyhedral aspect of [combinatorial optimization](#) since they provide an alternate characterization of a problem. Specifically, for any problem, the convex hull of the solutions is an integral polyhedron; if this polyhedron has a nice/compact description, then we can efficiently find the optimal feasible solution under any linear objective. Conversely, if we can prove that a [linear programming relaxation](#) is integral, then it is the desired description of the convex hull of feasible (integral) solutions.

Note that terminology is not consistent throughout the literature, so one should be careful to distinguish the following two concepts,

- in an *integer linear program*, described in the previous section, variables are forcibly constrained to be integers, and this problem is NP-hard in general,
- in an *integral linear program*, described in this section, variables are not constrained to be integers but rather one has proven somehow that the continuous problem always has an integral optimal value (assuming  $c$  is integral), and this optimal value may be found efficiently since all polynomial-size linear programs can be solved in polynomial time.

One common way of proving that a polyhedron is integral is to show that it is [totally unimodular](#). There are other general methods including the [integer decomposition property](#) and [total dual integrality](#). Other specific well-known integral LPs include the matching polytope, lattice polyhedra, [submodular](#) flow polyhedra, and the intersection of 2 generalized polymatroids/g-polymatroids --- e.g. see Schrijver 2003.

A bounded integral polyhedron is sometimes called a [convex lattice polytope](#), particularly in two dimensions.

## Solvers and scripting (programming) languages [\[edit\]](#)

**Free open-source [permissive](#) licenses:**

Name	License	Brief info
JOptimizer	<a href="#">Apache License</a>	Java library for convex optimization (open source)
<a href="#">OpenOpt</a>	<a href="#">BSD</a>	Universal cross-platform numerical optimization framework, see its <a href="#">LP</a> <a href="#">page</a> and <a href="#">other problems</a> <a href="#">involved</a>
<a href="#">Coopr</a>	<a href="#">BSD</a>	An open-source modeling language for large-scale linear, mixed integer and nonlinear optimization

**Free open-source [copyleft \(reciprocal\)](#) licenses:**

Name	License	Brief info
<a href="#">Cassowary constraint solver</a>	<a href="#">LGPL</a>	an incremental constraint solving toolkit that efficiently solves systems of linear equalities and inequalities

CLP	CPL	an LP solver from COIN-OR
glpk	GPL	GNU Linear Programming Kit, an LP/MILP solver with a native C <a href="#">API</a> and numerous (15) third-party wrappers for other languages. Specialist support for <a href="#">flow networks</a> . Bundles the <a href="#">AMPL</a> -like <a href="#">GNU MathProg</a> modelling language and translator.
LpSolve	LGPL	lp_solve is a free (see LGPL for the GNU lesser general public license) linear (integer) programming solver based on the revised simplex method and the Branch-and-bound method for the integers. LpSolve has an IDE, a native C <a href="#">API</a> , and many external language interfaces, for <a href="#">JAVA</a> , <a href="#">AMPL</a> , <a href="#">MATLAB</a> , <a href="#">O-Matrix</a> , <a href="#">Sysquake</a> , <a href="#">Scilab</a> , <a href="#">Octave</a> , <a href="#">FreeMat</a> , <a href="#">Euler</a> , <a href="#">Python</a> , <a href="#">Sage</a> , <a href="#">PHP</a> , <a href="#">R</a> and the <a href="#">Microsoft Solver Foundation</a> . It is compatible with <a href="#">Zimpl</a> <a href="#">modelling language</a> .
Qoca	GPL	a library for incrementally solving systems of linear equations with various goal functions
R-Project	GPL	a programming language and software environment for statistical computing and graphics

MINTO (Mixed Integer Optimizer, an [integer programming](#) solver which uses branch and bound algorithm) has publicly available source code<sup>[20]</sup> but is not open source.

#### Proprietary:

Name	Brief info
AIMMS	
AMPL	A popular modeling language for large-scale linear, mixed integer and nonlinear optimisation with a free student limited version available (500 variables and 500 constraints).
APMonitor	API to MATLAB and Python. Solve example <a href="#">Linear Programming (LP) problems</a> <a href="#">through MATLAB</a> , <a href="#">Python</a> , or a web-interface.
CPLEX	Popular solver with an API for several programming languages, and also has a modelling language and works with AIMMS, AMPL, <a href="#">GAMS</a> , MPL, OpenOpt, OPL Development Studio, and <a href="#">TOMLAB</a> . Free for academic use.
Excel Solver Function	A nonlinear solver adjusted to spreadsheets in which function evaluations are based on the recalculating cells. Basic version available as a standard add-on for Excel.
FortMP	
GAMS	
Gurobi	Solver with parallel algorithms for large-scale linear programs, quadratic programs and mixed-integer programs. Free for academic use.
IMSL Numerical Libraries	Collections of math and statistical algorithms available in C/C++, Fortran, Java and C#.NET. Optimization routines in the IMSL Libraries include unconstrained, linearly and nonlinearly constrained minimizations, and linear programming algorithms.
LINDO	Solver with an API for large scale optimization of linear, integer, quadratic, conic and general nonlinear programs with stochastic programming extensions. It offers a global optimization procedure for finding guaranteed globally optimal solution to general nonlinear programs with continuous and discrete variables. It also has a statistical sampling API to integrate Monte-Carlo simulations into an optimization framework. It has an algebraic modeling language ( <a href="#">LINGO</a> ) and allows modeling within a spreadsheet ( <a href="#">What'sBest</a> ).
Maple	A general-purpose programming-language for symbolic and numerical computing.
MATLAB	A general-purpose and matrix-oriented programming-language for numerical computing. Linear programming in MATLAB requires the <a href="#">Optimization Toolbox</a> in addition to the base MATLAB product; available routines include INTLINPROG and LINPROG
Mathcad	A WYSIWYG math editor. It has functions for solving both linear and nonlinear optimization problems.
Mathematica	A general-purpose programming-language for mathematics, including symbolic and numerical capabilities.
MOSEK	A solver for large scale optimization with API for several languages (C++, java, .net, Matlab and python).
	A collection of mathematical and statistical routines developed by the <a href="#">Numerical Algorithms Group</a> for multiple programming languages (C, C++, Fortran, Visual Basic, Java and C#) and

<a href="#">NAG Numerical Library</a>	packages (MATLAB, Excel, R, LabVIEW). The Optimization chapter of the NAG Library includes routines for linear programming problems with both sparse and non-sparse linear constraint matrices, together with routines for the optimization of quadratic, nonlinear, sums of squares of linear or nonlinear functions with nonlinear, bounded or no constraints. The NAG Library has routines for both local and global optimization, and for continuous or integer problems.
<a href="#">NMath Stats</a>	A general-purpose <a href="#">.NET</a> statistical library containing a simplex solver. <sup>[21]</sup>
<a href="#">OptimJ</a>	A Java-based modeling language for optimization with a free version available. <sup>[22][23]</sup>
<a href="#">SAS/OR</a>	A suite of solvers for Linear, Integer, Nonlinear, Derivative-Free, Network, Combinatorial and Constraint Optimization; the <a href="#">Algebraic modeling language OPTMODEL</a> <a href="#">↗</a> ; and a variety of vertical solutions aimed at specific problems/markets, all of which are fully integrated with the <a href="#">SAS System</a> .
<a href="#">SCIP</a>	A general-purpose constraint integer programming solver with an emphasis on MIP. Compatible with <a href="#">Zimpl</a> <a href="#">↗</a> modelling language. Free for academic use and available in source code.
<a href="#">XPRESS Solver Engine</a>	Solver for large-scale linear programs, quadratic programs, general nonlinear and mixed-integer programs. Has API for several programming languages, also has a modelling language Mosel and works with AMPL, <a href="#">GAMS</a> . Free for academic use.
<a href="#">VisSim</a>	A visual <a href="#">block diagram</a> language for simulation of <a href="#">dynamical systems</a> .








## See also [\[edit\]](#)

- [Convex programming](#)
- [Dynamic programming](#)
- [Linear-fractional programming \(LFP\)](#)
- [LP-type problem](#)
- [Mathematical programming](#)
- [Job shop scheduling](#)
- [Nonlinear programming](#)
- [Oriented matroid](#)
- [Quadratic programming](#), a superset of linear programming
- [Shadow price](#)
- [Simplex algorithm](#), used to solve LP problems
- [Semidefinite programming](#)

## Notes [\[edit\]](#)

- ↑ **<sup>a</sup>** **<sup>b</sup>** Gerard Sierksma (2001). *Linear and Integer Programming: Theory and Practice, Second Edition*. CRC Press. p. 1. ISBN 978-0-8247-0673-9.
- ↑ **<sup>a</sup>** **<sup>b</sup>** Alexander Schrijver (1998). *Theory of Linear and Integer Programming*. John Wiley & Sons. pp. 221–222. ISBN 978-0-471-98232-6.
- ↑ "Reminiscences about the origins of linear programming" [↗](#) (PDF). *Operations Research Letter* **1** (2): 43–48. April 1982. doi:10.1016/0167-6377(82)90043-8 [↗](#).
- ↑ Vazirani (2001, p. 112)
- ↑ **<sup>a</sup>** **<sup>b</sup>** **<sup>c</sup>** **<sup>d</sup>** Dantzig & Thapa (2003)
- ↑ **<sup>a</sup>** **<sup>b</sup>** **<sup>c</sup>** Padberg (1999)
- ↑ Bland (1977)
- ↑ **<sup>a</sup>** **<sup>b</sup>** Murty (1983)
- ↑ **<sup>a</sup>** **<sup>b</sup>** Papadimitriou (Steiglitz)
- ↑ **<sup>a</sup>** **<sup>b</sup>** **<sup>c</sup>** Fukuda & Terlaky (1997): Fukuda, Komei; Terlaky, Tamás (1997). Thomas M. Liebling and Dominique de Werra, eds. "Criss-cross methods: A fresh view on pivot algorithms". *Mathematical Programming: Series B* (Amsterdam: North-Holland Publishing Co.) **79** (1—3): 369–395. doi:10.1007/BF02614325 [↗](#). MR 1464775 [↗](#).
- ↑ Borgwardt (1987)
- ↑ Todd (2002)
- ↑ Roos (1990): Roos, C. (1990). "An exponential example for Terlaky's pivoting rule for the criss-cross simplex method". *Mathematical Programming. Series A* **46** (1): 79–84. doi:10.1007/BF01585729 [↗](#). MR 1045573 [↗](#).
- ↑ Strang, Gilbert (1 June 1987). "Karmarkar's algorithm and its place in applied mathematics". *The Mathematical Intelligencer* (New York: Springer) **9** (2): 4–10. doi:10.1007/BF03025891 [↗](#). ISSN 0343-6993 [↗](#). MR "883185" [↗](#).
- ↑ **<sup>a</sup>** **<sup>b</sup>** Gondzio & Terlaky (1996)
- ↑ Tibor Illés, Tamás Terlaky, Pivot versus interior point methods: Pros and cons, European Journal of Operational Research, 01/2002
- ↑ For solving network-flow problems in transportation networks, specialized implementations of the simplex

algorithm can dramatically improve its efficiency. Dantzig & Thapa (2003)

18. ^ Christos Koufogiannakis; Neal E. Young (2013). "A Nearly Linear-Time PTAS for Explicit Fractional Packing and Covering Linear Programs". *Algorithmica* **70**: 648–674. [arXiv:0801.1987](#) . doi:10.1007/s00453-013-9771-6 .
19. ^ Zeyuan Allen-Zhu; Lorenzo Orecchia (2015). *Using Optimization to Break the Epsilon Barrier: A Faster and Simpler Width-Independent Algorithm for Solving Positive Linear Programs in Parallel* . SODA.
20. ^ <http://coral.ie.lehigh.edu/~minto/download.html> 
21. ^ Linear programming page at CenterSpace Software 
22. ^ [http://www.in-ter-trans.eu/resources/Zesch\\_Hellingrath\\_2010\\_Integrated+Production-Distribution+Planning.pdf](http://www.in-ter-trans.eu/resources/Zesch_Hellingrath_2010_Integrated+Production-Distribution+Planning.pdf)   
OptimJ used in an optimization model for mixed-model assembly lines, University of Münster
23. ^ <http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/viewFile/1769/2076>  OptimJ used in an Approximate Subgame-Perfect Equilibrium Computation Technique for Repeated Games







## References [\[edit\]](#)

- Kantorovich, L. V. (1940). "Об одном эффективном методе решения некоторых классов экстремальных проблем" [A new method of solving some classes of extremal problems]. *Doklady Akad Sci USSR* **28**: 211–214.
- F. L. Hitchcock: *The distribution of a product from several sources to numerous localities*, Journal of Mathematics and Physics, 20, 1941, 224-230.
- G.B Dantzig: *Maximization of a linear function of variables subject to linear inequalities*, 1947. Published pp. 339–347 in T.C. Koopmans (ed.): *Activity Analysis of Production and Allocation*, New York-London 1951 (Wiley & Chapman-Hall)
- J. E. Beasley, editor. *Advances in Linear and Integer Programming*. Oxford Science, 1996. (Collection of surveys)
- R. G. Bland, New finite pivoting rules for the simplex method, *Math. Oper. Res.* 2 (1977) 103–107.
- Karl-Heinz Borgwardt, *The Simplex Algorithm: A Probabilistic Analysis*, Algorithms and Combinatorics, Volume 1, Springer-Verlag, 1987. (Average behavior on random problems)
- Richard W. Cottle, ed. *The Basic George B. Dantzig*. Stanford Business Books, Stanford University Press, Stanford, California, 2003. (Selected papers by [George B. Dantzig](#))
- George B. Dantzig and Mukund N. Thapa. 1997. *Linear programming 1: Introduction*. Springer-Verlag.
- George B. Dantzig and Mukund N. Thapa. 2003. *Linear Programming 2: Theory and Extensions*. Springer-Verlag. (Comprehensive, covering e.g. [pivoting](#) and interior-point algorithms, large-scale problems, [decomposition following Dantzig-Wolfe](#) and [Benders](#), and introducing [stochastic programming](#).)
- Edmonds, J. and Giles, R., "A min-max relation for submodular functions on graphs," *Ann. Discrete Math.*, v1, pp. 185–204, 1977
- Fukuda, Komei; Terlaky, Tamás (1997). Thomas M. Liebling and Dominique de Werra, eds. "Criss-cross methods: A fresh view on pivot algorithms". *Mathematical Programming: Series B* (Amsterdam: North-Holland Publishing Co.) **79** (1—3): 369–395. doi:[10.1007/BF02614325](#) [↗](#). MR [1464775](#) [↗](#).
- Gondzio, Jacek; Terlaky, Tamás (1996). "3 A computational view of interior point methods". In J. E. Beasley. *Advances in linear and integer programming* [↗](#). Oxford Lecture Series in Mathematics and its Applications **4**. New York: Oxford University Press. pp. 103–144. MR [1438311](#) [↗](#). [Postscript file at website of Gondzio](#) [↗](#) and [at McMaster University website of Terlaky](#) [↗](#).

- **Murty, Katta G.** (1983). *Linear programming*. New York: John Wiley & Sons, Inc. pp. xix+482. [ISBN 0-471-09725-X](#) [MR 720547](#) [↗](#). (comprehensive reference to classical approaches).
- Evar D. Nering and **Albert W. Tucker**, 1993, *Linear Programs and Related Problems*, Academic Press. (elementary)
- M. Padberg, *Linear Optimization and Extensions*, Second Edition, Springer-Verlag, 1999. (carefully written account of primal and dual simplex algorithms and projective algorithms, with an introduction to integer linear programming --- featuring the [traveling salesman problem](#) for *Odysseus*.)
- **Christos H. Papadimitriou** and Kenneth Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Corrected republication with a new preface, Dover. (computer science)
- Michael J. Todd (February 2002). "The many facets of linear programming". *Mathematical Programming* **91** (3): 417–436. [doi:10.1007/s101070100261](#) [↗](#). (Invited survey, from the International Symposium on Mathematical Programming.)
- **Vazirani, Vijay V.** (2001). *Approximation Algorithms*. Springer-Verlag. [ISBN 3-540-65367-8](#). (Computer science)

## Further reading [\[edit\]](#)

A reader may consider beginning with Nering and Tucker, with the first volume of Dantzig and Thapa, or with Williams.

[Library resources](#) about  
**Linear programming**  
[Resources in your library](#) [↗](#)

- Dmitris Alevras and Manfred W. Padberg, *Linear Optimization and Extensions: Problems and Solutions*, Universitext, Springer-Verlag, 2001. (Problems from Padberg with solutions.)
- Mark de Berg, Marc van Kreveld, [Mark Overmars](#), and Otfried Schwarzkopf (2000). *Computational Geometry* (2nd revised ed.). [Springer-Verlag](#). [ISBN 3-540-65620-0](#). Chapter 4: Linear Programming: pp. 63–94. Describes a randomized half-plane intersection algorithm for linear programming.
- **Michael R. Garey** and **David S. Johnson** (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. [ISBN 0-7167-1045-5](#). A6: MP1: INTEGER PROGRAMMING, pg.245. (computer science, complexity theory)
- Bernd Gärtner, [Jiří Matoušek](#) (2006). *Understanding and Using Linear Programming*, Berlin: Springer. [ISBN 3-540-30697-8](#) (elementary introduction for mathematicians and computer scientists)
- Cornelis Roos, Tamás Terlaky, Jean-Philippe Vial, *Interior Point Methods for Linear Optimization*, Second Edition, Springer-Verlag, 2006. (Graduate level)
- Alexander Schrijver (2003). *Combinatorial optimization: polyhedra and efficiency*. Springer.
- Alexander Schrijver, *Theory of Linear and Integer Programming*. John Wiley & sons, 1998, [ISBN 0-471-98232-6](#) (mathematical)
- **Robert J. Vanderbei**, *Linear Programming: Foundations and Extensions* [↗](#), 3rd ed., International Series in Operations Research & Management Science, Vol. 114, Springer Verlag, 2008. [ISBN 978-0-387-74387-5](#). (An on-line second edition was formerly available. Vanderbei's site still contains extensive materials.)
- H. P. Williams, *Model Building in Mathematical Programming*, Third revised Edition, 1990. (Modeling)
- Stephen J. Wright, 1997, *Primal-Dual Interior-Point Methods*, SIAM. (Graduate level)
- **Yinyu Ye**, 1997, *Interior Point Algorithms: Theory and Analysis*, Wiley. (Advanced graduate-level)
- **Ziegler, Günter M.**, Chapters 1–3 and 6–7 in *Lectures on Polytopes*, Springer-Verlag, New York, 1994. (Geometry)

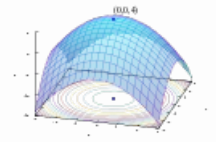
## External links [\[edit\]](#)



This article's **use of external links** may not follow **Wikipedia's policies or guidelines**. Please [improve this article](#) by removing [excessive](#) or [inappropriate](#) external links, and converting useful links where appropriate into [footnote references](#). *(August 2010)*

- [Guidance On Formulating LP Problems](#) [↗](#)
- [Mathematical Programming Glossary](#) [↗](#)
- [The Linear Programming FAQ](#) [↗](#)
- [Benchmarks For Optimisation Software](#) [↗](#)
- [2013 Linear Programming Software Survey](#) [↗](#) - *OR/MS Today*
- [George Dantzig](#) [↗](#)
- [Linear Programming \(LP\) and Operations Research \(OR\) resources for students](#) [↗](#)

v · t · e	<b>Optimization: Algorithms, methods, and heuristics</b>	[hide]
	<b>Unconstrained nonlinear: Methods calling ...</b>	[show]
	<b>Constrained nonlinear</b>	[show]
	<b>Convex optimization</b>	[show]
	<b>Combinatorial</b>	[show]
	<b>Metaheuristics</b>	[show]
	<b>Categories</b> (Algorithms and methods · Heuristics) · <b>Software</b>	
v · t · e	<b>Complementarity problems and algorithms</b>	[show]
<b>Authority control</b>	NDL: 00570683 <span><span></span></span>	



Categories: Linear programming | Convex optimization | Operations research | Geometric algorithms  
P-complete problems | Mathematical and quantitative methods (economics)

This page was last modified on 17 August 2015, at 15:48.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Mobile view

