B

Linear and Integer Programming

Linear programs and integer programs are optimization problems with linear objective functions and linear constraints. These problems are very general: linear programs can be used to express a wide range of problems such as bipartite matching and network flow, while integer programs can express an even larger range of problems, including all of those in NP. We define each of these formalisms in turn.

B.1 Linear programs

Defining linear programs

linear program

A *linear program* is defined by:

- a set of real-valued variables;
- a linear objective function (i.e., a weighted sum of the variables); and
- a set of linear constraints (i.e., the requirement that a weighted sum of the variables must be less than or equal to some constant).

Let the set of variables be $\{x_1, \ldots, x_n\}$, with each $x_i \in \mathbb{R}$. The objective function of a linear program, given a set of constants $w_1, \ldots, w_n$, is

$$\text{maximize} \sum_{i=1}^{n} w_i x_i .$$

Linear programs can also express minimization problems: these are just maximization problems with all weights in the objective function negated.

Constraints express the requirement that a weighted sum of the variables must be greater than or equal to some constant. Specifically, given a set of constants $a_{1j}, \ldots, a_{nj}$ and a constant b_j, a constraint is an expression

$$\sum_{i=1}^{n} a_{ij} x_i \leq b_j .$$

This form actually allows us to express a broader range of constraints than might immediately be apparent. By negating all constants, we can express greater-than-or-equal constraints. By providing both less-than-or-equal and greater-than-or-equal constraints with the same constants, we can express equality constraints.

By setting some constants to zero, we can express constraints that do not involve all of the variables. Furthermore, even problems with piecewise-linear constraints (e.g., involving functions like a max of linear terms) can sometimes be expressed as linear programs by adding both new constraints and new variables. Observe that we *cannot* always write strict inequality constraints, though sometimes such constraints can be enforced through changes to the objective function. (For example, see the linear program given in Equations (4.42)–(4.44) on p. 108, which enforces the strict inequality constraints given in Equations (4.39)–(4.41).)

Bringing it all together, if we have m different constraints, we can write a linear program as follows.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} w_i x_i & \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij} x_i \leq b_j & \forall j = 1 \ldots m \\ & x_i \geq 0 & \forall i = 1 \ldots n \end{aligned}$$

Observe that the requirement that each x_i must be nonnegative is not restrictive: problems involving negative variables can always be reformulated into equivalent problems that satisfy the constraint.

A linear program can also be written in matrix form. Let $\mathbf{w}$ be an $n \times 1$ vector containing the weights w_i, let $\mathbf{x}$ be an $n \times 1$ vector containing the variables x_i, let $\mathbf{A}$ be an $m \times n$ matrix of constants a_{ij}, and let $\mathbf{b}$ be an $m \times 1$ vector of constants b_j. We can then write a linear program in matrix form as follows.

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

In some cases we care to satisfy a given set of constraints, but do not have an associated objective function; any solution will do. In this case the LP reduces to a constraint satisfaction or *feasibility* problem, but we will sometimes still refer to one as an LP with an empty objective function (or, equivalently, the trivial one).

primal problem

dual problem

Finally, every linear program (a so-called *primal problem*) has a corresponding *dual problem* which shares the same optimal solution. For the linear program given earlier, the dual program is as follows.

$$\begin{aligned} \text{minimize} \quad & \mathbf{b}^T \mathbf{y} \\ \text{subject to} \quad & \mathbf{A}^T \mathbf{y} \geq \mathbf{w} \\ & \mathbf{y} \geq \mathbf{0} \end{aligned}$$

In this linear program our variables are $\mathbf{y}$. Variables and constraints effectively trade places: there is one variable $y \in \mathbf{y}$ in the dual problem for every constraint from the primal problem and one constraint in the dual problem for every variable $x \in \mathbf{x}$ from the primal problem.

Solving linear programs

In order to solve linear programs, it is useful to observe that the set of feasible solutions to a linear program corresponds to a convex polyhedron in n-dimensional space. This is true because all of the constraints are linear: they correspond to hyperplanes in this space, and so the set of feasible solutions is the region bounded by all of the hyperplanes. The fact that the objective function is also linear allows us to conclude two useful things: any local optimum in the feasible region will be a global optimum, and at least one optimal solution will exist at a vertex of the polyhedron. (More than one optimal solution may exist if an edge or even a whole face of the polyhedron is a local maximum.)

simplex algorithm

The most popular algorithm for solving linear programs is the *simplex algorithm*. This algorithm works by identifying one vertex of the polyhedron and then taking uphill (i.e., objective-function-improving) steps to neighboring vertices until an optimum is found. This algorithm requires an exponential number of steps in the worst case, but is usually very efficient in practice.

interior-point method

Although the simplex algorithm is not polynomial, it can be shown that other algorithms called *interior-point methods* solve linear programs in worst-case polynomial time. These algorithms get their name from the fact that they move through the interior region of the polyhedron rather than jump from one vertex to another. Surprisingly, although these algorithms dominate the simplex method in the worst case, they can be much slower in practice.

B.2 Integer programs

Defining integer programs

integer program

Integer programs are linear programs in which one additional constraint holds: the variables are required to take integral (rather than real) values. This makes it possible to express combinatorial optimization problems such as satisfiability or set packing as integer programs. A useful subclass of integer programs are 0–1 *integer programs*, in which each variable is constrained to take either the value 0 or the value 1. These programs are sufficient to express any problem in NP. The form of a 0–1 integer program is as follows.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} w_i x_i & \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij} x_i \leq b_j & \forall j = 1 \ldots m \\ & x_i \in \{0, 1\} & \forall i = 1 \ldots n \end{aligned}$$

mixed-integer program

Another useful class of integer programs is *mixed-integer programs*, which involve a combination of integer and real-valued variables.

Finally, as in the LP case, both integer and mixed-integer programs can come without an associated objective function, in which case they reduce to constraint-satisfaction problems.

Solving integer programs

The introduction of an integrality constraint to linear programs leads to a much harder computational problem: in the worst case integer programs are undecidable. When variables' domains are finite sets of integers, they are NP-hard. Thus, it should not be surprising that there is no efficient procedure for solving integer programs.

branch-and-bound search

The most commonly used technique is *branch-and-bound search*. The space of variable assignments is explored depth-first: first one variable is assigned a value, then the next, and so on; when a constraint is violated or a complete variable assignment is achieved, the search backtracks and tries other assignments. The best feasible solution found so far is recorded as a lower bound on the value of the optimal solution. At each search node the *linear program relaxation* of the integer program is solved: this is the linear program where the remaining variables are allowed to take real rather than integral values between the minimum and maximum values in their domains. It is easy to see that the value of a linear program relaxation of an integer program is an upper bound on the value of that integer program, since it involves a loosening of the latter problem's constraints. Branch-and-bound search differs from standard depth-first search because it sometimes prunes the tree. Specifically, branch-and-bound backtracks whenever the upper bound at a search node is less than or equal to the lower bound. In this way it can skip over large parts of the search tree while still guaranteeing that it will find the optimal solution.

linear program relaxation

Other, more complex techniques for solving integer programs include branch-and-cut and branch-and-price search. These methods offer no advantage over branch-and-bound search in the worst case, but often outperform it in practice.

Although they are computationally intractable in the worst case, sometimes integer programs are provably easy. This occurs when it can be shown that the solution to the linear programming relaxation is integral, meaning that the integer program can be solved in polynomial time. One important example is when the constraint matrix is *totally unimodular* and the vector $\mathbf{b}$ is integral. A unimodular matrix is a square matrix whose determinant is either -1 or 1; a totally unimodular matrix is one for which every square submatrix is unimodular. This definition implies that the entries in a totally unimodular matrix can only be -1, 0, and 1.

total unimodularity