11

The simplex method

Chapters 11–15 treat the algorithmic side of polyhedra and linear programming. In the present chapter we discuss the most prominent algorithm, the *simplex method*.

The simplex method was designed by Dantzig [1951a], and is, at the moment, *the* method for linear programming. Although some artificial examples show exponential running time, in practice and on the average the method is very efficient.

We do not aim at teaching the simplex method here—for this we refer to the text-books on linear programming mentioned at the end of Part III.

In Section 11.1 we describe the simplex method, and show that it terminates. In Section 11.2 we discuss how the method can be performed in practice, with simplex tableaux and the pivoting operation. In Section 11.3 we make some remarks on pivot selection and cycling, and on the theoretical and practical complexity. Next, in Sections 11.4 and 11.5 we discuss the worst-case and the average running time of the simplex method. Finally, in Sections 11.6 and 11.7 we give two variants of the simplex method: the *revised simplex method* and the *dual simplex method*.

11.1. THE SIMPLEX METHOD

The idea of the simplex method is to make a trip on the polyhedron underlying a linear program, from vertex to vertex along edges, until an optimal vertex is reached. This idea is due to Fourier [1826b], and was mechanized algebraically by Dantzig [1951a]. We describe the simplex method, including 'Bland's pivoting rule' [1977a].

The simplex method if a vertex is known. First suppose that we wish to solve

(1) $\max\{cx \mid Ax \leqslant b\}$

and that we know a vertex x_0 of the feasible region $P := \{x \mid Ax \leqslant b\}$, which we assume to be pointed. We assume the inequalities in $Ax \leqslant b$ to be ordered:

(2) $a_1x \leqslant \beta_1, \ldots, a_mx \leqslant \beta_m.$

Choose a subsystem $A_0x \leqslant b_0$ of $Ax \leqslant b$ such that $A_0x_0 = b_0$ and A_0 is nonsingular. Determine u so that $c = uA$ and u is 0 at components outside A_0 (so cA_0^{-1} is calculated, and 0's are added).

Case 1. $u \geqslant 0$. Then x_0 is optimal, since

(3) $$cx_0 = uAx_0 = ub \geqslant \min\{yb \mid y \geqslant 0; yA = c\} = \max\{cx \mid Ax \leqslant b\}.$$

So at the same time, u is an optimal solution for the dual problem of (1).

Case 2. $u \ngeqslant 0$. Choose the smallest index i^* for which u has negative component υ_{i^*}. Let y be the vector with $ay = 0$ for each row a of A_0 if $a \neq a_{i^*}$, and $a_{i^*}y = -1$ (i.e. y is the appropriate column of $-A_0^{-1}$). [Note that, for $\lambda \geqslant 0$, $x_0 + \lambda y$ traverses an edge or ray of P, or is outside P for all $\lambda > 0$. Moreover,

(4) $$cy = uAy = -\upsilon_{i^*} > 0.]$$

Case 2 splits into two cases:

Case 2a. $ay \leqslant 0$ for each row a of A. Then $x_0 + \lambda y$ is in P for all $\lambda \geqslant 0$, and hence the maximum (1) is unbounded (using (4)).

Case 2b. $ay > 0$ for some row a of A. Let λ_0 be the largest λ such that $x_0 + \lambda y$ belongs to P, i.e.

(5) $$\lambda_0 := \min\left\{\frac{\beta_j - a_j x_0}{a_j y} \,\middle|\, j = 1, \ldots, m; a_j y > 0\right\}.$$

Let j^* be the smallest index attaining this minimum. Let A_1 arise from A_0 by replacing row a_{i^*} by a_{j^*}, and let $x_1 := x_0 + \lambda_0 y$. So $A_1 x_1 = b_1$, where b_1 is the part of b corresponding to A_1. Start the process anew with A_0, x_0 replaced by A_1, x_1.

Repeating this we find $A_0, x_0; A_1, x_1; A_2, x_2; \ldots$

Theorem 11.1. *The above method terminates.*

Proof. Denote by $A_k x \leqslant b_k$, x_k, u_k, y_k the subsystem of $Ax \leqslant b$, the vertex, the vectors u, y as they are in the kth iteration. From (4) we know

(6) $$cx_0 \leqslant cx_1 \leqslant cx_2 \leqslant \ldots$$

where $cx_k = cx_{k+1}$ only if $x_{k+1} = x_k$ (as if $x_{k+1} \neq x_k$, then $\lambda_0 > 0$, and hence $cx_{k+1} > cx_k$).

Suppose the method does not terminate. Then there exist k, l such that $k < l$ and $A_k = A_l$ (since there are only finitely many choices for A_k). Hence $x_k = x_l$, and therefore $x_k = x_{k+1} = \cdots = x_l$. Let r be the highest index for which a_r has been removed from A_t in one of the iterations $t = k, k+1, \ldots, l$, say in iteration p. As $A_k = A_l$, we know that a_r also has been added to A_q in some iteration q with $k \leqslant q < l$. It follows that

(7) $$\text{for } j > r\text{: } a_j \text{ occurs in } A_p \Leftrightarrow a_j \text{ occurs in } A_q.$$

By (4), $u_p A y_q = cy_q > 0$. So $\upsilon_{pj}(a_j y_q) > 0$ for at least one j (denoting $u_p = (\upsilon_{p1}, \ldots, \upsilon_{pm})$). However: