THE (DANTZIG) SIMPLEX METHOD FOR LINEAR PROGRAMMING

George Dantzig created a simplex algorithm to solve linear programs for planning and decision-making in large-scale enterprises. The algorithm's success led to a vast array of specializations and generalizations that have dominated practical operations research for half a century.

George Dantzig, in describing the "Origins of the Simplex Method,"[1] noted that it was the availability of early digital computers that supported and invited the development of LP models to solve real-world problems. He agreed that invention is sometimes the mother of necessity. Moreover, he commented that he initially rejected the simplex method because it seemed intuitively more attractive to pursue the objective function downhill—as in the currently popular interior-point methods—rather than search along the constraint set's edges. It is this latter approach that the simplex method uses, which should not be confused with the J.A. Nelder and R. Mead's function-minimization method,[2] also associated with the word simplex.

1521-9615/00/$10.00 © 2000 IEEE

JOHN C. NASH
University of Ottawa

When Dantzig introduced his method in 1947, it was somewhat easier to sort out the details of the simplex method than to deal with the "where are we in the domain space" questions that are, in my opinion, the core of interior-point approaches. Easier does not mean simpler, however.

Chapter and verse

The LP problem is, in one of the simplex method's many forms,

$$\text{Minimize (with respect to } \boldsymbol{x}) \ \boldsymbol{c}' \boldsymbol{x} \qquad (1a)$$
$$\text{subject to} \quad \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \qquad (1b)$$
$$\text{and} \quad \boldsymbol{x} \geq 0 \qquad (1c)$$

This is not always the problem we want to solve, though—we might want to have the matrix $\mathbf{A}$ and vector $\boldsymbol{b}$ partitioned row-wise so that

$$\mathbf{A} = \begin{matrix}(\mathbf{A}1)\\(\mathbf{A}2)\\(\mathbf{A}3)\end{matrix} \qquad \boldsymbol{b} = \begin{matrix}(\boldsymbol{b}1)\\(\boldsymbol{b}2)\\(\boldsymbol{b}3)\end{matrix}$$

so that we can write an LP problem as

$$\text{Minimize (with respect to } \boldsymbol{x}) \quad \boldsymbol{c}'\boldsymbol{x} \tag{2a}$$

$$\text{subject to} \quad \mathbf{A}1\,\boldsymbol{x} \le \boldsymbol{b}1 \tag{2b}$$

$$\mathbf{A}2\,\boldsymbol{x} = \boldsymbol{b}2 \tag{2c}$$

$$\mathbf{A}3\,\boldsymbol{x} \ge \boldsymbol{b}3 \tag{2d}$$

$$\boldsymbol{x} \ge 0 \tag{2e}$$

However, the use of slack and surplus variables (the $\boldsymbol{x}$ variables we use to augment our problem) can transform the problems from one to the other. In fact, we must add slacks to the left-hand side of Equation 2b and subtract them from Equation 2d to arrive at equalities. Moreover, the special non-negativity conditions in Equations 1c and 2e could be subsumed into the matrix equation or inequation structure, but the LP tradition calls for listing them separately. The number of rows in **A** apart from the non-negativity constraints we call m, and the number of variables (including the slacks and surpluses) we call n. Typically, $m \le n$.

The simplex method assumes we have an initial basic feasible solution $\boldsymbol{x}_{\text{init}}$, that is, a solution that is feasible and that has just m of the $\boldsymbol{x}$'s nonzero. The $m\boldsymbol{x}$ values multiply m columns of **A**, which we call the basis. "Basic" definitely has a technical rather than a general meaning here. We assume this set of basis vectors (columns of **A**) is linearly independent and that it has full rank. Therefore, the core of the simplex method is to exchange one of the columns of **A** that is in the basis for one that is not. This corresponds to increasing one of the nonbasic variables while keeping the constraints satisfied, a process that reduces some of the $\boldsymbol{x}$'s. We choose to increase the $\boldsymbol{x}$ that gives us the most decrease in the objective function, and we continue to increase it until one of the current basic variables goes to zero, which means we now have a new basis. Moreover, the new solution turns out to be a new vertex of the simplex that the constraints define, and it is also a neighboring vertex to that described by our starting basic feasible solution. We have exchanged one vertex of the simplex for one of its neighbors and done so in a way that moves the objective function toward the minimum.

This exchange, or pivoting, seems to be a topic especially devised to torture business school undergraduates. Nevertheless, it is theoretically and practically a relatively simple process, unless, of course, you have a computer without enough memory to hold everything all at once. Additionally, there are some nasty details we have to sort out:

- How do we get an initial feasible solution? Or, more importantly, how do we tell if there is no feasible solution?
- What happens if the constraints don't stop the objective from being reduced? That is, what if the solution is unbounded?
- What happens if several vertices give equal objective values?

These questions, and the memory-management issues, occupied many researchers and generated many journal and book pages over the years. For example, more than half the pages in S. Gass's classic text are devoted to such details;[3] the applications come so late in the book that students of slow lecturers must have wondered what all the fuss was about.

Researchers handled the questions mentioned earlier in a variety of ways. Clever management and programming tactics overcame the memory issue. In early years, storage was so slow that programmers spent considerable effort to arrange computations that would complete in time to read and write data as disks and drums presented the appropriate tracks and sectors. Such complications help to obscure the central ideas of the algorithms in a muddle of detail, mostly irrelevant now.

The initial feasible solution issue was addressed, perhaps surprisingly, by adding m more variables, called artificial variables, which form the initial basis, then minimizing an objective function that drives them all out of the basis.[4] If we cannot drive them out, we have an infeasible problem.

Unbounded solutions turn out to be rather simple to detect, but the "equal objective function" or degeneracy issue worried people a great deal, because they implied the possibility that the algorithm might not terminate. In practice, perturbation or rule-based methods can avoid the cycling.

Nice extras

Modern textbooks spend much less ink on the details.[5] Similarly, recent discussions[6] of the computational techniques have abandoned the tableaux and the jargon of earlier work[3,4] regarding the matrix notations of numerical linear algebra. What remains of great interest, both mathematically and computationally, is the duality feature of LP problems. That is, an LP written as

$$\text{(Primal) minimize } \boldsymbol{c}'\,\boldsymbol{x} \text{ such that } \mathbf{A}\,\boldsymbol{x} \ge \boldsymbol{b},\ \boldsymbol{x} \ge 0$$

has a dual equivalent

(Dual) maximize $\boldsymbol{b}'\boldsymbol{y}$ such that $\mathbf{A}'\boldsymbol{y} \leq \boldsymbol{c}, \boldsymbol{y} \geq 0$.

Moreover, the optima, if they exist, have the same objective value. Duality also leads to useful interpretations of solutions in terms of prices or values when the LP problems under consideration have economic or related contexts.

Practical importance

The simplex method's importance really lies in the value of the LP applications, even when the LP model is only a crude approximation to the real world. In the 1940s, many organizations were very hungry for solutions to LP problems, even if they did not realize what LP was. Oil and chemical companies led the way, especially for optimizing product mix from multiple sources or multiple sites. Transportation companies and the military recognized quite early that transportation and logistics problems could be formulated as LPs. Large-scale agricultural economic problems were also early applications. Examples also became realized in production planning, staff and resource scheduling, and network or traffic flows. LP can even be applied to maintaining the confidentiality of government statistics.[7]

For example, the Diet Problem has as objective function elements ($\boldsymbol{c}$) the costs per unit of food ingredients. The right-hand side values ($\boldsymbol{b}$) are the required minimum amounts of each of a list of m nutrients, and the constraint coefficients $\mathbf{A}$ give the amount of each nutrient in a unit of each ingredient. Thus we want to have $\mathbf{A}\,\boldsymbol{x} \geq \boldsymbol{b}$ to satisfy the nutrient requirements (we could also put in upper limits of nutrients like vitamin A that can be toxic), but we also want the cheapest blend. Of course, the resulting solution might not be very tasty. It is simply nutritious, and by construction will be the cheapest such recipe. Perhaps this is the origin of institutional cafeteria food.

Although the mathematics and computational algorithms might be fascinating, my view is that the value of the applications is the first reason for the importance of the simplex method. Promotion to the "top 10" category still needs something else, and this is the particular efficiency of the simplex method in finding the best vertex of the simplex. Most constraint matrices in practical LP problems are quite sparse, and the number of iterations where we move from one vertex to another is generally very small relative to the number of variables $\boldsymbol{x}$. Even better, we can explore the sensitivity of the optimal solution to small changes in the constraint and objective very easily. This is not always the case for some other methods.

Did Dantzig realize this efficiency as he first coded the simplex method? My guess is that he did not, although it is clear from his reminiscences[1] that such an understanding was not long in coming. In looking back over half a century, I find it remarkable that so many workers could sort through the jungle of awkward, and to some extent unnecessary, details to see the underlying tool's value. CiSE

References

1. S.G. Nash, *A History of Scientific Computing*, ACM Press, New York, 1990, pp. 141–151.
2. J.A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Computer J.*, Vol. 7, 1965, pp. 308–313.
3. S. Gass, *Linear Programming*, 2nd ed., McGraw-Hill, New York, 1964.
4. G. Hadley, *Linear Programming*, Addison-Wesley, Reading, Mass., 1962.
5. S.G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, New York, 1996.
6. J.L. Nazareth, *Computer Solution of Linear Programs*, Oxford Science Publications, New York, 1987.
7. G. Sande, "Automated Cell Suppression to Preserve Confidentiality of Business Statistics," *Statistical J. United Nations ECE*, Vol. 2, 1984, pp. 33–41.

John C. Nash is a professor at the University of Ottawa. He is the author or coauthor of three books on computation, and his research and writings cover a wide range of topics in computers, mathematics, forecasting, risk management, and information science. He has been a mathematics columnist for *Interface Age* and the scientific computing editor for *Byte* magazine. He obtained his BSc in chemistry at the University of Calgary and his DPhil in mathematics from Oxford. Contact him at the Faculty of Administration, Univ. of Ottawa, 136 J-J Lussier Private, Ottawa, Ontario, K1N 6N5, Canada; jcnash@uottawa.ca; macnash.admin.uottawa.ca.