ORC 65-9
MARCH 1965

LARGE-SCALE SYSTEM OPTIMIZATION: A REVIEW

by

George B. Dantzig

OPERATIONS RESEARCH CENTER

COLLEGE OF ENGINEERING

UNIVERSITY OF CALIFORNIA - BERKELEY

LARGE-SCALE SYSTEM OPTIMIZATION: A REVIEW

by

George B. Dantzig
Operations Research Center
University of California, Berkeley

March 1965 ORC 65-9

This research has been partially supported by the Office of Naval Research under Contracts Nonr-222(83) and Nonr-3656(02) with the University of California.

Large-Scale System Optimization: A Review

by

George B. Dantzig

Mathematical programming is a generic term for the related fields of Linear Programming, Network Flow Theory, Integer Programming, Convex and Non-Linear Programming, and Programming under Uncertainty. Its research has problems, particularly those problems where random events and decision events occur alternately in successive stages. In problems where such uncertainty occurs, what is usually done in formulating is to replace the uncertain elements with their expected values (with possible an added safety factor). It is well known that a plan based on expected values of its coefficients and constraints can lead to answers that are not correct. Although the use of expected value does not lead to the best answer, it is entirely possible that it could lead to excellent plans indistinguishable from the optimum in the run-of-the-mill application. When one considers instead, a direct attack on uncertainty via mathematical programming, it inevitably leads to the consideration of large-scale systems. These, because of their structure, have proven difficult of solution so far, but, I believe, of intensive investigation in the future.

Mathematical programming is a term invented by Robert Dorfman of Harvard around 1950. He felt that at that time, the fundamentals of linear programming were well enough known that the wave-of-the-future lay in the extension of the methods of linear programming into the non-linear programming field. Certainly we today, 15 years later, feel this is true. In the Calculus, the derivative (or first order approximation) plays a key role. Applied to non-linear inequality systems, it leads to approximation by linear inequality systems. This is one way which these extensions have taken place, and illustrates why the various fields

comprised under mathematical programming are related. Here are some other ways:

One attempts to extend the concept of duality to non-linear systems. Having done so, one tries to combine the combinatorial power of linear programs with the classical steepest descent processes to solve non-linear programs.

One attempts, as we have just noted, to reduce problems involving uncertainty to equivalent deterministic systems and to large-scale systems with special structure.

One tries to solve an integer program by replacing it with an equivalent linear program; that is to say, by cleverly building up a set of linear inequalities that are both necessary and sufficient.

In all of these developments, one characteristic stands out; namely in one way or another, techniques for solving large-scale systems play a dominant role.

Accordingly, let us look first at direct methods for handling large problems. Around 1954 or so, under the auspices of The RAND Corporation, William Orchard-Hays produced the first truly commercial linear programming code. It had many features that helped amateurs to get their problem on the machine with a reasonable chance of getting an answer. Today, the building of a linear programming code (complete with all the special features) is a major undertaking which is expensive to produce and to maintain.

As applications grow, there has been an increasing demand to handle truly enormous systems. The Russian, Kantorovich, in his 1939 pamphlet, envisioned such a possibility. Already, linear programming models of industrial systems have been solved with more than 10^6 variables and 10^4 equations. These models, of course, do not have general matrix structure and it is not likely that any instance of a large practical problem will ever have general structure. The reason is obvious. Just imagine the physical task alone of finding all the coefficients for a thousand by ten thousand general linear programs (there could be a high as 10^7 non-

zero coefficients).

Fortunately, large-scale practical models tend to have a low percentage of non-zero coefficients; in fact, under 5%, sometimes under 1%. Orchard-Hays' first code exploited this characteristic by making use of a "pricing vector". This made inexpensive the selection of the pivot column directly from the original non-dense data. The pivot column here refers to one of the steps in the simplex method for solving linear programs.

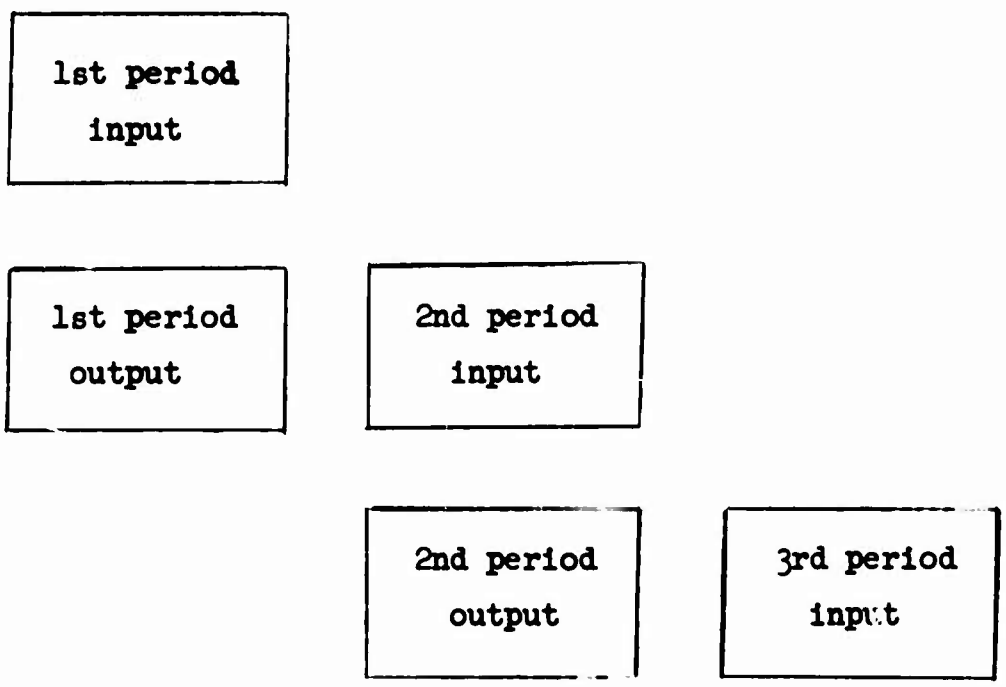
As systems have grown in size, every advantage has also been taken of the characteristics of the improved computers. It has been discovered recently that the size of the inverse representation of the basis in the simplex method could have an important effect on running time. Therefore, compact-inverse schemes along the lines first proposed by Harry Markowitz of RAND have become increasingly important. Recently, two groups working independently, developed this approach with astounding results. For example, the Standard Oil Company of California group reports running-time on some of their typical large problems cut to 1/4.

How to find the most compact inverse representation of a sparse matrix is still an unsolved problem:

Conjecture: If a non-singular matrix has K non-zero elements, it is always possible to represent them as a product of elementary matrices such that the total number of non-zero entries (excluding their diagonal unit elements) is at most K. [Incidentally, the impirical schemes just mentioned often have no more than K + 10%K non-zeros in the inverse representation.]

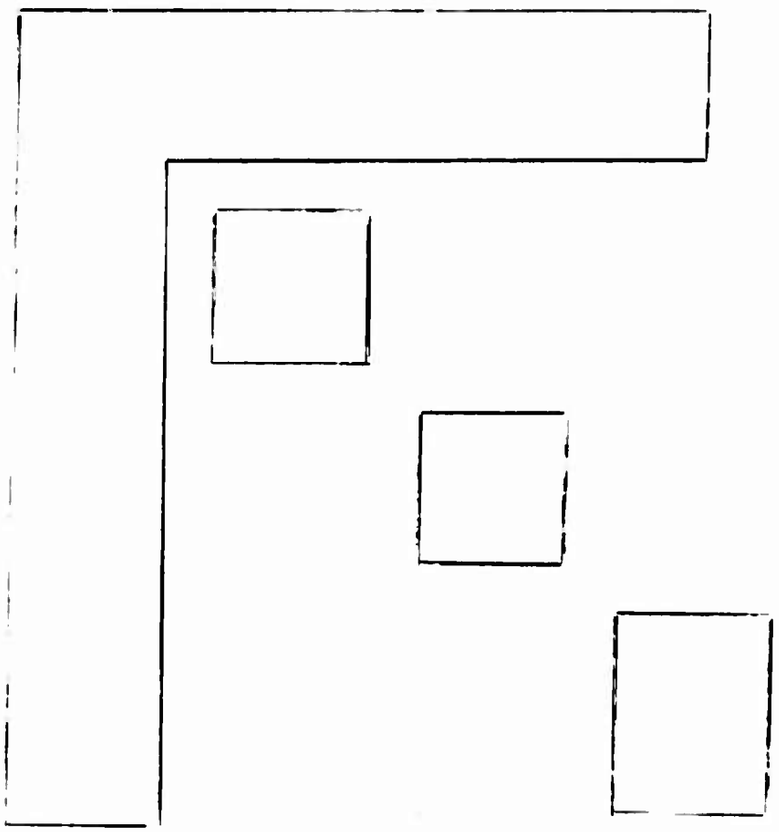
Dynamic structures are interesting in themselves, and could have important applications. One such is the linear control processes proposed by Pontryagin.

(I will speak of his problem later in connection with the decomposition principle.) As early as 1954, I published a paper on how to compact the inverse representation of the basis with a staircase structure (see figure). Again in 1962, I discussed another method which permitted one to find a compact inverse and then efficiently maintain this compactness in moving from one iteration to the next. There have been several other proposals, all excellent, that seek to apply the simplex method to the full system by compacting the inverse. As far as I know, none of these direct proposals have been realized in computer codes.



Large-scale systems have been attacked indirectly by means of the decomposition principle. Several codes have been written, and some of the recent experiences have been encouraging. J. F. Benders in his thesis "Partitioning in Mathematical Programming 1960", developed the dual of the decomposition principle, and shows how this approach can be used to deal with the mixed-integer programming problem. Rosen and Beale have each proposed partitioning methods

for solving systems whose structures fit into the framework of a common horizontal and/or vertical border while the remainder is a diagonal set of independent blocks.



Purely combinatorial problems form an important division of mathematical programming. They fall briefly into two categories. The first are those problems whose structures are special -- like the transportation problem -- or the minimum number of arcs which "cover" nodes in a network (graph). For these, special methods have been sought.

One of the most tantalizing problems of this type has been the travelling salesman problem. It is so close to a network-flow type problem that one would hope to find some easy representation of the faces of its polyhedral solution set. So far, none has been discovered. There is also a close relation between

covering problems and the famous four-color problem. The other approach to combinatorial problems is through integer programming. This was first used in 1954 to solve a particular large-scale travelling salesman problem by Fulkerson, Johnson, and myself. In 1958, Gomory laid the foundations of this field by showing how to systematically determine a necessary and sufficient system of linear inequalities.

The inter-relation between large-scale system methods and integer programming was brought out in a recent paper of Gomory entitled "Large and Non-convex Problems in Linear Programming". Here, Gomory reviews in a unified manner, how the ideas of integer programming and those of the decomposition principle can be combined to solve many important applications such as the paper-trim problem, multi-commodity flows in networks, programming of economic lot sizes, etc.

Integer programming methods are being experimented with in a number of places. It seems likely that we are nearing a threshold, and that we will soon see some excellent commercial codes produced and used successfully for certain problems.

I will not, in this presentation, describe the developments in non-linear programming. Rather, I have chosen to illustrate the power of certain non-linear programming ideas, such as the generalized linear program of Wolfe to an interesting problem in linear control theory.

But first, I would like to review the concept of a Generalized Program. This differs from an ordinary linear program. Instead of coefficients in each column being known, the column P_j may be freely drawn from a convex set, C_j.

PROBLEM: Find Min X_0 , $\{X_j \geq 0,\ P_j \epsilon C_j\}$, $P_0 \epsilon C_0 Q \epsilon C_q$ such that

$$P_0X_0 + -- + P_nX_n = Q \quad .$$

As an example, consider the CONVEX PROGRAM: Find $(x_1, x_2, \ldots, x_n) \epsilon$ C compact and convex such that

$$\phi_1(x) \leq 0$$

$$\phi_2(x) \leq 0$$

$$\cdot$$

$$\cdot$$

$$\cdot$$

$$\phi_m(x) \leq 0$$

$$\phi_0(x) = Z(\text{Min})$$

where $\phi_i(x)$ are continuous convex functions of $X \epsilon C$. Let us assume an $x = x^0$ is known such that $\phi_i(x^0) < 0$ for $i \neq 0$. It can be shown that the generalized program on the following page is equivalent.

PROBLEM: Find Min Z and λ, $\lambda_i \geq 0$, $\mu_i \geq 0$ such that

						Prices
1		1			= 1	π_0
$\phi_1(x)$	$\lambda +$	$\phi_1(x^0)$	$\lambda_0 + \mu_1$		= 0	π_1
.		.	$+\mu_2$		= 0	π_2
.		.	.		.	.
.		.	.		.	.
$\phi_m(x)$		$\phi_m(x^0)$		$+\mu_m$	= 0	π_m
$\phi_0(x)$		$\phi_0(x^0)$		(-Z)	= 0	1

The position of an initial basic-set of columns is indicated by heavy dots. The associated set of simplex multipliers are denoted by π_i ; initially, $\pi = \pi^0$ where $\pi_1^0 = 0$ for $i \neq 0$ and $\pi_0^0 = -\phi_0(x^0)$. The next step is to form the Lagrangian

$$\phi(X) = \phi_0(x) + \sum_{i=1}^{m} \pi_i \phi_i(x) + \pi_0$$

for $\pi = \pi^0$ and to minimize $\phi(x)$ for $x \epsilon C$. Notice that the Lagrangian is precisely what we get if we were to "price out" the general column of coefficients of the variable λ using the price vector to form the inner product. Thus, we wish to choose x such that this scalar product is minimum.

Let $x = x^0$ be the value of the minimizing x . If $\phi(x^0) = 0$, then x^0 is an optimum solution. If $\phi(x^0) < 0$, an extra column is inserted in the generalized program with coefficients $[1, \phi_1(x^1), \ldots, \phi_m(x^1), \phi_0(x^1)]$ and vari-

able λ_1 . The problem, restricted to those variables whose columns have known coefficients, is then optimized using the simplex method. This gives rise to a new set of simplex multipliers $\pi = \pi'$. This gives rise to a new solution $x = x^2$, etc. At any stage, the approximate solution is $x = \Sigma\lambda_i x^i$ using for λ_i those λ_i which solve the problem restricted to those variables with known coefficients.

Let us turn to a problem in control theory. The application of mathematical programming methods to solve control problems has been studied by Zadeh and Whalen, by Ben rosen, and others. I would like to confine myself, however, to Linear Control Theory as described by Pontryagin, Botlyanski, Gemkrelidge, and Mischenko in Chapter III of their book on this subject.

We consider an "object" defined by its $n + 1$ coordinates $x = (\xi_0 , \xi_1 ,\ldots,\xi_n)$ whose "motion", described as a function of a "time" parameter t , can be written as a linear system of differential equations

$$\frac{dx}{dt} = Ax + Bu \tag{1}$$

where $u = (u_1 , u_2 ,\ldots,u_r)$ is a control vector that must be chosen for each t from a convex compact set $U(t)$. The initial conditions at $t = 0$ are

$$x^0 = (0 , \xi_1^0 , \xi_2^0 ,\ldots,\xi_n^0) , \text{ (fixed)} .$$

The terminal conditions at $t = T$ is obtained by setting

$$x^T = \bar{X}^T + ZE_0 \tag{2}$$

where

$$\bar{x}^T = (0, \xi_1^T, \xi_2^T, \ldots, \xi_n^T), \text{ (fixed)}$$

$$E_0 = (1,0,0,\ldots,0)$$

and by requiring that $u = u(t)$ to be chosen such that

(3) $$Z \text{ is minimum} .$$

As given in Chapter III of the book "Mathematical Theory of Optimal Control Processes" by Pontryagin, Boltyashii, Gemkrelidge, Mischenko, the final state may be written in the form

(4) $$-ZE_0 + \int_0^T P_{T-t} B\, u(t)dt = b$$

where $b = \bar{x}^T - P_T x^0$ is a known vector, and $P_t = e^{tA}$ matrix that may be conveniently computed as a function of t. For example, for the case of real distinct characteristic roots λ_1 of A:

(5) $$P_t = e^{tA} \sum_0^n M_i e^{\lambda_i t}$$

where M_i are square matrices independent of t. The latter formula for the M_i is developed in "An Introduction to the Application of Dynamic Programming to Linear Control Systems" by F. T. Smith in RAND Report RM-3526-PR, February 1963.

We may formally write (4) as a generalized linear program.

PROBLEM: Find Min Z, $\mu \geq 0$ such that

$$-ZE_0 + Y\mu = b \tag{6}$$

$$\mu = 1$$

where Y may be freely chosen from the convex set defined by

$$Y \geq \int_0^T P_{T-t}Bu(t)dt \tag{7}$$

for all possible choices of $u(t) \in U(t)$.

The method for solving the generalized linear program described earlier for convex programming can be applied. For brevity, we omit the question of how to obtain the initializing basic set, except to say it is the analog of the phase I procedure of the ordinary simplex method.

As soon as $\pi = \pi^k$ is determined for iteration k we seek a solution of the sub-problem such that the inner product

$$\text{Min } \pi^k Y = \text{Min } \pi^k \int_0^T P_{T-t}Bu(t)dt$$

$$= \int_0^T [\text{Min}(\pi^k P_{T-t}Bu(t)]dt \quad \text{for } u(t) \in U(t) .$$

It is important to note that for each t , $\pi^k P_{T-t}B$ is some known vector c^t .

Thus, for each t we must solve:

SUB-PROBLEM: Find Min $c^t u$ for $u \in U(t)$.

If U(t) is a polyhedral set, the sub-problem is simply a linear program. If

$U(t)$ is the same for all t, then the linear programs are the same for all t except for the varying objective forms $c^t u$. Interesting enough, the sub-programs turn out to be the same as what Pontryagin obtains using his maximal principle.

A control problem is an example of a dynamic system which, is t is treated by the straightforward procedure of discretizing the time, would lead to a large-scale computational problem. The generalized linear program (or decomposition principle approach), however, provides us with a procedure which does not require the discretizing of the time interval.

The last twenty years have been marked by the accelerated trend toward automation. Many believe that not only simple control processes, but soon the more complex control processess will be mechanized. If so, whether we like it or not, decisions will be made for us by machines. Whether or not they will be good decisions will depend on how cleverly we have instructed the machines. This in turn will depend heavily on how clever we have been in developing solution techniques for solving large-scale systems.

To this end, we have sketched several ideas: (1) taking advantage of the low density of the non-zero coefficients in the original matrix, (2) finding a compact inverse representation of the basis using the simplex method, and (3) making use of the generalized linear program or decomposition principle approach. We illustrated the latter on a linear-control problem and found that it led to the maximal principle with the added bonus, however, that it can be used to constructively converge to an optimal solution.