Annals of Operations Research 5(1985/6)395–411

INTEGRATING MODELING, ALGORITHM DESIGN, AND COMPUTATIONAL IMPLEMENTATION TO SOLVE A LARGE-SCALE NONLINEAR MIXED INTEGER PROGRAMMING PROBLEM†

F. GLOVER

Department of Management Science, University of Colorado, Boulder, CO 80302, USA

D. KLINGMAN

David Bruton, Jr., Centennial Chain in Business Decision Support Systems, University of Texas at Austin, Austin, TX 78712, USA

N. PHILLIPS

Department of Operations Research, University of Texas at Austin, Austin, TX 78712, USA

and

G.T. ROSS

Production and Distribution Planning Systems, Analysis, Research, and Computation, Inc., Austin, TX 78767, USA

Abstract

This paper describes the formulation of a nonlinear mixed integer programming model for a large-scale product development and distribution problem and the design and computational implementation of a special purpose algorithm to solve the model. The results described demonstrate that integrating the art of modeling with the sciences of solution methodology and computer implementation provides a powerful approach for attacking difficult problems. The efforts described here were successful because they capitalized on the wealth of existing modeling technology and algorithm technology, the availability of efficient and reliable optimization, matrix generation and graphics software, and the speed of large-scale computer hardware. The model permitted the combined use of decomposition, general linear programming and network optimization within a branch and bound algorithm to

†This research was supported in part by the Office of Naval Research Contract N00014-78-C-0222, by the Center for Business Decision Analysis*, by the University of Texas at Austin, and by the David Bruton, Jr., Centennial Chair in Business Decision Support Systems. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

*Center for Business Decision Analysis, Graduate School of Business – GSB 3.126, University of Texas, Austin, Texas 78712, USA.

overcome mathematical complexity. The computer system reliably found solutions with considerably better objective function values 30 to 50 times faster than had been achieved using general purpose optimization software alone. Throughout twenty months of daily use, the system was credited with providing insights and suggesting strategies that led to very large dollar savings.

Keywords and phrases

Nonlinear programming, decomposition, branch and bound, network, transportation, mixed integer programming.

Authors' note

A confidentiality agreement between the authors and the firm facing the problem described in the paper prohibits disclosure of the firm's name and operational or financial information relating to this application. We have, therefore, camouflaged a number of details concerning the business problem. The mathematical model, the algorithm and the key aspects of the implementation, however, remain intact.

1. Introduction

The need to maximize expected future revenue under complex contractual terms provided the impetus to develop a mixed integer nonlinear mathematical programming model and to implement a customized algorithm for its solution. The context which gave rise to the model is described here, although specifics of the actual case are disguised. How the structure of the model was exploited by a branch and bound algorithm that made combined use of the general purpose linear programming optimizer and a general purpose network optimizer is also discussed.

The technical accomplishments reported here are several.

First, the solutions provided by the model were high quality along several dimensions. In formal terms, they were provably 90–95% optimal. In relative terms, they were at least 20% better than any the company had devised previously, and this savings amounted to a very large sum of money. In functional terms, they were relied upon by management in making operating decisions over a period of twenty months.

Second, the computational performance of the algorithm is noteworthy, given the scale and complexity of the problem. Early attempts by the company to optimize other formulations with general purpose algorithms had faired poorly. After eight cpu

hours of time on an IBM 3033 computer, IBM's general purpose mixed integer programming system MPSX/370-MIP had found only a single feasible solution known to be only 75% optimal. (However, as discussed further below, the superiority of this suboptimal solution over the best solution derived by manual efforts convinced management of the value of an optimization model and spawned the project described here.) In contrast, the special purpose procedure described below routinely found an incumbent solution at the *initial* node of the branch and bound tree and verified 90–95% optimal solutions in 1/2 to 1 cpu hour. Typical problems had 8500 linear constraints, 100 nonlinear constraints, 12 000 continuous variables and 110 to 200 integer variables.

Third, the features of the algorithm show how optimization methodology and available optimization software tools may be combined to overcome problem complexity. Insights concerning exploitable problem structure which gave rise to the algorithm and which made the problem tractable are discussed in detail. The key features of the algorithm are decomposition (facilitated by replacing a large number of constraints by a very few constraints), relaxation and restriction (obtained by replacing nonlinear equations by linear ones), and network optimization (permitted by the decomposition).

The software implementation of the algorithm was carried out by Analysis, Research and Computation, Inc. (ARC). It involved IBM's general purpose linear programming system MPSX/370 [4], ARC's primal network optimization software ARCNET [1], a custom branch and bound routine written in FORTRAN which called both MPSX/370 and ARCNET as subroutines, and interface routines written in IBM's *e*xtended *c*ontrol *l*anguage (ECL). Further details are given later in the paper.

Computer graphics were also instrumental in the successful use of the model. Two graphs, originally designed by the user as a mechanism for manually computing a solution, were subsequently plotted from the model solution. The graphs summarized in a comprehensive and clear fashion the results of an optimization run. These graphs were helpful both in revealing the implications of a solution and in identifying anomalies in solutions when the system was being debugged and refined.

From a broader perspective, the success of this effort illustrates two important points about the state-of-the-art of mathematical programming. Foremost, it gives evidence of the value of optimization methodology in practical applications. Prior to building the model described here, the company had formed a large task force of various professionals (managers, engineers, lawyers, economists and system analysts) to explore alternatives and to develop recommendations for resolving the business issues. Some of these persons spent considerable amounts of time developing specific strategies by hand (in essence, attempting to solve the model by hand). For their time-consuming and tedious efforts, they produced answers which were technically infeasible and, without assigning a penalty for infeasibility, substantially inferior from

a revenue perspective to the suboptimal solution produced by an early mathematical programming model. Thus, while the initial mathematical programming model developed for this application was impractical to use from a computer run-time viewpoint, it was very successful in demonstrating the potential of computer-based optimization. A very conservative estimate of the benefit cost ratio for this project is 500 to 1.

In the second place, the success of the specialized algorithms demonstrates that mathematical programmers have developed robust modeling, solutions and implementation technologies. Although problems may be difficult to solve by general purpose algorithms and software, highly specialized approaches that integrate model, algorithm and computer implementation are viable.

2. Problem description and mathematical model

The ABC Corporation, as we shall refer to the company that faced the problem, was interested in establishing a five-year monthly operating plan for introducing new products and determining product distribution to customers consistent with its standing contracts, technological limitations, and governmental regulations. ABC operated in an environment characterized by volatile selling prices and changing governmental regulations which forced it to exercise care in establishing and fulfilling its obligations to customers.

The ABC Corporation had three types of customers which we will refer to by the collective designations of customer 1, 2, and 3. Customer 3 was a 'standard' customer group who paid the prevailing market price for ABC products. For customers 1 and 2, all of ABC's products were interchangeable, and they had negotiated long-term contracts which assured them access to products, independent of product type, at a price considerably below the prevailing market price of ABC's least expensive product. More precisely, according to the contractual agreement, customer 1 was entitled to receive a fixed total amount of product over the life of the agreement. This amount was provided to customer 1 at a fairly stable supply rate by month. Customer 2, likewise, was entitled to purchase a fixed amount of product at a low price and at a steady supply rate until a particular moment in time, called the critical date. Thereafter, customer 2 purchased those product types which he was supplied during a critical period at a higher price but below the market price. (The critical period was a period of time immediately preceding the critical date.) Except for a provision to be discussed, customer 2 had the right to purchase after the critical date all the remaining supply of those product types sold to him during the critical period. ABC's production of each product type was heavily dependent on the availability of certain depletable inputs and thus, annual production levels of each product type normally diminished over time.

One further proviso of the agreement complicated ABC's arrangement with customers 1 and 2. After the critical date, ABC could, at its option, deliver to customer

1 a portion of the supply of products to which customer 2 is entitled. The stipulation was that customer 1 could be sold no greater percentage of the remaining supply of a product type than the share he had received of the total amount of that product provided to customers 1 and 2 during the critical period. For example, suppose 105 units of a certain type were available during the critical period, 90 of which were sold to customer 3, 10 to customer 2, and 5 to customer 1. Then *after* the critical period, customer 1 could receive up to 1/3 of the product type available (1/3 = 5/(5 + 10)), customer 2 would receive at least 2/3 (10/(5 + 10)) of the product type available, and customer 3 would receive none. Further, no matter how small the amount of a particular type of product was used to supply customer 2 during the critical period, his rights to all of the remaining supply were assured.

Additional complications arose out of governmental regulations. In essence, ABC's products may be conceived as coming in two major classes, Orange and White. Government policy dictated that during any calendar year, not more than one-half of the commodity delivered to customer 1 could be white.

The mathematical formulation that expresses the foregoing conditions follows. A detailed explanation of the problem notation follows the mathematical statement.

$$\text{Maximize} \quad \sum_{t \in T} \sum_{i \in I} \sum_{j \in J} c_{ijt} x_{ijt} - \sum_{i \in I_N} e_i y_i \tag{1}$$

$$\text{subject to} \quad \sum_{i \in I} x_{ijt} = d_{jt} \quad \text{for } j = 1, 2,\ t \in T_B \tag{2}$$

$$\sum_{j \in J} x_{ijt} = a_{it} \quad \text{for } i \in I_E,\ t \in T_B \tag{3}$$

$$\sum_{j \in J} x_{ijt} = a_{it} y_i \quad \text{for } i \in I_N,\ t \in T_B \tag{4}$$

$$\sum_{i \in I_W} \sum_{t \in T_k} x_{i1t} \leqslant 0.5 \sum_{t \in T_k} d_{1t} \quad \text{for } k \text{ such that } T_k \text{ in } T \tag{5}$$

$$x_{i2t} \leqslant a_{it} z_i \quad \text{for } i \in I,\ t \in T_c \tag{6}$$

$$\frac{\sum_{t \in T_c} x_{i2t}}{\sum_{t \in T_c} (x_{i1t} + x_{i2t})} = q_i \quad \text{for } i \in I \tag{7}$$

$$\sum_{i \in I} x_{i1t} = d_{1t} \quad \text{for } t \in T_A \tag{8}$$

$$\sum_{j \in J} x_{ijt} = a_{it} \quad \text{for } i \in I_E,\ t \in T_A \tag{9}$$

$$\sum_{j \in J} x_{ijt} = a_{it} y_i \quad \text{for } i \in I_N,\ t \in T_A \tag{10}$$

$$x_{i1t} \leqslant a_{it}(1 - q_i) \quad \text{for } i \in I,\ t \in T_A \tag{11}$$

$$x_{i3t} \leqslant a_{it}(1 - z_i) \quad \text{for } i \in I,\ t \in T_A \tag{12}$$

$$\epsilon z_i \leqslant q_i \leqslant z_i \quad \text{for } i \in I \tag{13}$$

$$z_i \leqslant y_i \quad \text{for } i \in I_N \tag{14}$$

$$0 \leqslant z_i \leqslant 1 \quad \text{and integer for } i \in I \tag{15}$$

$$0 \leqslant y_i \leqslant 1 \quad \text{and integer for } i \in I_N \tag{16}$$

$$x_{ijt} \geqslant 0 \quad \text{for } i \in I,\ j \in J,\ t \in T, \tag{17}$$

where

I denotes the set of all product types,
I_N denotes the set of *new* types of product,
I_E denotes the set of *existing* types of product,
I_W denotes the set of product types belonging to the *white* class,
J denotes the set of customer classes,
T denotes the set of months in the time horizon,
T_c denotes the set of months in the critical period in T,
T_k denotes the set of months in the kth calendar year in T,
T_B denotes the set of months *before* the critical date (includes T_c),
T_A denotes the set of months *after* the critical date,
$x_{ijt} \equiv$ units of the product type i sold to customer j in month t,

$z_i \equiv \begin{cases} 1 & \text{if type } i \text{ is used to supply customer 2 during the critical period} \\ 0 & \text{otherwise} \end{cases}$

$q_i \equiv$ portion of the amount delivered of type i to customer 2 of the total delivered to customers 1 and 2 during the critical period,

$y_i \equiv \begin{cases} 1 & \text{if new type } i \text{ product is acquired} \\ 0 & \text{otherwise} \end{cases}$

$a_{it} \equiv$ forecasted supply of type i in month t,

$d_{jt} \equiv$ expected demand of customer j in month t,

$c_{ijt} \equiv$ discounted net present value of gross margin (selling price – delivered cost) per unit of product of type i sold to customer j in month t,

$e_i \equiv$ investment cost required to acquire product of type i,

$\epsilon \equiv$ very small positive value to assure that q_i is positive if z_i is one.

The foregoing mathematical representation is admittedly somewhat complex. However, a brief descriptive breakdown of its components will help to clarify the model structure and make its features more understandable. The following remarks should be interpreted relative to the verbal description of the problem that preceded the formulation.

The objective function (1) measures the total discounted net present value of future revenues associated with sales to all customers, less any cost associated with the acquisition of new types of products. The first few constraints are chiefly 'technological' constraints affecting the critical period. Constraints (2) ensure that customers 1 and 2 will receive their appropriate known amounts during the time periods preceding the critical date. Constraints (3) and (4) ensure that a product of a given type delivered to customers equals the amount available in each time period before the critical date. Constraints (4) also reflect the impact of deciding not to acquire a new product type.

The next several constraints have to do with regulatory and contractual considerations. Constraints (5) ensure meeting the governmental regulation limiting delivery of white product to customer 1 during any calendar year of the model horizon. Constraints (6) and (7) are best understood when considered in conjunction with constraints (11) and (12). Constraints (11) reflect that the maximum share of the supply of a product type that can be sold to customer 1 is one minus the share that must go to customer 2. Constraints (12) preclude any sales to customer 3 after the critical date of those types used to supply customer 2 during the critical period.

Constraints (8), (9) and (10) perform the same role in the model for time periods after the critical date as constraints (2), (3) and (4) do for periods before

the critical date. Note, however, that there is fixed demand only for customer 1, which will be important in the decomposition decsribed below. Constraints (13) through (17) express basic relationships needed for logical consistency.

The constraints that pose the greatest mathematical complexity are those of (7), each of which is nonlinear and has a point of discontinuity. These constraints, the large problem size, and the presence of integer variables, combine to make the problem exceedingly difficult to solve by any standard measure. In fact, the size of the problem itself is formidable. Depending upon the particular scenario management needed to analyze, a typical model comprised 40 time periods, 70–130 types of existing products, and 5 to 10 types of new products. This resulted in models with 7000 to 8500 constraints (not counting simple upper bounds on variables) and 12 000 continuous variables. The number of discrete z_i variables (and associated q_i variables) varied between 60 and 200 depending upon the scenario, with 100 to 130 being typical. This meant that there were normally 100 to 130 constraints of the form (7) in the model.

3. Solving the model

When we were confronted with this problem, there was no existing algorithm that could reasonably be expected to solve the model as specified. Most nonlinear programming algorithms require that nonlinear functions be differentiable and the combination of non-differentiability and integer variables puts the problem beyond the scope of those algorithms [3,5,6]. Other formulations had been considered by ABC, including variants which allowed q_i to take on discrete values (e.g. 0.05, 0.10, 0.15), but they increased the number of constraints and the number of integer variables. Attempts were made by ABC to solve one such model, but it proved exceedingly difficult to optimize. IBM's system for mixed integer linear programming MPSX/370-MIP required eight cpu hours on an IBM 3033 computer to find a single integer solution, and this could only be proven to be within 25% of optimality. Next, ABC tried Beale's separable programming approach, without success [2].

In the context of a branch and bound method for handling the discrete variables, the structure of the model and the terms of the contract itself suggested that a decomposition approach be used to obtain tractable subproblems to solve at each step. The natural idea is to segregate the problem into two major components, creating one model for decisions up through the critical period and a second model for time periods following the critical date. The naturalness of this approach was reinforced by the fact that the number of periods before (19 periods) and after (21 periods) the critical date were close to the same.

However, an additional more powerful strategy was required to make this type of decomposition effective. Assume for the moment that no constraint of (5) includes variables from both $t \in T_B$ and $t \in T_A$. (This assumption is incorrect but

can be overcome, as will be shown.) Then it follows that the model decomposes into one covering T_B and one covering T_A. The T_B model includes constraints (2)–(7) and (13)–(17). The T_A model includes constraints (8)–(12) and (17). Constraints (17) appear in both models, as do the relevant portions of the objective function. Disregarding the fact that the T_B model is a difficult nonlinear mixed integer problem, solving it would yield values for z_i and q_i to use in the T_A model. However, solutions to the T_B model might provide terribly bad values for z_i and q_i because they do not appear in the T_B objective, and they are the single most important factors affecting feasibility and the objective function value of T_A.

In the following, we describe the special observations and techniques we developed to overcome the limitations noted above and transform the indicated decomposition approach into a highly successful solution strategy.

PROPOSITION A

Given values for z_i, q_i and y_i, the T_A model can be formulated as a set of disjoint (one per calendar year) capacitated trans-shipment problems.

The justification of this proposition requires some mathematical manipulation. Before providing a technical exposition, we simply illustrate the form of the resulting capacitated trans-shipment problem and comment on its features. Figure 1 provides such an illustration for a simplified problem in which T_A consists of two time periods, both time periods are in calendar year T_k, and there is exactly one type of orange product and one type of white product. Constraints (8)–(12) are handled in this diagram as follows. The nodes labeled $S_{i,t}$ denote product type i in period t and those labeled C_{1t} denote customer 1 in period t. The node O_{T_k} represents orange product sold to customer 2 or 3 during calendar year T_k. Each S node has three arcs directed from it representing flows to customers 1, 2 and 3, respectively. The lower and upper bounds on the arcs (denoted by the quantities in parentheses) reflect the impact of the z_i and q_i variables from the T_B model. The upper bound on the arc from O_{T_k} to W_{T_k} restricts the flow of orange product to customers 2 and 3, thereby forcing at least half of customer 1's demand to be satisfied by orange product and ensuring no more than half of the demand would be satisfied by white product.

Proof

Clearly, for given values of z_i, q_i, and y_i, the T_A model separates by calendar year. Constraint (5) couples together period constraints within a calendar year. For a given calendar year T_k in T_A, constraints (5) can be rewritten, however, as follows:

$$\sum_{i \in I-I_W} \sum_{t \in T_k} (x_{i2t} + x_{i3t}) \leqslant \sum_{i \in I-I_W} a_{it} z_i - 0.5 \sum_{t \in T_k} d_{1t}. \qquad (5.1)$$

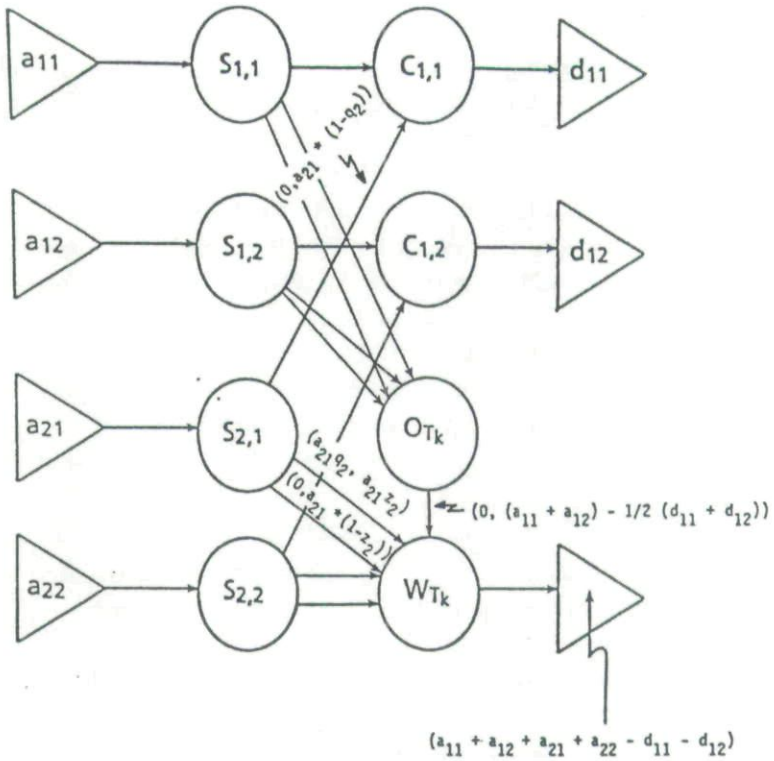

Fig. 1. A network flow representation of a T_A model.

Constraint (5.1) may be replaced by constraints (5.2) and (5.3).

$$\sum_{i \in I - I_W} \sum_{t \in T_k} (x_{i2t} + x_{izt}) - y = 0 \tag{5.2}$$

$$0 \leqslant y \leqslant \sum_{i \in I - I_W} a_{it} z_i - 0.5 \sum_{t \in T_k} d_{1t}. \tag{5.3}$$

From constraints (8), (9) and (10), form the redundant constraint:

$$\sum_{t \in T_k} \sum_{i \in I} \sum_{j \in J} x_{ijt} - \sum_{t \in T_k} \sum_{i \in I} x_{ijt}$$

$$= \sum_{i \in I_E} a_{it} y_i + \sum_{i \in I_N} a_{it} y_i - \sum_{t \in T_k} d_{1t}. \tag{18}$$

Subtracting (5.2) from (18) yields the redundant constraint:

$$\sum_{t \in T_k} \sum_{i \in I_W} (x_{i2t} + x_{i3t}) + y = \sum_{i \in I_E} a_{it} + \sum_{i \in I_N} a_{it} y_i - \sum_{t \in T_k} d_{1t} . \quad (19)$$

Replacing constraint (5) with constraints (5.2) and (5.3) and augmenting each calendar year submodel of the T_A model with constraint (19) results in a T_A model composed of a set of disjoint capacitated trans-shipment problems for given values of z_i, q_i, and y_i.

The major effect of proposition A was to permit the T_A model to be decomposed into a set of six network flow models (one for each calendar year spanning the 61 periods of T_A), each capable of being solved very quickly by the general purpose network flow code ARCNET. The typical size of each network was 650 nodes and 1900 arcs. The optimization time per network never exceeds one second. ECL made it easy to pass variable values from MPSX to ARCNET, and the general purpose features of ARCNET made it easy to edit the network problem files given these values.

The reformulation of the T_A model to a set of network models and subsequent analysis of the special structure of these networks lead to two additional important insights which are characterized in proposition B and C below.

PROPOSITION B

A set of simple constraints can be generated whose feasible solutions are necessary and sufficient to provide values for q_i that are feasible to the network flow representation of the T_A model.

Proof

Consideration of the network flow problem illustrated in fig. 1 discloses that q_i must satisfy the following conditions for there to be any feasible solution to the T_A model:

$$\sum_{i \in I} a_{it} q_i \leqslant \sum_{i \in I_E} a_{it} + \sum_{i \in I_N} a_{it} y_i - d_{1t} \quad \text{for each } t \in T_A . \quad (20)$$

Were constraints (20) not satisfied, there would be insufficient supply available to meet customer 1's demand in each period.

The network formulation also implies that the q_i must satisfy the following constraints for there to be any feasible solution to the T_A model:

$$\sum_{t \in T_k} \sum_{i \in I_O} a_{it} q_i \leqslant \sum_{t \in T_k} \left(\sum_{i \in I_O \cap I_E} a_{it} + \sum_{i \in I_O \cap I_N} a_{it} y_i \right) - \sum_{t \in T_k} 0.5\, d_{1t} \tag{21}$$

for all k such that T_k in T_A.

Here, I_O denotes the set of product types belonging to the orange class. These constraints ensure that the restrictions governing the percentage of orange product which must go to customer 1 on a calendar year basis are met. They are based upon the conservation of flow through node O_{T_k} in the network diagram. That is, since customer 1 can receive no more than one half of his demand in white product, he must receive at least half of it in orange. This, in turn, bounds the amount of orange product that can go to customers 2 and 3. The sufficiency of these constraints to provide feasible network solutions follows from the fact that there is only one specified demand per time period in the T_A model. Constraint (20) ensures that there is sufficient product to meet this demand and constraint (21) ensures that the orange product percentage can be accommodated while meeting the requirement that total demand equal total supply.

Constraints (20) and (21), which numbered about 24 in the actual problem, were added to model T_B to ensure feasibility for approximately 4000 constraints in model T_A.

It was assumed earlier that no constraint (5) included variables from both $t \in T_B$ and $t \in T_A$ (or in other words, that T_B ended at the end of a calendar year and T_A began at the start of the subsequent calendar year). In fact, this was not the case. For exactly one k, T_k partitioned into two subsets $T_k \cap T_B$ and $T_k \cap T_A$. To effect the decomposition, it was necessary, for that k only, to change constraints (5) in model T_B to:

$$\sum_{i \in I_O} \sum_{t \in T_k \cap T_B} x_{i1t} + u_k = 0.5 \sum_{t \in T_k} d_{1t},$$

where u_k is an unrestricted variable, and for this k to change constraint (21) to

$$\sum_{t \in T_k \cap T_A} \sum_{i \in I_O} a_{it} q_i \leqslant \sum_{t \in T_k \cap T_A} \left(\sum_{i \in I_O \cap I_E} a_{it} + \sum_{i \in I_O \cap I_N} a_{it} y_i \right) - u_k.$$

The variable u_k measures the extent to which the restriction that no more than one-half of the demand during a calendar year be met with orange product is satisfied by flows in T_B so that flows in T_A must compensate to ensure the constraint is met.

PROPOSITION C

Penalty coefficients for the q_i variables to incorporate into the T_B objective can be deduced from the network flow model.

This proposition can be demonstrated straightforwardly as follows. Since customer 3 pays more for product than customer 2, the solution to the network form of the T_A model with all $z_i = q_i = 0$ represents an upper bound C on the revenues that could be obtained from sales during T_A. Moreover, as q_i increases from 0, the effect on the solution of the T_A model is that flow on the arc corresponding to sales to customer 3 will move to the arc corresponding to sales to customer 2. The impact on the model T_A objective is a change of at least $\Sigma_{t \in T_A} a_{it}(C_{i3t} - C_{i2t})q_i$. Hence $\Sigma_{t \in T_A} a_{it}(C_{i3t} - C_{i2t})$ is a legitimate penalty to attach to q_i in the T_B model objective. Coupling the use of this penalty with adding the constant C to the T_B model objective produces an optimistic estimator of the sum of the T_A and T_B objectives.

Propositions A, B and C, in combination, make the decomposition approach a viable solution strategy. Its effectiveness derives from the following principal reasons:

(a) Model T_A is very easy to solve.

(b) The solution of a suitable linear relaxation of T_B by *itself* yields a strong bound in a branch and bound scheme.

(c) The segregation of the T_A and T_B components provides a strong problem representation because feasibility to the constraints of model T_A are incorporated in constraints (20) and (21) and made part of model T_B.

To provide a complete solution algorithm based on this decomposition, it was critical that some means be found to solve a linear relaxation of model T_B effectively. The chief requirement is an effective mechanism for handling the nonlinear constraints of (7).

The last major key to the success of the decomposition is the following:

NONLINEAR CONSTRAINT STRATEGY

Constraints (7) are replaced alternately by linear constraints which underestimate and overestimate the true value of each q_i.

An underestimating value for each q_i is easy to achieve. For the purpose of computing such an estimate, the constraints of (7) may clearly be replaced by

$$\frac{\sum_{t \in T_c} x_{i2t}}{\sum_{t \in T_c} a_{it}} = q_i . \tag{7u}$$

For the purpose of finding feasible solutions to T_B, two possibilities exist:

(a) Constraint (7) could be replaced by (7u) and the LP relaxation of model T_B solved. The true value of each q_i could be computed using the underestimating solution, using the values of the x_{i2t} found using (7u) in the expression for q_i in (7). If these values are feasible to constraints (20) and (21) and if the y_i and z_i are integer (or rounded), then a candidate incumbent solution can be determined for the problem as a whole by re-solving model T_B with q_i fixed at the true values, z_i and y_i fixed, and solving model T_A with the same values of q_i, z_i and y_i.

(b) The LP relaxation of model T_B can be solved with constraint (7) replaced by

$$\frac{\sum_{t \in T_c} (x_{i2t} + x_{i3t})}{\sum_{t \in T_c} a_{it}} = q_i , \tag{7o}$$

which clearly overestimates the true value of q_i. If the LP relaxation of model T_B is feasible, then the true q_i corresponding to the overestimate of q_i are feasible to model T_A because constraints (20) and (21) are satisfied by the unnecessarily large values of the q_i. Further, all of the z_i and y_i can be rounded to yield a feasible solution to model T_B. Thus, a candidate incumbent solution can be generated as described above.

4. Computational implementation of the decomposition algorithm

Implementation of the algorithm was a challenging task because of the variety of optimization problems that might be solved at any node of the branch and bound tree and the attendant 'housekeeping' that was required. For example, to obtain the constant C, the network flow model T_A had to be solved with all $z_i = q_i = 0$. When any incumbent was to be generated, the network flow model had to be edited to reflect the appropriate values of z_i and q_i in the lower and upper bounds associated with an arc. Further, several LP relaxations of model T_B might need to be solved at a node. These included forms with penalty coefficients on the q_i and either the under- or overestimating constraints (7o) or (7u) active, and others with 0 coefficients on the q_i variables and each q_i assigned a fixed value in constraints (7).

Thus, there needed to be routines to perform such edits on the LP matrix as fixing variable bounds, activating or deactivating rows and changing objective function coefficients 'on the fly' during the branch and bound. To expedite the solution process, optimal bases needed to be saved and restored for the overestimating, underestimating or incumbent defining forms of the problem. Finally, to permit restart capability in case of a system crash during an extended run, it was desirable for the

program to perform periodic array dumps. This same capability enabled users to solve a problem in several computer runs, to review the progress made in the latest run, or to change parameters (such as the increasing optimality tolerance) between runs.

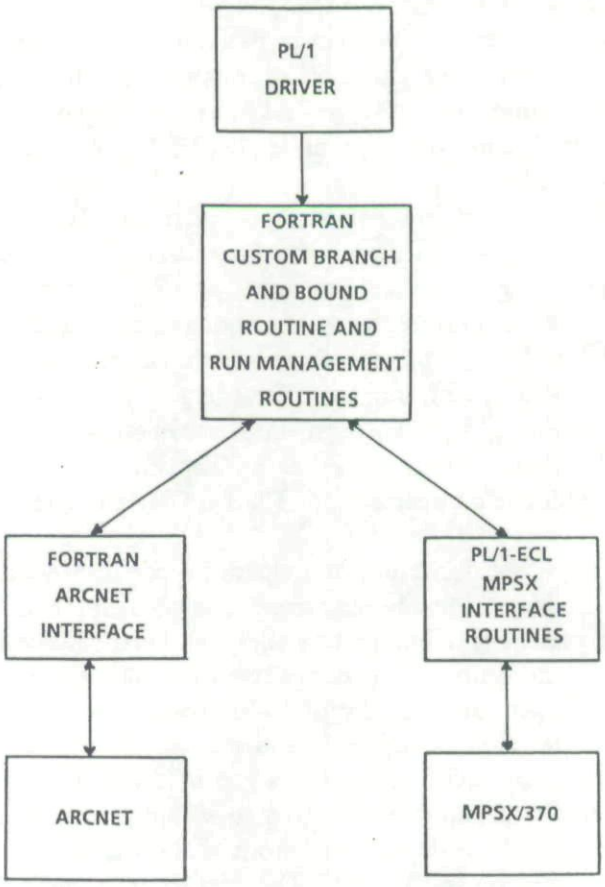

Fig. 2. Principal software components and interrelationships.

The major components of the system and their interrelationships are represented in fig. 2. This design exploits the capability of PL/1 to interface simultaneously with FORTRAN and with MPSX (by the way of the extended control language feature of MPSX). First, a PL/1 main program was written for the sole purpose of initiating the FORTRAN branch and bound routine that was the heart of the system. The FORTRAN program was responsible for the usual branch and bound tasks of maintaining the candidate list, selecting separation variables, and maintaining a list of

incumbents. Further, it ascertained from the model solution and the stage of the algorithm what problem edits were necessary. Second, the retrieval of solutions and the edits to problems were carried out via special purpose routines written in PL/1–ECL. Optimal solutions could be passed from MPSX arrays to the FORTRAN program and lists of parameters, variables or structural components and how they should be changed could be passed via arrays to MPSX. In all, some sixteen P/L1 routines were used. Third, similar routines were written in FORTRAN to enable the network flow problems to be edited and to pass solution information back to the branch and bound routine. Lastly, two optimizers, IBM's general linear programming system MPSX/370 and ARC's primal simplex network optimizer ARCNET, provided the solution power on which the decomposition rested.

Initial implementation required three calendar months. Refinements, mostly to the branching strategy, continued on a part-time basis over three additional months. The branching strategy that worked best gave priority to those q_i for which the difference between the underestimate and the actual value q_i was greatest. The degree of error weighted by the penalty coefficient (defined by proposition C) was the deciding factor, so that a q_i with large error and large penalty would lead to a separation based upon setting $q_i = 0$. Through input parameters, the branch and bound routine could be directed to select candidate problems in LIFO order or based upon the value of each candidate's upper bound. The LIFO strategy reliably outperformed the alternative.

By all measures, the decomposition algorithm performed extremely well. The overestimating constraints (7o) were instrumental in identifying an incumbent at the initial node of the branch and bound tree that was invariably within 12%–17% of the upper bound on the optimum objective function value determined at that stage of the solution. In longer runs (usually of 1000 nodes or less), the gap between the best incumbent and the optimum upper bound was usually reduced to 7% and often to 5% or less. A 95% optimality level (i.e. a gap of 5%) was used as a termination criterion. Several unique scenarios were solved to within 1% or 2% of optimality and required very few nodes. Run times varied from as short as a few cpu minutes to as long as fourteen cpu hours on an IBM 3033. The time to solve the initial linear program was typically around six cpu minutes and reoptimization after a branch took around forty cpu seconds. ARCNET required consistently 3–4 cpu seconds in total to solve all of the networks that comprised model T_A. Large numbers of incumbents were generated in every run, and a list of the five best incumbents encountered was maintained by the branch and bound routine. This experience documents that the combination of off-the-shelf optimizers, sophisticated interfaces like ECL and customized algorithms can be very effective and very powerful.

Computer graphics played an important role in reporting solutions and summarizing input data. As mentioned earlier, the persons responsible for formulating strategies by hand for fulfilling contract terms had been using two graphs to develop their plans. When the first solution based on a mathematical model was plotted in the

same format, it was immediately apparent that the model had found a markedly better solution. This contributed greatly to the credibility of a modeling approach because the users could literally see the improvements. The graphical representation was extremely powerful for validating solutions and for comparing solutions to different scenarios. Once, early in the project, an error in a right-hand side coefficient led the model to generate a solution that violated contract terms and this was readily apparent on one of the graphs. That a solution to a 12 000 variable 8 000 row mathematical programming problem can be so well communicated by two graphs speaks highly for the power and importance of graphics in presenting complex results in a friendly comprehensible fashion.

5. Summary

This paper documents the power that mathematical programming based systems offer management in developing good solutions to difficult, substantive problems. The stakes in this application justified the effort to build a special purpose algorithm and software system for solving the problem. Clearly, building blocks exist which can be used readily to develop valuable decision analytical systems.

References

[1] *ARCNET User's Manual* (Analysis, Research and Computation, Inc., Austin, Texas, 1981).
[2] E.M.L. Beale, *Mathematical Programming in Practice* (Pitman, 1968).
[3] D.M. Himmelblau, *Applied Nonlinear Programming* (McGraw-Hill, New York, 1972).
[4] *IBM Mathematical Programming System Extended/370 (MPSX/370) Program Reference Manual,* 3rd Edition (International Business Machines Corporation Technical Publications Department, White Plains, NY, 1978).
[5] L. Lasdon and A. Waren, Survey of nonlinear programming applications, Oper. Res. 28 (1980)1029.
[6] F. Palacios-Gomez, L. Lasdon and M. Engquist, Nonlinear optimization by successive linear programming, Management Science 28(October, 1982).