# THE NETFORM CONCEPT: A MORE EFFECTIVE MODEL FORM AND SOLUTION PROCEDURE FOR LARGE SCALE NONLINEAR PROBLEMS

by

Fred Glover and Claude McMillan, Professors, University of Colorado
Darwin Klingman, Professor, University of Texas, BEB 608, Austin, TX 78712

## ABSTRACT

Recent years have seen many important advances in the solution of network problems. New solution algorithms and implementation techniques have dramatically reduced the cost of solving linear and convex network flow problems. For example, the cost of solving network problems with 2400 equations and 500,000 arcs on an IBM 360/65 has been reduced from $30,000 in 1968 to $300 in 1976 by these advances. In addition, these advances have stimulated the development of new nonlinear modeling techniques for handling a multitude of problems that arise in applications of scheduling, routing, resource allocation, production, inventory management, facilities location and other areas.

This paper presents modeling techniques which are mathematically and symbolically linked to network and augmented network structures. These modeling techniques are called the NETFORM (network formulation) concept or approach. The pictorial aspect of this approach has proven to be extremely valuable in both communicating and refining nonlinear and combinatorial relationships. Additionally, the NETFORM concept often yields a formulation that enables the problem to be solved as a sequence of linear network problems with dramatic gains in efficiency over alternative approaches. The paper illustrates these attributes by providing a concrete example of a NETFORM model construction. Three real world applications are then described which have profited by the use of NETFORM techniques.

## 1. INTRODUCTION

Recently many important problems have been modeled and solved as networks. This is partially due to the development of extremely efficient solution procedures for solving networks. These advances are documented by an impressive array of empirical studies and successful real world solution efforts. (See, for example, the survey articles [3] and [4].)

The significance of these advances is illustrated by the fact that a network problem with 1,000 nodes and 7,000 arcs can be solved in 6 seconds on an IBM 370/155 using the highly efficient network code of [2]. The cost of such a computer run is about $1. To solve such a problem using the best commercial LP code would take about 10 minutes and cost $200. For larger network problems these advances have had especially dramatic consequences. A recent practical illustration is that of a network flow model used by the U.S. Treasury to handle microdata file merging problems. The problem is immense, involving 5,000 nodes and 625,000 arcs. Using the in-core out-of-core code of [1], this problem was solved in 9 minutes on a UNIVAC 1108, at a cost of about $90. Allowing an optimistic estimate, a good commercial LP code would require somewhat more than 20 hours of central processing time to solve this problem, at a cost of about $24,000.

In view of these developments the question arises as to whether comparable gains might be possible for the multitude of nonlinear, "network-related" problems that arise in scheduling, routing, resource distribution, facilities location, production and inventory management, and related areas. We argue on the basis of practical experience that such gains are possible not only for problems conspicuously related to networks, but for many other problems as well by use of a new modeling technique called *NETFORM*.

The NETFORM concept is the driving principle (or philosophy) behind a growing body of techniques by which a wide variety of nonlinear problems can be given a pictorial, network-related formulation. These network-related formulations often lead to different and more effective ways of handling these problems than in the customary (often more laborious) model formulations. This is especially true for problems attended by combinatorial (i.e., integer programming) types of restrictions.

Once an individual learns to visualize problems in network-related frameworks, the initial "recognition" of many nonlinear problems is made far easier. This change of orientation by which problems are directly visualized and represented as NETFORMS has highly beneficial effects along several dimensions. First, management scientists are able to use these NETFORMS to explain their models to the users more effectively. Second, the nonscientific user can be easily taught how to formulate his problems via such diagrams and pictorial representations which management scientists can then analyze and refine. The combined effects of these advantages make it possible for management scientists and users to interact more fully, thereby leading to clearer mutual understanding of the problem formulation and of the potential meaning and usefulness of its solution.

In the subsequent sections we will first present an example of how to model a zero-one integer programming problem, unrelated to networks, as a NETFORM. Then we will describe three real world

applications that have profited by the use of NETFORM techniques.

## 2. THE NETFORM APPROACH ILLUSTRATED IN THE CONTEXT OF AN IP PROBLEM

In the NETFORM approach to problem modeling, as in traditional network modeling, we begin with a verbal description of the problem. From that we formulate a NETFORM schematic. This schematic, accompanied by stipulations concerning the flows on particular arcs (such as the requirement that some flows bear specific relationships to other flows, or that some flows must be either 0 or 1, etc.), in itself provides a rigorous problem formulation. For more complex problems, it is convenient to devise a preliminary NETFORM schematic that is supplemented by accompanying verbal notations. Such a preliminary schematic is especially useful, at all stages, for communicating with nontechnical people about the problem.

The NETFORM representation is mathematically equivalent to any of the algebraic representations that can be arrived at by customary mathematical programming formulation techniques. However, by the creation of the network-related structures, which may or may not be implicit in any customary formulation, the NETFORM representation permits the application of specialized solution methods, tailored to employ recent algorithmic advances for exploiting the graphical relationships of network components. The progression, then, is from problem description to NETFORM schematic to specialized solution procedure.

In the illustration that follows, however, we demonstrate the NETFORM approach by starting with the algebraic representation of zero-one IP problem and displaying its NETFORM equivalent. Thus, in this illustration we progress from a standard algebraic representation to a NETFORM schematic.

Consider the 0-1 problem:

Problem 1

$$\text{Minimize} \quad 3x_1 + 4x_2 + 7x_3 + 5x_4$$

Subject to:

$$2x_1 + 4x_3 - 4x_4 \geq 2$$

$$-1x_1 + 10x_2 - 2x_3 - 6x_4 \leq 7$$

$$4x_1 + 1x_2 + 4x_4 = 5$$

$$1x_3 + 1x_4 \geq 1$$

$$x_1, x_2, x_3, x_4 = 0 \text{ or } 1$$

### 2.1 Graph Construction

In the first constraint, note that the lower bound identified by the right hand side ( $\geq 2$) may be viewed as a form of demand. If variable $x_1$ is given a value of 1 rather than 0 then $x_1$ serves to help meet the demand, and thus plays a supply role. In a similar way, setting $x_1 = 1$ contributes to (or detracts from) demands expressed by the other constraints. The "supply/demand" orientation is useful for understanding the intuitive basis of the NETFORM representation that we now develop for this example.

To construct this representation, we create nodes $x_1$, $x_2$, $x_3$, and $x_4$, one for each 0-1 variable. We also create origin node 0 and provide arcs, called *variable arcs*, by which node 0 supplies these *variable nodes*, as shown in Figure 1. Flows on these four variable arcs correspond to the values of the four 0-1 variables. To insure that flow on these arcs lies in the interval from 0 to 1 we associate a lower bound of 0 and an upper bound of 1 on each variable arc. These bounds are displayed within parentheses in Figure 1. (Optimal values of these flows are ultimately determined by the solution procedure to be applied to the NETFORM.)

Costs on the arcs originating at 0 correspond to the objective function coefficients of the associated variables, respectively, and are shown in the rectangles on these arcs. A flow of 1 unit on arc $(0,x_3)$, for example, corresponds to setting $x_3 = 1$ in the algebraic representation of the problem and incurs a cost of 7.

Thus far nothing new has been introduced; only traditional network conventions have been utilized. One of the additional conventions often used by the NETFORM approach is the introduction of an *arc multiplier*, or gain factor, on some arcs. In Figure 1 a multiplier is specified on the arc to which it applies by enclosing it inside a triangle.

The addition of a multiplier to an arc creates what is called a *generalized arc*. The multiplier indicates that the flow entering the arc is multiplied by the value of the multiplier as that flow leaves the arc. For instance, a flow of 1 unit entering arc $(0,x_1)$ is multiplied by 3 to become a flow of 3 units as it leaves the arc at node $x_1$. The arc cost, it should be noted, and the arc's lower and upper bounds, apply only to the units of flow *entering* the arc.

In Figure 1 we associate with each variable arc a multiplier whose value is equal to the number of constraints in which that variable appears with a nonzero coefficient in Problem 1. Thus, arc $(0, x_1)$ has a multiplier of 3, since variable $x_1$ appears in constraints 1, 2, and 3. The reason for selecting this multiplier value will be discussed subsequently.

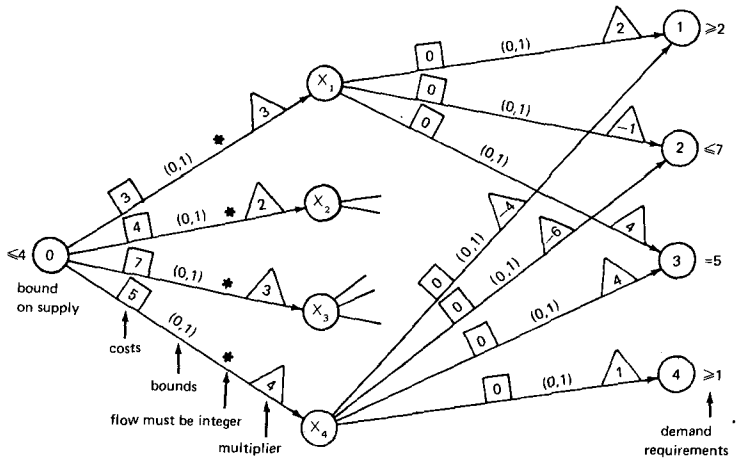
Problems with generalized arcs may have arc flows that are fractional valued (i.e., non-integer) in an optimal basic solution. Thus, it is necessary to employ a second convention, common to the NETFORM approach, of requiring certain arcs to receive integer-valued flows. This requirement is indicated by attaching an asterisk to an arc.

In Figure 1 the flow on each variable arc is required to be integer-valued. Coupling this restriction with the lower and upper bound restriction of the variable arcs implies that the flow of these arcs must be 0 or 1. Further, the multiplier assigned to each variable arc together with the preceding observation, implies that either zero flow is transmitted to a variable node or a flow equal to the number of constraints that this variable appears in the algebraic statement is transmitted to its variable node.

Using these observations, we can now complete an equivalent graphical formulation of the algebraic statement of Problem 1. To do this, we create *constraint nodes* 1, 2, 3, 4 for the four constraints. Each variable node is then connected to the constraint nodes, via *constraint arcs*. These constraint arcs, for a given variable, are associated with the constraints in Problem 1 in which that variable appears with a nonzero coefficient. See Figure 1. (To keep the schematic simple, arcs out of nodes $x_2$ and $x_3$ have been left incomplete.) Each constraint arc is given a lower bound of 0, an upper bound of 1, a cost of 0, and a multiplier

Figure 1

NETFORM FORMULATION




equal to the actual constraint coefficient of the variable with which this arc is associated. To illustrate, the multipliers on the constraint arcs leaving node $x_1$ are 2, -1, and 4, which are the coefficients of variable $x_1$ in Problem 1 for constraints 1, 2, and 3, respectively.

Finally, we associate with each constraint node a demand equal to the constraint bound (inequalities and equalities involving the right hand side coefficients) specified in Problem 1. Figure 1 contains the completed formulation.

## 2.2 Model Equivalence

The equivalence of Problem 1 and Figure 1 will now be made apparent by extending our earlier observations concerning the combined effect of the arc and node attributes. Consider origin node 0 as a supply node, equipped to supply a maximum of four units (the maximum number of 0-1 variables that can receive a value of 1). If flow on the arc $(0,x_1)$ is 0 (i.e., $x_1 = 0$ in Problem 1) then no flow is transmitted to any of the constraint nodes via this variable arc and variable $x_1$ contributes nothing to meeting the right hand side requirements.

If on the other hand the flow on the arc $(0, x_1)$ is 1, then the multiplier causes 3 units of flow to enter node $x_1$. The bounds of (0,1) on each constraint arc leaving node $x_1$ assure that each of these arcs receives exactly 1 unit of flow. These flows, amplified by the multipliers on the constraint arcs, reflect variable $x_1$'s contribution to the three constraints in which $x_1$ appears. In a similar fashion, the contributions of variables $x_2$, $x_3$, and $x_4$ to the four constraints are also reflected correctly in Figure 1.

These flow possibilities may also be viewed alternatively from the standpoint of the constraints. Thus, for example, flows into constraint node 1 account for the contribution of variables $x_1$, $x_2$, $x_3$, and $x_4$ to the demand requirements specified by constraint 1 in Problem 1.

It should be evident that the schematic in Figure 1 is the precise equivalent of Problem 1. Note further that the constraints in the algebraic problem can be generated from the NETFORM schematic of Figure 1. Viewed from the standpoint of its graphical structure, this schematic also gives rise to a nonstandard algebraic formulation of the 0-1 IP problem (via node-arc incidence relationships). Significantly, however, transformation to an algebraic form is not required. Solution methods that deal with the problem in its graphical network-related form are generally more powerful than those that deal with problems in an algebraic form.

## 2.3 Solution and Alternative NETFORM Formulations

A supply/demand and arc flow orientation is central to the NETFORM approach. The utility of this orientation is substantially responsible for the popularity of networks as model forms. However, by adding only a few additional features (arc multipliers, integer restrictions, etc.) and with the exercise of a bit of imagination, a much broader scope of real world problems, broader than that dealt with in traditional network modeling, can be modeled in the network schematic, as subsequent examples in this paper will demonstrate.

The combination of arc multipliers and the 0-1 integer restriction, as in the NETFORM representation just illustrated, gives rise to what we call a *0-1 generalized network problem.* Efficient procedures [6] have been developed for solving 0-1 generalized networks.

It is important to note that the derivation of this type of NETFORM in the preceding illustration produced the same number of integer variables as in the original algebraic formulation. The number of continuous variables added to the problem depends on the particular transformation used. (The simple approach illustrated in Figure 1 introduces more arcs and continuous variables than necessary.) In fact, just as there are different IP formulations for the same problem, we are finding that there are different types of NETFORM formulations (not all of which are "obviously" related to the type of formulation just illustrated). For example, by creating additional nodes and arcs, it is also possible to transform 0-1 IP problems into "0-U" transshipment problems where the flows on certain arcs must be equal to their upper or lower bounds [7]. Also in the case of problems with special structure, there are several specializations and simplifications of these basic ideas

that hold promise of permitting more effective treatment by appropriately designed algorithms.

The preceding formulation techniques extend quite naturally to accommodate mixed integer 0-1 problems where the continuous part of the problem is a transportation, transshipment, or generalized network problem itself. Many important real world applications have such a "mixed" structure; among them are a variety of energy models, plant location models, and physical distribution models.

## 3. THREE REAL WORLD APPLICATIONS

### 3.1 Air Force Course Scheduling

Undergraduate Flight Training (UFT) graduates are required upon graduation to take advanced flight training and survival training courses enroute to their first operational assignment. The purpose of the advanced flight training is to qualify a pilot for a specific aircraft. Advanced flight training is offered only in formal schools usually by the Major Air Command, the principal aircraft user. Newly qualified UFT graduates will additionally require from one to four extra training courses before being assigned to a crew--e.g., basic survival (Washington), water survival (Florida), air weapons delivery (Texas), etc. These courses are only offered at certain times, have enrollment limits, and may have prerequisites. The identification of schedules is further complicated by attendance requirements at Combat Crew Training courses, various modes of transportation, the number of dead days in the pipeline, the opportunity for the UFT graduates to take leave as desired, etc.

To solve this UFT graduate scheduling problem, the Air Force developed a computer program (called the UFT Pipeline Scheduling Model) which generates from one to five feasible least cost schedules for each graduate. Using these schedules and course enrollment limits, the personnel manager in the Training Pipeline Management Division manually assigns each graduate to one of his feasible schedules. Clearly, this is a difficult and time consuming task to do by hand; further, the total cost of these manual assignments may be far from optimal.

In search of a better way, the Air Force formulated this problem as the following integer programming problem.

$$\text{Minimize} \quad \sum_{\substack{i\in I \\ j\in J}} c_{ij}x_{ij}$$

$$\text{Subject to:} \quad \sum_{j\in J(i)} x_{ij} = 1, \; i\in I = \{1,2,\ldots,r\}$$

$$b_k \le \sum_{\substack{i\in I \\ j\in J(i)}} d^k_{ij}x_{ij} \le e_k, \; k\in K = \{1,2,\ldots,m\}$$

$$x_{ij} = 0 \text{ or } 1 \text{ for } i\in I,\; j\in J(i)$$

where I is the set of men, J(i) is the set of schedules for man i, K is the set of classes, $c_{ij}$ is the cost of assigning man i to his $j^{th}$ schedule, $b_k$ is the lower class enrollment limit, $e_k$ is the upper class enrollment limit, $x_{ij}$ is 1 if man i is assigned to his $j^{th}$ schedule and 0 otherwise, and $d^k_{ij}$ = 1 if class k is in man i's $j^{th}$ schedule and 0 otherwise. The first set of constraints requires that each man be assigned to exactly one schedule. The second set of constraints assures that the class enrollment limits are satisfied.

Figure 2 illustrates an equivalent NETFORM formulation of the UFT problem. In Figure 2, the node Mi represents the $i^{th}$ man and has a supply of exactly 1. Each man node is connected by arcs to its set of man/schedule nodes. These connecting man/schedule arcs have a multiplier $a_{ij}$ equal to the number of classes in the schedule (that is, $a_{ij} = \sum_{k\in K} d^k_{ij}$) and a cost $c_{ij}$ equal to the cost of assigning man i to his $j^{th}$ schedule. The asterisk again indicates that flow must be integer-valued. The arcs emanating from a man/schedule node in Figure 2 lead to the individual classes making up the schedule. Each of these arcs has an upper bound of one. Thus, if a particular schedule is "selected," then every class in the schedule is also automatically selected. The objective is to pick a schedule for each man that will minimize the value of the assignments on the overall program, subject to the upper and lower attendance limits for each class, expressed as bounds on the arcs from class nodes to the sink nodes of Figure 2. All arc costs, except for those attached to the man/schedule arcs, are thus equal to 0.

Typically the UFT problem involves 120 men, 200 classes, and 460 schedules, giving rise to a 0-1 formulation with 520 constraints and 460 0-1 variables. The NETFORM formulation involves 460 0-1 variables, 2,200 continuous variables and 780 nodes. This represents a fair increase in size as an LP problem, but provides a relatively small network problem. The NETFORM of this application was solved using a specialized branch and bound procedure with generalized transshipment subproblems. The optimal solution was often found and verified after only 30 seconds and in some cases only required a total solution time of 10 seconds on a CDC 6600.

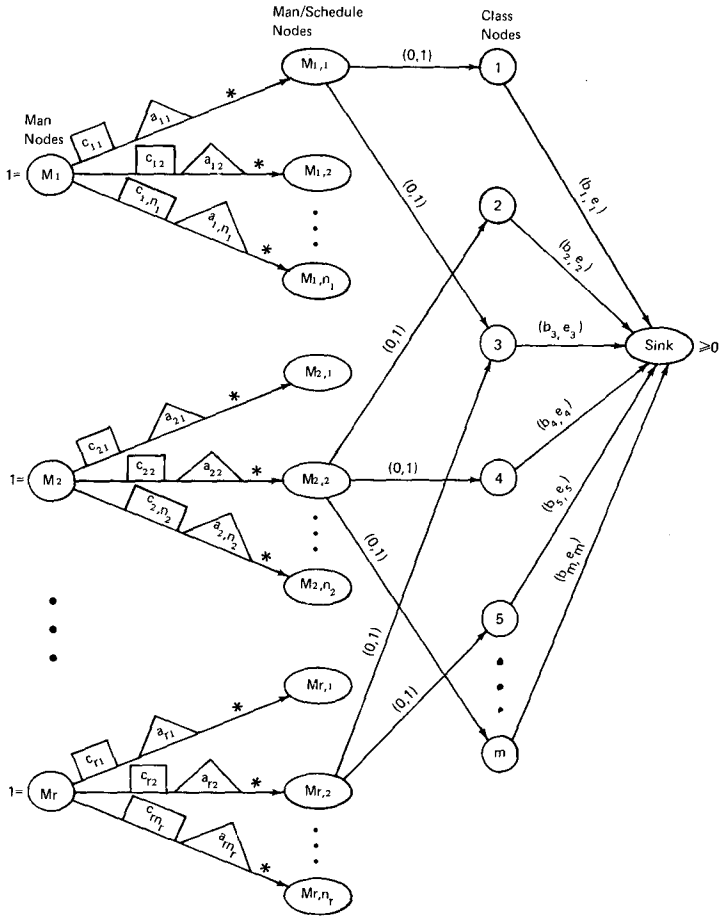
### 3.2 Optimal Lot-Sizing and Machine Loading for Multiple Products

This section describes a model currently used by a major manufacturing firm in solving a large-scale task allocation problem. The manufacturing problem involves the determination of lot-sizes for each of n products and the assignment of their production to m machines in such a way that combined set-up, production, and holding cost per unit time is minimized. The principal characteristics of the problem are:

1. The planning horizon is a single period, t weeks in length.
2. The products are designed to meet different needs and cannot be substituted for one another. Production of each product is a single-stage process.
3. Lot-sizes are selected from a predetermined finite set of $\ell$ possible lot-sizes.
4. All lots of any single product must be produced on the same machine.
5. The machines work in parallel. They are similar in function, but they may differ in their rate and cost of operation. Some machines may be capable of producing several (or all) of the products while others may be more specialized.
6. The production capacities of all machines over the planning horizon are known constants. Each machine can produce only one product at a time.
7. Demand for each product is assumed to occur continuously at a known constant rate.

Figure 2

**UFT Netform Formulation**




The characteristics just described give rise to the following mathematical model.

The binary valued decision variable $x_{ijk}$ is defined to be 1 if product j is produced on machine i in the $k^{th}$ possible lot-size and 0 otherwise. The combined set-up, production, and holding cost (per unit time) incurred when product j is produced on machine i in the $k^{th}$ possible lot-size is denoted by $c_{ijk}$. Similarly, $r_{ijk}$ denotes the capacity required on machine i to produce product j in the $k^{th}$ possible lot-size. Finally, $b_i$ denotes the aggregate production capacity of machine i over the t week planning horizon.

The problem is to determine values for the variables that

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{\ell} c_{ijk} x_{ijk}$$

Subject to:

$$\sum_{i=1}^{m} \sum_{k=1}^{\ell} x_{ijk} = 1 \text{ for } j=1,\ldots,n \quad (1)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{\ell} r_{ijk} x_{ijk} \leq b_i \text{ for } i=1,\ldots,m \quad (2)$$

$$x_{ijk} = 0 \text{ or } 1 \text{ for } i=1,\ldots,m;$$

$$j=1,\ldots,n; \; k=1,\ldots,\ell \quad (3)$$

Constraints (1) and (3) together insure that one and only one machine and lot-size combination is selected for each product. Constraint (2) insures that each machine is assigned production tasks commensurate with its capacity.

It is clear from the formulation above that the model is designed only to load the machines and that it does not schedule the work on each machine. In fact, the lot-sizes selected by the model may generate scheduling conflicts on any given machine. Although such potential conflicts are important in theory, they have not caused any difficulty in practice. Moreover, managers primarily use the model for capacity planning and prefer to retain the option of scheduling on the basis of what is "hot." That is, management retains the prerogative of determining the precise sequence and timing for implementing the candidate assignment over the horizon, in accordance with the objectives of this application. This provides flexibility to make adjustments to special conditions and changed demands, while simultaneously aiding planning functions (such as evaluating the possible use of overtime shifts in periods when the candidate assignments

tax weekly production capacities). For this type of flexibility and responsiveness to the needs of management, and to further support the analyses based on alternative assumptions of demands and capacities, it is especially important to be able to solve the model quickly for different (or recently updated) sets of data. Indeed, management has found that the best way to handle various planning and scheduling contingencies, and to obtain a clear picture of options within its complex operating environment, is to re-solve the model many times in the process of composing a weekly production schedule. Thus, the success of the application depends in large measure on the ability to solve the problem efficiently.

The firm initially tried to solve this problem using the efficient 0-1 code, RIP 30-C developed by Geoffrion at UCLA. This proved to be unsuccessful for two reasons: (1) the large array requirements of RIP-30-C made it impossible to accommodate large problems; and (2) the method required excessive computation times even to solve problems with no more than 50 variables.

Consequently, it was apparent that an alternative modeling and solution approach was needed. Figure 3 illustrates the NETFORM formulation of this problem. This NETFORM is quite simple to explain. There are two sets of nodes--a product node set and a machine node set. The arcs joining these node sets correspond to the variables $x_{ijk}$; the cost and multiplier of each of these arcs correspond to the cost and resource consumption of the associated variable. This NETFORM representation has a special bipartite network structure which is called a *generalized assignment* problem. Extremely effective techniques for solving such problems have been developed by Ross and Soland [9] and embedded in a computer code called Big-A.

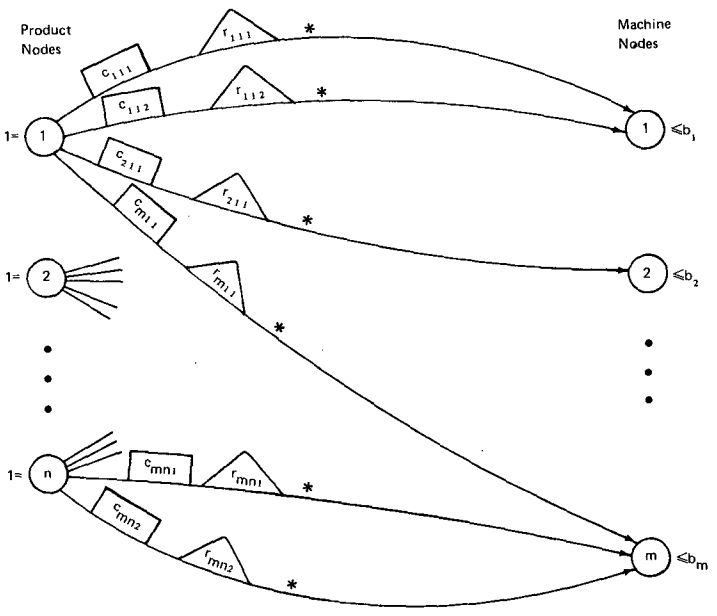
A comparison of the Big-A code with the RIP 30-C code for this problem shows that the Big-A code is from 300 to 1000 times faster. In addition, the Big-A code readily handles problems of up to 4000 variables within available computer memory. Thus, the firm now uses the NETFORM formulation coupled with the Big-A code to solve the problem. This approach makes it possible to solve problems with 106 machines, 182 products, 4 lot-size options per machine/product combination and 3772 zero-one variables in .64 seconds on a CDC 6600.

### 3.3 Refueling Nuclear Reactors

Another problem whose solution has been improved by the use of the NETFORM concept is a mixed integer programming problem for determining the minimum cost refueling schedule for nuclear reactors. This problem was initially modeled by Kazmersky [8] as a mixed integer programming problem with no apparent connection to networks. However, after working closely with Dr. Kazmersky, we discovered a way to express the problem by a NETFORM representation that was not only equivalent to the original formulation but that also succeeded in reducing the size of the problem. The transformation of the original problem to a NETFORM will not be shown because the mathematics are somewhat intricate and the original formulation by itself consumes more than twenty pages of [8]. However, making use of the NETFORM, we were able to develop a specialized solution process employing network optimization procedures we had previously developed for subproblem relaxations [5], and solved four versions of this problem with data supplied by the TVA. The first three versions, while requiring half an hour to two hours to solve with standard procedures, were easily solved in seconds to minutes using the NETFORM representation and the specialized solution approach. The fourth version was by far the most difficult, involving 173 constraints, 126 zero-one variables, and 511 continuous variables. The original mixed IP formulation was run for eight hours on an IBM 370/168 using MPSX, and then taken off the machine to avoid fur-

Figure 3

**Netform Representation**



ther computer run costs. At the end of this time the best (minimum cost) solution obtained had an objective function value of $136,173,440. With a time limit of 30 minutes imposed on the NETFORM solution effort, a solution was obtained that had an objective function value of $125,174,727, which constitutes more than a $10,000,000 improvement. Consequently, this application shows that the use of NETFORM representations can provide improved solutions for problems too complex to be solved optimally (within practical time limits) by standard approaches.

ACKNOWLEDGEMENTS

This research was partly supported by ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute and by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas.

REFERENCES

1. Analysis, Research, and Computation, Inc. Development and computational testing on large-scale primal simplex network codes. ARC Technical Research Report, P.O. Box 4067, Austin, Texas 78765 (1974).

2. Barr, R. S., Glover, F., and Klingman, D. Enhancements of spanning tree labeling procedures for network optimization. CCS Report 262, Center for Cybernetic Studies, University of Texas, Austin, Texas 78752 (1976).

3. Charnes, A., Glover, F., Karney, D., Klingman, D., and Stutz, J. Past, present, and future of development, computational efficiency, and practical use of large-scale transportation and transshipment computer codes. *Computers and Operations Research* 2, 2 (1975).

4. Glover, F. and Klingman, D. Real world applications of network related problems and breakthroughs in solving them efficiently. *ACM Transactions on Mathematical Software* 1, 1 (1975), 47-55.

5. Glover, F., Karney, D., and Klingman, D. Implementation and computational study on start procedures and basic change criteria for a primal network code. *Networks* 4, 3 (1974), 191-212.

6. Glover, F., Klingman, D., and Stutz, J. Implementation and computational study of a generalized network code. 44th National Meeting of ORSA, San Diego, California (1973).

7. Glover, F. and Mulvey, J. Equivalence of the zero-one integer program to discrete generalized and pure networks. *MSRS* 75-19, University of Colorado (1975).

8. Kazmersky, P. A computer code for refueling and energy scheduling containing an evaluation of nuclear decisions for operation. Unpublished Ph.D., Ohio State University (1974).

9. Ross, G. T. and Soland, R. A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming* 8, 1 (1975), 91-104.