ALGORITHMS FOR THE HAND-COMPUTATION SOLUTION OF THE
TRANSHIPMENT PROBLEM AND MAXIMUM FLOW
IN A RESTRICTED NETWORK

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By

EDWARD A. BROWN, B. S., M. S.

The Ohio State University
1959

Approved by

Paul N. Lehoczky
Adviser
Department of Industrial Engineering
PREFACE

The task of giving credit where it is due, as usual, is hopeless. I can only mention here a few of the many who have contributed to this work. A special note of thanks is due to Doctor Paul N. Lehoczky, my adviser, for providing the encouragement, assistance, and proper atmosphere required for the completion of this work. Without his advice and guidance, the work might not yet be finished.

Doctor Paul M. Pepper gave generously of his time and assistance. He undertook the reading of the original manuscript in great detail and, with the editorial pen of a mathematician, has corrected many errors and points of confusion in the original manuscript. He has also made many suggestions and several important contributions which are included in the work as it now appears. To him must go the credit for devising the method of determining whether or not the maximum flow in a network is finite, and for generalizing the procedures for the transhipment problem to a network with "handling costs" at the nodes themselves.

To Doctors David Baker, Albert Bishop, Daniel Howland, and William Morris—thank you for lending me your time and your ear. While they are excused from responsibility for any mistakes that may still exist in the work, they have all been responsible for some correct parts that do appear. They have all helped with the many hours of discussion and interest (and reading of the manuscript) that are a part of a work such as this.
Doctor Loring G. Kitten deserves a special measure of credit, because many of the ideas developed in this work are his. While he was at Ohio State, he freely shared with me his thoughts on what should and could be done in programming areas. He had been attempting to develop a generalized algorithm (like the Transhipment Algorithm contained in this work), but neither of us knew how to solve the Maximum Flow Sub-Routine. In fact, on my General Examinations, he asked me how to find the maximum flow in a restricted network—I did not know how at that time either. I undertook the present work with his blessing.

While writing this, as always, foremost in my mind are thoughts of my wife Eileen (and our four children—the fifth is due with the completion of this work). More important than her assistance with the actual preparation of this work, was her ability to pacify the "gang" when Daddy could not play, and to pacify him when he could. Finishing this on our Seventh Wedding Anniversary makes it a suitable present to my family.

Edward A. Brown
2 February 1959
TABLE OF CONTENTS

CHAPTER I. INTRODUCTION . 1

- Linear programming origins . 3
- Military sponsorship . 3
- Simplex limitations . 6
- Special computational procedures . 8
- Primal-dual methods . 10
- Summary . 15

CHAPTER II. THE FLOW ALGORITHM . 18

- The network . 18
- The augmented network . 20
- Flows in the network . 23
- Mandatory assignments . 25
- Feasibility considerations . 26
- Formal statement of the flow problem . 28
- The set-labeling procedure . 31
- Termination of set-labeling . 32
- The max-flow min-cut theorem . 34
- Algebraic derivation of maximum flow . 36

CHAPTER III. THE TRANSHIPMENT PROBLEM . 40

- The problem . 40
- Feasibility considerations . 44
- Conventions . 47
- Formal statement of the problem . 48
- Minimizing the primal . 49
- The dual problem . 53
- The algorithm . 56
- The starting procedure . 59
- The iterative procedure . 61
- The sub-routine . 65
- Increase in the dual value . 66
- Summary . 67

CHAPTER IV. THE TRANSHIPMENT ALGORITHM . 68

- A transhipment problem . 69
- The transhipment algorithm . 73
- Example solution . 75

CHAPTER V. CONCLUSION . 108

- Generalizations . 108
- Other problems . 109
- Summary . 111

BIBLIOGRAPHY . 112

AUTOBIOGRAPHY . 120
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>FIGURE DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Directed arcs</td>
<td>19</td>
</tr>
<tr>
<td>2.</td>
<td>Reduction of the network</td>
<td>21</td>
</tr>
<tr>
<td>3.</td>
<td>Sample restriction matrix</td>
<td>22</td>
</tr>
<tr>
<td>4.</td>
<td>Sample cost matrix</td>
<td>42</td>
</tr>
<tr>
<td>5.</td>
<td>A feasible network</td>
<td>44</td>
</tr>
<tr>
<td>6.</td>
<td>Example transhipment problem</td>
<td>70</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

The Transhipment Problem is a generalization of a class of Linear Programming problems which have received much attention and wide application during the past several years. The transhipment problem concerns the shipment of "goods" from the input nodes of a network, through the network to the output nodes, at a minimum of cost. The input or output quantity is specified for each node of the network. And for each pair of nodes, both the cost of shipping a unit of the goods along the arc connecting the two nodes and the capacity restriction which places an upper bound on the amount of flow permitted in the arc are given. The solution of the problem consists in finding the value of the flow in each arc of the network such that the overall shipment through the network is made at minimum cost.

From the above description it can be seen that what is usually referred to as The Transportation Problem is a special case of the transhipment problem—that is, one in which there are no intermediate or transport nodes, and shipping is made directly from the input nodes to the output nodes. And what is usually referred to as The Assignment Problem is a further specialization of the transportation problem—not only are there no intermediate or transport nodes, but in addition the input and output quantities are all unity, that is, there is only one item of supply at each of the input nodes to be assigned to satisfy the unit requirement at each of the output nodes.
The Maximum Flow Problem arises naturally with each of the problems mentioned above. It is embedded in each of these problems, and a solution to the maximum flow problem answers the feasibility question of whether any flow, regardless of cost considerations, can be made to ship the input quantities through the network to the output nodes. The given information consists only of the input or output quantity at each node and the upper bound restriction on the flow in each arc of the network. The solution for the problem consists in finding the values of the flows in the arcs, such that the input quantities are shipped through the network to the output nodes, without regard to cost considerations.

In both the transhipment problem and the maximum flow problem, it is desired to maximize or minimize (or, better still, optimize) a linear form or functional, subject to linear inequality restraints. For example, in the transhipment problem, it is desired to minimize the expression for the total cost, subject to the inequality condition that the flow in any arc is "equal to or less than" the upper bound restriction on the flow in that arc. The field of mathematics which has most successfully dealt with the solution of such problems has come to be known as Linear Programming.

To properly place the history and development of the transhipment and flow problems, it is necessary to outline the development of linear programming. However, no attempt will be made to give a comprehensive history because this would be a major undertaking in itself, and because excellent outlines of the history, and significant publications, are already available in Koopmans [71], Dantzig [15], and Rohde [91], and in several of the other works listed in the bibliography.
Linear programming origins. Although linear programming can be traced back to the middle 1700's when economists first tried to describe economic systems in mathematical terms, little was done to exploit the linear-type models for almost two hundred years, and the effort during this time was of a qualitative, descriptive-economics nature.

The first quantitative effort to construct practical mathematical models of economic systems did not come until the nineteen thirties when Leontief [77] [78], with the stimulus of the great depression years, studied the basic structure of the American economy and formulated his inter-industry input-output models. During this same period, a general linear programming model of production and economic equilibrium was formulated by von Neumann [83]. Still the effort was devoted almost entirely to the development of normative economics, that is, how best to allocate limited means to accomplish desired ends. And the work was carried out mostly by economists and businessmen who did not fully appreciate the potential of the linear programming model because "... to many economists, the term linearity is associated with narrowness, restrictiveness, and inflexibility of hypothesis" [71, p 6]. However, it was these economic theories which stimulated the development of suitable mathematical models and methods of solution, like Dantzig's "objective function" [25], which was to become the basis of the present-day Simplex method of computation for linear programming problems.

Military sponsorship. During World War II, many practical but more detailed applications of the allocation and programming models arose in connection with the mobilization of the United States and its allies, and the utilization of the military forces. While working for the
British-American Combined Shipping Adjustment Board dealing with merchant shipping problems during the war, Koopmans published his model of transportation [72], in ignorance of an earlier formulation that was made by Hitchcock [64]. This Hitchcock-Koopmans Transportation Problem has become the most-spoken-of linear programming problem and has been the most successful application of linear programming techniques to the problems of management and industry. (The problem of interest in the present work is a generalization of this transportation model, from one consisting only of input and output nodes to one consisting of shipment from input nodes, through a network of transport nodes to output nodes.)

Because of the applicability and success of the linear programming approach to military planning, The United States Air Force sponsored further work at The RAND Corporation (for Research AND Development) in Santa Monica, California. At RAND a group headed by George B. Dantzig, who has been the major contributor to the field of linear programming, developed programming models to deal with the scheduling of the interdependent activities of a large organization, and with the choice of a best combination of activities toward the achievement of a stated goal.

At this time, both the economists and the mathematicians recognized the tremendous potential of linear programming models for studying economic systems. However, there had been only limited success with the necessary mathematical techniques for solving the models. Dantzig's simplex technique was the accepted solution procedure. (With the corrections, modifications, and generalizations that have since been made to the original method, it is still the most general of the solution procedures in use today.)
By 1949 there was a considerable body of literature from various economic, philosophical, and mathematical points of view. The more significant contributions were presented at a Conference of the Cowles Commission for Research in Economics, held in Chicago on 20-24 June 1949. Koopmans' edition [71] of the proceedings of this conference remains the main source book of information and references on linear programming from its origins up to 1950. The conference stimulated interest in both the models themselves and in mathematical techniques of solution, and the number of publications in the field of linear programming began to grow by leaps and bounds.

Encouraged by the progress (and volume) of the work being done at RAND, The Air Force sponsored The First [60] and Second [2] "SCOOP" Symposiums in Linear Programming in 1952 and 1955. (Good bibliographies are provided in the proceedings of both of these symposiums.) By 1953, the mathematical solution techniques had been refined sufficiently to enable Charnes, Cooper, and Henderson [10] to publish a 74 page "text" on linear programming and the simplex technique of solution.

In 1956, there were so many publications concerned with linear programming that Rohde [91] was able to publish a somewhat comprehensive bibliography consisting of 266 articles (not including closely related topics on game theory or pure mathematics). In this work, Rohde traces the significant publications in special problem areas and in general linear programming. No sooner had Rohde's bibliography appeared than Wagner [100] published a supplemental bibliography of 193 articles that had not been included by Rohde. Publications and developments since that time have continued at a brisk rate.
**Simplex limitations.** Soon after the Cowles Commission Conference in 1949, there were many attempts to solve linear programming problems, using, for the most part, the simplex technique proposed by Dantzig. Dantzig [12] applied the simplex procedure to a Hitchcock-Koopmans transportation problem [64] [72] [73]. Then in 1954, Flood [41] used the simplex computation on the problem of scheduling a military tanker fleet. Dantzig and Fulkerson [32] also solved a large-scale fixed-schedule transportation problem with the simplex technique. Work on such problems was continually in progress at RAND, and with the experience gained, Dantzig, Orden, and Wolfe [36] were able to make necessary adjustments in the procedure and devise a more general simplex method.

In general, the simplex technique was a direct attack on the primal problem itself, that is, an attempt to evaluate the unknowns directly, and it was primarily useful when programmed for machine computation. Aside from the high machine cost, there were many technical difficulties with the procedure, for instance, cases of degeneracy and cycling where the solution procedure would not work; and the solution procedure did not guarantee an improvement in the solution at each cycle of application. But these were not the type difficulties with which we are presently concerned. The more serious limitations of the simplex technique were the practical ones that became apparent when the attempt was made to solve the large-scale problems. In October 1954, before the Institute of Management Sciences in Pittsburgh, Dantzig, talking about the status of solution of large-scale problems, said that "... the pessimistic note ... concerns the inability of the problem solver to compute models by general techniques [simplex] when they are large scale" [29].
Some success had already been achieved with the use of special computational algorithms for the assignment and transportation problems, and Dantzig said: "Often these new methods are powerful enough to do by hand what cannot be achieved by machine... A transportation problem of the Hitchcock-Koopmans variety involving, say, a hundred rows and columns combined or one involving 10 rows and hundreds of columns can be nicely handled by clerks" [29, p 6]. In 1955, Dantzig pointed out one reason why the special hand-computational procedures promised success: "The human mind seems to have a remarkable facility for scanning many combinations and arriving at what appears to be either a best one or a very good one..." [17, p 267]. And in 1956, Dantzig made "... a short plea that linear programmers pay greater attention to special methods for solving the larger matrices that are encountered in practice..." [26, p 139]. And after describing the capabilities of the IBM computer codes for the simplex technique, he says: "However, it is self-evident that no matter how much the general purpose codes are perfected, they will be unable to solve the next generation of problems which will be larger in size" [26, p 142].

Because of the practical difficulties which had been experienced with the general simplex technique, and also because of the high cost of machine computation time, the demand for special hand-computation algorithms has continued to the present. Some very successful algorithms have been developed for the assignment and the transportation problems. But the demand continues for more and more general algorithms which will handle larger scale systems and problems. (The present work is an attempt to provide such an algorithm for the transhipment problem.)
Special computational procedures. In response to the demand, several specialized computational procedures have been suggested, all the way from analogue solutions [1] to graphical solutions [67]. However, the special procedures of interest here, which have provided the background for the present undertaking, are the development of combinatorial type algorithms, the use of the extended Lagrange multiplier technique, primal-dual procedures, and the solution of the maximum flow problem.

The first of the special computational techniques that was widely accepted was an algorithm proposed by Kuhn [74] [75] called "The Hungarian Method for the Assignment Problem." Kuhn treated the assignment problem as a combinatorial problem in which it was desired to pick that assignment of men to jobs, out of all the possible assignments, that gave the optimal total score (or cost) for all the assignments. In this approach is to be found the beginning of several other successful algorithms. It was soon demonstrated that the transportation problem could be reduced to a combinatorial problem by considering that each of the input quantities was a unit input and had to be assigned to satisfy one unit output requirement. The extension of Kuhn's Hungarian method to the transportation problem can be found in the work of Ford and Fulkerson [49], and again in the work of Munkres [82]. Munkres' algorithms for the assignment problem and for the transportation problem were well received and widely used. In 1958, Professor Loring G. Mitten taught Munkres' algorithms in a programming course at The Ohio State University [81]. It was while studying under Mitten that the author first became aware of the need for a more general, and yet more simple algorithm than the ones proposed by Munkres.
In 1956, Dantzig [26] pointed out that because of the success with the combinatorial approach to the assignment and transportation problem, it seemed to be worth while to try to find further applications of such methods. He pointed out that combinatorial applications depend on the integral character of the basic solutions and said: "there is a combinatorial aspect contained in every linear programming problem. Indeed, the basic problem is one of selecting from the class of extreme points of a polyhedral convex the one which maximizes a given linear form. The fact that there are procedures ... which are fairly efficient in selecting such combinations is the reason why ... [they are] tried for certain combinatorial problems. Indeed, it is just those problems where the extreme points of a convex can be identified with the combinations of interest where this approach has paid off" [26, p 138].

Houthakker [65] showed how Kuhn's method for solving problems of the transportation type could be made considerably more efficient by getting an initial solution of assignments made on a "mutually preferred" basis, that is, when a cost figure is the lowest in both its column and row, it is likely that this assignment will appear in the overall optimal assignment. After such assignments are made and an approximate solution is achieved, the optimal solution can more quickly be found by the applications of the algorithm. (Such a procedure does not seem to be applicable to transhipment problems in the more general networks. This is because it is not such a simple matter to pick out a "mutually preferred" path through the complete network as it is to pick out a mutually preferred assignment being made directly from an input node to an output node.)
Also important to the present development is the suggestion to extend the Lagrange multiplier technique to optimizing a function subject to inequality constraints. Some early work had been done at RAND on the use of such techniques, and there is a recent article by Tucker [94] on the subject; however, the more important reference was by Klein [70] on the direct use of the Lagrange technique for optima constrained by inequalities. In Klein's proposal, the inequalities are transformed to equalities by adding "slack" variables, and then the classical technique is applied directly. (This approach is used in the present work for finding the conditions required of an optimal solution of the transhipment problem.) Dantzig [21] pointed out that difficulty could arise in using this technique when the original problem is not feasible, for example, if it is desired to minimize a cost function and there is no finite lower bound on this function. However, no practical difficulty will arise in the present work, because, as will be shown later in a section on feasibility of the transhipment problem, every transhipment problem can be expressed in a form which does have a finite lower bound on the cost function. In addition, even if the original statement of the problem is not extended to ensure feasibility, a "really" optimal solution will always result from the application of the algorithm even when the original problem was not mathematically feasible.

Primal-dual methods. Most important to the present work, was the development of primal-dual methods for solving linear programming problems by way of the computation of maximum flows in networks. As indicated earlier, in every problem of the transhipment type there is embedded a maximum network flow problem. In the primal-dual method, the
primal problem is solved indirectly by means of the dual of the original problem. Starting from an initial feasible solution of the dual, and on each subsequent cycle of the solution procedure, a partial-primal maximum flow problem is solved to indicate what must next be done in the dual problem to make a closer approach to the optimum solution. In the present work, the transhipment problem will be solved indirectly by solving the dual of the original transhipment problem. At each step of the solution procedure, the maximum flow in part of the original network will be determined, and the solution of this partial-network maximum flow problem will indicate how to further improve the dual solution.

The publication which led to a suitable computational procedure for finding the maximum flow in a restricted network was made in 1954 by Ford and Fulkerson [46]. This work contained a proof of their "Max-Flow Min-Cut" Theorem which has been the basis for much additional work. The problem, originally formulated by Harris at RAND, is as follows: Find the maximum flow from the origin node to the terminal node in a network connecting the two* nodes, where each arc has an upper bound capacity restriction on flow in the arc. Although this problem could be set up as a general linear programming problem with as many equations as there are nodes in the network, and therefore could be solved by the simplex technique, Ford and Fulkerson felt "... it turns out that in the cases of most practical interest ... a much simpler and more efficient hand computing procedure can be described" [46, p 1].

---

*The case where there are several input and output nodes can always be reduced to the case where there is a single input and a single output node. This will be shown in the chapter on The Flow Algorithm.
A more detailed development of the max-flow min-cut theorem will be given later. However, for the present, the following general description should suffice. When flows have been made through a restricted network and a maximum flow has been obtained, there will be no more paths for flow from the origin node to the terminal node because certain of the arcs of the network will have become saturated with flow. In this event, the origin has been "cut off" from the terminal by the saturated arcs which form this "cut." Since the maximum flow must pass completely through these saturated arcs of the cut, the amount of the flow must be equal to the sum of the capacity restrictions on the saturated arcs in the cut.

The proof given by Ford and Fulkerson was a non-constructive and indirect proof based on convexity arguments. However, this proof did enable the authors to propose a computational scheme which was completely independent of the simplex technique. However, we will not describe their computational procedure because it is somewhat cumbersome and awkward for our present purposes. A simpler and more suitable procedure will be suggested in the development that follows.

By the beginning of 1955, Dantzig and Fulkerson [33] had a better developed approach to the max-flow min-cut theorem and its proof, and they generalized the theorem to include capacity restrictions on the nodes themselves. (In the present work, to make the overall explanation clearer, we are restricting ourselves to consideration of the problem with capacity limits only on the arcs of the network. In the final chapter, it will be suggested that the present methods be generalized to include both node capacities and flow in both directions in the arcs.)
By the end of 1955, Fulkerson and Dantzig [56] published a direct proof of the max-flow min-cut theorem based on the simplex algorithm of linear programming. They also proposed a computational procedure, but once again the procedure is not convenient for the hand-computational algorithms to be developed in the present work.

In May 1955, Robacker [89] presented a rigorous mathematical treatment of the theory of networks and of the max-flow min-cut theorem. This work attempted to establish a standard terminology for work on network flows. At about the same time, Ellis [38] proved that in any network consisting of two or more nodes, the network could be constructed to be planar and irreducible with respect to its origin and terminal. This made all of the literature that was already available on planar graphs applicable to network flows. This was also important because some of the earlier work on network flow theory was based on the assumption that the networks were planar.

Also during 1955, Boldyreff [5] published a paper on a "flooding" technique for determining the maximum flow in a network. This was a trial-and-error method which only worked in special cases. Although this solution procedure itself is not of much interest as far as the present development is concerned, Boldyreff did show numerous ways of simplifying a network which are useful with all methods of solution. In this simplification process, or method of "successive reductions," he shows how to eliminate series connections by arc absorption, and the elimination of parallel arcs, and so on. This reference might well be consulted before work on any actual large-scale problem is begun in order to ensure that the problem is undertaken in its simplest form.
In December 1955, Ford and Fulkerson [49] applied the flow computation procedure as a sub-routine in solving the Hitchcock-Koopmans transportation problem not involving capacity restrictions on the flow. The authors note that "... this is of course a linear programming problem, and hence may be solved by Dantzig's simplex algorithm. In fact, the simplex computation for a problem of this kind is particularly efficient... However, for the flow problem, we shall describe what appears to be a considerably more efficient algorithm ... readily learned by a person with no special training, and [one that] may easily be mechanized for handling large networks. We believe that problems involving more than 500 nodes and 4,000 arcs are within reach of present [1955] machines" [49, p 3]. In 1956, Ford and Fulkerson [48] [50] generalized the procedure to the solution of the transportation problem with capacity restrictions on the flows. Once again commenting on the efficiency of these special techniques, the authors state that the procedure "has been compared with the simplex method on a number of randomly chosen problems and has been found to take roughly half the effort for small problems. We believe that, as the size of the problem increases, the advantages of the present method become even more marked" [50, p 1].

Recognizing the advantages of the primal-dual approach to the solution of linear programming problems, the methods developed by Ford and Fulkerson were extended to the general linear programming case in 1956 by Dantzig, Ford, and Fulkerson [31]. In this generalized procedure, the primal problem is used as a sub-routine in optimizing the dual of the problem, while the restricted primal problem itself is solved by the simplex technique.
Orden [85] published a paper in 1956 pointing out that the transshipment problem (the one with which we are presently concerned) can be converted into a standard transportation problem and then solved by Dantzig's simplex technique. Orden says: "The transshipment problem will be solved by converting it in a specified way to a computation problem which has the form of [a transportation problem]... After conversion ... Dantzig's simplex technique provides a satisfactory computation technique..." [85, p 278]. Finally, in 1958, Fulkerson [53] proposed that the flow computation procedure be used to solve the transshipment problem by primal-dual methods. (And such a solution to the transshipment problem is the purpose of the present work.)

**Summary.** Both linear programming, in general, and the transportation type models, in particular, are important tools in management science, operations research, and systems study. Although the central mathematical problem is to solve the linear inequality systems, more important is the fact that "... linear programming is a technique for building a model for describing the interrelations of the components of a system. As such it is probably the simplest mathematical model that can be constructed of any value for broad programming problems of industry and government. Thus the importance of the linear programming model is that it has wide applicability" [18, p 1].

Smith [93], reporting on a panel discussion on the current use of linear programming in American business, said that, of the applications which had made any apparent contribution to managerial decisions "...the largest number of cases were transportation problems in both senses. That is, they were concerned with the most economical [optimal] way of
transporting materials between various sources and destinations and were solved by one of the special algorithms for the 'transportation' problem. ... Considerable planning for further applications of this kind was reported by various members of the panel [93, p 156]. And because of the use of the special hand-computation procedures, Smith continues, "... in several cases this has meant that manual computation has sufficed on problems where a computer would have been required in order to use a general algorithm [like simplex]" [93, p 157].

From what has been said already, it should be clear that there is a definite requirement for simple, and yet general, hand-computation procedures for classes of linear programming problems. The intention in the present work is to develop hand-computation algorithms for the problem of determining the maximum flow in a restricted network, and for the solution of the transshipment problem in a restricted network. Although both of these problems can be solved by the very general simplex method, and even though the flow problem can be solved by the flow algorithm of Ford and Fulkerson—a procedure which appears to be unnecessarily complicated for hand-computation applications—it is felt that the attempt to produce the simpler and more general algorithms presented herein is justified on several counts. The objectives in developing these algorithms were to produce a procedure that could solve even complex network problems quickly and efficiently without being dependent on the use of high-speed computers or other specialized equipment, and that the method be simple enough to be explained to relatively untrained personnel with a few hours of explanation. In this way, it is hoped that the solution of the class of transshipment problems will be possible in the smaller
research and academic groups where the expense of computer time might otherwise prohibit an attempt to solve the problems. The transhipment algorithm to be presented will, naturally, solve the special cases of the assignment problem and the transportation problem. The use of the proposed transhipment algorithm for all of these problems should reduce the time required to teach a course in programming techniques from a full academic quarter to one or two academic weeks.

Finally, it is hoped that the present work finds some merit, if not for the generality of the problems which it will handle, then for the very simplicity of the procedures. The derivations and proofs that will be given are at times difficult, and at worst tedious. In working through the development in Chapters II and III, there is much detail to be noted, and an effort must be expended to keep track of the meaning of the single and double subscripts on symbols. However, the algorithms themselves are easy to follow (after working through the sample problem) and the solution procedure itself is carried out with only addition and subtraction.

The Flow Algorithm for finding the maximum flow in a restricted network (without regard to cost considerations) will be developed in Chapter II. The Transhipment Problem and its solution algorithm will be developed in Chapter III. In Chapter IV, The Transhipment Algorithm will be stated informally (with a minimum of mathematics), and a sample problem will be completely worked out to illustrate the procedure. In Chapter V, some suggestions will be given for generalizing the solution procedures, and for further work required in these areas.
CHAPTER II

THE FLOW ALGORITHM

In this chapter an algorithm is developed which is suitable for the hand-computation of a maximum flow \( \lambda = \max \) in a network with non-negative, integral capacity restrictions \( r = \left\{ r_{ij} \right\} \) on the flows \( x = \left\{ x_{ij} \right\} \) in the arcs \( ij \) of the network. And to specialize the procedure for use as a sub-routine in the solution of the transhipment problem, mandatory flows \( x_{ij} = r_{ij} \) will be specified for certain of the arcs.

The network. Consider a network consisting of a set \( N \) of \( n \) nodes,

\[
N = [1, 2, \ldots, n].
\]  

(1)

The \( n \) nodes consist of three mutually exclusive subsets: \( N_1 \) of \( n_1 \) "input" nodes; \( N_2 \) of \( n_2 \) "transport" or conservative nodes; and \( N_3 \) of \( n_3 \) "output" nodes.

\[
N_1 = [1, 2, \ldots, n_1],
\]

(2)

\[
N_2 = [n_1+1, \ldots, n_1+n_2],
\]

(3)

and

\[
N_3 = [n_1+n_2+1, \ldots, n],
\]

(4)

where

\[
n = n_1+n_2+n_3,
\]

(5)

and

\[
N = N_1 \cup N_2 \cup N_3.
\]

(6)

From the definitions (1) through (6) it can be seen that, without loss of generality, we have numbered the nodes so that the lower numbers are designations for the input nodes, the higher numbers are for the output nodes, and the transport nodes are designated in between.

Associated with certain pairs \( ij \) of the nodes, where \( i \in N_1 \) and \( j \in N_3 \), there is specified a non-negative capacity restriction \( r_{ij} \) which places
an upper bound on the arc flow $x_{ij}$ in the arc from $i$ to $j$. If $r_{ij} = 0$, then no real flow is permitted in the network along the arc from $i$ to $j$. In the solution procedure, we will not make zero entries in the original restriction matrix at the positions where $r_{ij} = 0$. Therefore, we adopt the convention of only making entries in the original matrix of capacity restrictions at those positions where

$$C < r_{ij} \leq \infty.$$  \hspace{1cm} (7)

It should be noted that we are developing a solution procedure for a directed network in which the real flow in any arc can be in only one direction. Although it is not significantly more difficult to develop the solution procedure for an undirected network, for clarity in what follows, we will reduce the undirected network to a directed network by reducing each undirected arc. Dantzig and Fulkerson [33] give a method for making this reduction; however, for our procedure, the following reduction is more suitable. Where the original $r_{ij} > 0$ and $r_{ji} > 0$, let the flow from $j$ to $i$ be made through a fictitious node $k$, as indicated in Figure 1, with $r_{jk} = r_{ki} = r_{ji} > 0$. Now the entries at $ij$ and $ji$ will always be such that

$$\text{if } r_{ij} > 0, \text{ then } r_{ji} = 0.$$  \hspace{1cm} (8)

*In the solution procedure, it will be permitted to make positive entries even at those positions where the original restriction is zero. Such entries will be treated as fictitious capacity restrictions, because they arise when a backward flow is made through the network.
We are interested in making a flow or shipment of some resource or item from the input nodes, through the network, to the output nodes where there is a requirement for these resources. Therefore, at each of the nodes icN, there is a given quantity $r_i$ which is the total "input" restriction on the flow into the real network from outside of the real network. Naturally, since the $r_i$ are input quantity restrictions, the $r_i$ are positive for the input nodes, zero for the transport nodes, and negative for the output nodes. That is,

$$r_i > 0, \text{ for } icN_1, \quad (9a)$$

$$r_i = 0, \text{ for } icN_2, \quad (9b)$$

and

$$r_i < 0, \text{ for } icN_3. \quad (9c)$$

**The augmented network.** It can be seen from Figure 2 that the problem of finding the maximum flow through the network is not in any way changed by the device of reducing the network with several input and output nodes to a network with a single input and a single output node. In this device, a fictitious origin node 0 is connected to each of the input nodes by an arc of capacity restriction equal to the input quantity $r_i$ for the node; and each of the output nodes is connected to the fictitious terminal node t by an arc of capacity restriction equal to the output quantity $-r_i$ for the node.

Backward flow from the input nodes icN_1 to the origin 0, or from the terminal t to the output nodes icN_3, is never permitted. In the augmented network, then, we have the following additional restrictions:

$$r_{i0} = r_{ti} = 0, \text{ for } icN, \quad (10)$$

$$r_{0j} = r_{j}, \text{ for } icN_1, \quad (11)$$

and

$$r_{it} = -r_i, \text{ for } icN_3. \quad (12)$$
In Figure 2, the connections that are completely within the real network, that is, those for which \( r_{ij} > 0 \), for \( i \in \mathcal{N} \) and \( j \in \mathcal{N} \) (and \( i \neq j \)), are omitted from the drawing—they are understood to lie completely within the box which represents the real network. Only the arcs from the origin 0 to the input nodes \( \mathcal{N}_1 \), and from the output nodes \( \mathcal{N}_3 \) to the terminal \( t \), are shown on the drawing. The capacity restrictions on these arcs are given by the relations (9), (10), (11), and (12).

![Diagram](image)

Fig. 2.—Reduction of the network.

In the actual computation of a maximum network flow according to the Flow Algorithm to be developed, it is suggested that an \((n+1)\) by \((n+1)\) matrix or array be laid out similar to the sample restriction matrix shown in Figure 3. (In the sample of Figure 3, \( N_1 = [1, 2] \), \( N_2 = [3, 4, 5] \), and \( N_3 = [6, 7] \). These node sets are the ones which will be used in the example problem to be given in Chapter IV.) The rows are labeled 0, 1, 2, ..., \( n \), and the columns are labeled instead by \( t, 1, 2, ..., n \). To prevent making an entry at the position \( ij \) where \( i = j \), the main diagonal of the matrix should be X'd out. Also since the only flows from 0 are to the input nodes, the other positions in row 0 are X'd out; similarly, since the only flows to \( t \) are from the output nodes, the other positions in column \( t \) are X'd out.
Directions: In the original restriction matrix $r = \begin{bmatrix} r_{1} & r_{2} & \cdots & \cdots \\ r_{3} & r_{4} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ r_{n} & r_{n+1} & \cdots & r_{2n} \end{bmatrix}$, entries are made at all those positions where $r_{ij}$ is greater than zero. Note that, if $r_{ij} > 0$, then $r_{ij} = 0$, and no entry need be made at $ji$. Also in the original restriction matrix, the "input" restrictions $r_i$ for $icN_1$ are entered in row 0, and the "output" restrictions $-r_i$ for $icN_2$ are entered in column $t$.

In the reduced capacity restriction matrices $r' = \begin{bmatrix} r'_{1} & r'_{2} & \cdots & \cdots \\ r'_{3} & r'_{4} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ r'_{n} & r'_{n+1} & \cdots & r'_{2n} \end{bmatrix}$ in which the maximum flow computation is carried out, it will be permitted to make positive entries at the position $ji$ (according to the directions of the solution procedure to be developed), even when no entry was made at the position $ji$ in the original restriction matrix.

In all cases, when no entry appears at the position $ij$, it is to be understood that the capacity for flow through the arc $ij$ is zero.

Fig. 3.—Sample restriction matrix.
In the computations for determining the maximum flow in a restricted network, two matrices like the one shown in Figure 3 should be used. In the first, called the original restriction matrix \( r = [r_{ij}] \), all the information which is given in the problem can be entered and saved until the computation procedure is completed. The input and output quantities, \( r_i \) for \( i \in \mathcal{N}_1 \) and \( -r_i \) for \( i \in \mathcal{N}_3 \), can be entered in their proper places in row 0 and column \( t \). The entries \( r_{ij} \geq 0 \) are made in those positions where flow in the arc from \( i \) to \( j \) is permitted. All entries in the original restriction matrix must satisfy (7) through (12).

The second matrix is called a reduced restriction matrix (stages of the solution indicated by superscripts) \( r' = [r'_{ij}] \), because flows are made in this matrix by reducing the capacity restrictions for flow. (Entries are quickly erased and changed as required by the procedure.)

**Flows in the network.** The solution to the problem is carried out in the second restriction matrix by finding a matrix \( x = [x_{ij}] \) of arc flows \( x_{ij} \), such that the total flow \( f \) from origin 0 to the terminal \( t \) is a maximum. The flow \( f \) through the network is always given by the total flow from 0 to the input nodes, or from the output nodes to \( t \).

\[
f = \sum_{j \in \mathcal{N}_1} x_{0j} = \sum_{j \in \mathcal{N}_1} x_{0j} = \sum_{i \in \mathcal{N}_3} x_{it} = \sum_{i \in \mathcal{N}_3} x_{it}
\]

(13)

The flow is found by an iterative procedure of locating paths (formally called chains) along which partial trial flows \( \Delta f \) may be made from 0 to \( t \). When a path from 0 to \( t \) has been located and a flow \( \Delta f \) is made, the restrictions \( r'_{ij} \) are reduced in the forward direction of flow by the amount \( \Delta f \), and increased by \( \Delta f \) in the backward direction of flow. The procedure is repeated in the reduced restriction matrix \( r' = [r'_{ij}] \).
until no more paths with positive capacity for flow from 0 to \( t \) exist, and no further increase in the total flow is possible. At termination, the flows \( x_{ij} \) will be given by the relation

\[
x_{ij} = \max \{ r_{ij} - r_{ij}', 0 \}.
\]

(14)

When the computation has ended and no further increase \( \Delta f \) in the total flow is possible, it will be shown that (13) will give the value of the maximum flow \( f_{\text{max}} \), where \( r_{0j} = r_j \) for \( j \in N_1 \), and \( r_{0j}' \) is the entry in row 0 at \( j \in N_1 \) in the final reduced restriction matrix.

There are two restrictions on the flows in the real network. The first is that the flows in the arcs be non-negative and not greater than the restrictions for the arcs, that is,

\[
0 \leq x_{ij} \leq r_{ij}, \quad \text{for } i \in N, j \in N.
\]

(15)

The second restriction is that the sum of the "input" quantity to node \( i \) from outside the real network plus the flow to node \( i \) within the real network, must equal the flow from \( i \) to all other nodes of the network,

\[
r_i + \sum_{j} x_{ji} = \sum_{j} x_{ij}, \quad \text{for } i \in N, j \in N,
\]

or

\[
\sum_{j} (x_{ij} - x_{ji}) = r_i, \quad \text{for } i \in N, j \in N.
\]

(16)

If we consider the augmented network of Figure 2, we can combine the restrictions (15) and (16) on \( x_{ij} \) to the following *

\[
0 \leq x_{ij} \leq r_{ij}, \quad \text{for } i,j \in [0 \cup N \cup t],
\]

(17)

\[
\sum_{j} (x_{ij} - x_{ji}) = 0, \quad \text{for } i \in N, j \in [0 \cup N \cup t].
\]

(18)

*The indexes \( i \) and \( j \) normally take values in the range of the set \( N \) of nodes. Whenever \( i \) or \( j \) is permitted to take on the additional values 0 or \( t \) of the augmented network, the range of \( i \) and \( j \) will be stated to clearly include the values 0 or \( t \)—otherwise, \( i \) and \( j \) can not take the values 0 or \( t \). Also, throughout the text, when a set of nodes, say \( N \) or \( S \), is augmented by the addition of the node set [0] or [\( t \)], the augmented node set will simply be designated by [0 \cup N \cup t], say, instead of [0] \cup N \cup [t]. No confusion should arise when the symbol 0 or \( t \) is used for the set [0] or [\( t \)].
\[ \sum_j (x_{0j} - x_{j0}) = f, \quad \text{for } i = 0, j \in \{0 \cup N \cup t\}, \quad (19) \]

and

\[ \sum_j (x_{tj} - x_{jt}) = -f, \quad \text{for } i = t, j \in \{0 \cup N \cup t\}. \quad (20) \]

The solution procedure itself and the definition of \( x_{ij} \) given by (14) will ensure that (17) is always satisfied. And since the flows \( \Delta f \) are always made from 0 to \( t \), the conservation equations (18) will hold for all intermediate nodes \( i \in N \), and therefore, the flow into the network from 0 will equal the flow out of the network to \( t \), as given by the equations (13), (19), and (20).

**Mandatory assignments.** To use the Flow Algorithm as a sub-routine in the Transhipment Algorithm (which will be developed in the following chapters), consider the following sets of pairs which occur in the cost matrices which are used in the solution of the transhipment problem:

\[ M = [ij; x_{ij} \text{ must } = r_{ij}], \quad (21) \]

\[ Z = [ij; 0 \leq x_{ij} \leq r_{ij}], \quad (22) \]

and

\[ P = M \cup Z. \quad (23) \]

The set \( M \) consists of the pairs \( ij \) for which it is mandatory that \( x_{ij} = r_{ij} \). In the Transhipment Algorithm, the set \( M \) arises naturally as the set of minus values in the reduced cost matrices, and the minus values require that \( x_{ij} = r_{ij} \). The set \( Z \) is the set of pairs \( ij \) for which \( x_{ij} \) can take any value in its range from 0 to \( r_{ij} \). In the Transhipment Algorithm, \( Z \) arises as the set of zero values which permit this flow. The set \( P \), then, which contains the pairs \( ij \) of both \( M \) and \( Z \), defines a partial network, or subset of the pairs \( ij \) for which \( i \in N \), \( j \in N \), \( i \neq j \), and \( r_{ij} > 0 \), and where the flow can be \( 0 \leq x_{ij} \leq r_{ij} \).

If the Flow Algorithm is not being used as a sub-routine in the Transhipment Algorithm, then the set \( M \) is empty, and the set \( P \) consists
merely of all the pairs $ij$ for which $r_{ij} > 0$. However, when the Flow Algorithm is used as a sub-routine in the Transhipment Algorithm, the set $M$ may not be empty, and there is a feasibility question as to whether any flow can be found where the matrix of flows will permit $x_{ij}$ to be equal to $r_{ij}$ for $ij \in M$. No difficulty exists, because the only condition under which a pair $ij$ can become an element of the set $M$ is if a flow $x_{ij} = r_{ij}$ has already been made at a previous cycle of the solution. (When a pair $ij$ becomes an element of the set $M$, we will bracket the entry at $ji$ in the restriction matrix to indicate that $x_{ij} = r_{ij}$ is mandatory. This will prevent a further flow $\Delta f$ from unsaturating $ij$.)

In the Transhipment Algorithm, the set $P$ is the set of pairs $ij$ at which positions there are zeros or minus values in the reduced cost matrix. Only at these positions can the flow $x_{ij}$ be greater than zero. Therefore, we have the relations,

\begin{align*}
\text{if } ij \notin P, & \quad \text{then } x_{ij} = 0, \quad (24) \\
\text{if } ij \in P, & \quad \text{then } 0 \leq x_{ij} \leq r_{ij}, \quad (25) \\
\text{if } ij \in Z, & \quad \text{then } 0 \leq x_{ij} \leq r_{ij}, \quad (26) \\
\text{and } \quad \text{if } ij \in M, & \quad \text{then } x_{ij} \text{ must } = r_{ij}. \quad (27)
\end{align*}

**Feasibility considerations.** In determining the maximum flow in a network, a point which requires examination is whether there is a finite upper bound on the maximum flow $F_{\text{max}}$ of the complete network. There are three capacities for flow which we must consider, $F_1$, $F_2$, and $F_3$. Let $F_1$ be the maximum flow capacity from the fictitious origin $O$ to all of the input nodes $i \in N_1$; let $F_2$ be the maximum flow capacity of the real network itself without the additional input-output quantity restrictions $r_1$; and let $F_3$ be the flow capacity from the output nodes $j \in N_3$ to $t$. 


\[ F_1 = \sum_j r_{0j} = \sum_j r_{ij}, \quad \text{for } icN_1, \quad (28) \]
\[ F_3 = \sum_i r_{it} = \sum_i r_{it}, \quad \text{for } icN_3, \quad (29) \]
and
\[ F_{\text{max}} = \min \{F_1, F_2, F_3\}. \quad (30) \]

Since the \( r_{ij} \) are known quantities, \( F_1 \) and \( F_3 \) are easily determined. When the flow computation procedure is being used as a sub-routine in the Transhipment Algorithm, \(-\infty < r_{ij} < \infty\), and so \( F_1, F_3 \), and therefore \( F_{\text{max}} \) are all finite. So, for applications to the transhipment problem, we make the following assumption (which will be considered further in the chapter on the transhipment problem in a section on feasibility):

\[ \text{ASSUMPTION: } F_{\text{max}} < \infty. \quad (31) \]

However, it may also be of interest to determine the maximum flow in a real network itself, without the additional input-output restrictions of the transhipment problem. To do so, let \( r_{ij} = -\infty \) for \( icN_1 \) and \( r_{ij} = -\infty \) for \( icN_3 \). In this case, \( F_1 = F_3 = -\infty \), and \( F_{\text{max}} = F_2 \). For \( F_2 \) to be infinite, there must be at least one path of infinite capacity for flow through the real network from the origin 0 to the terminal \( t \), and every arc itself in such a path must have infinite capacity for flow. Therefore, as suggested by Paul Pepper, to determine whether \( F_{\text{max}} = F_2 \) is finite or infinite, remove every arc of the network which has finite capacity, and consider the derived network which contains only those arcs of the original network which have infinite capacity for flow. The flow computation procedure to be developed will determine whether any such "infinite" path exists in the derived network—if there is such a path, then \( F_{\text{max}} = F_2 = -\infty \); if not, then \( F_{\text{max}} = F_2 < \infty \). (If all \( r_{ij} \) in the real network are finite, since \( F_{\text{max}} \) can not exceed the sum of all of these capacities, \( F_{\text{max}} = F_2 < \infty \).)
Formal statement of the flow problem. At each cycle of the Transshipment Algorithm, the set $P = M \cup Z$ is defined. The set $P$ may not include all of the arcs of the original network for which $r_{ij}$ is greater than zero—that is, $P$ may only describe a partial network, a part of the whole network of which $F^\text{max}$ is the maximum flow. Let $f^\text{max}$ be the partial-network maximum flow, where obviously

$$f^\text{max} \leq F^\text{max}.$$  \hfill (32)

To make the Flow Algorithm useful as a sub-routine in the Transshipment Algorithm, we will develop the procedure for determining $f^\text{max}$ in the partial network described by $P = M \cup Z$. A formal statement of the Maximum Flow Problem can now be given as follows.

Given: the sets $P = M \cup Z$ (where $M$ is non-empty only if the procedure is being used as a sub-routine of the Transshipment Algorithm); and a matrix of restrictions $r = [r_{ij}]$ (where only $r_{ij} > 0$ are entered, and all other positions are understood to be zero); and the input quantities $r_i$ ($r_i > 0$ for $i \in N_1$, $r_i = 0$ for $i \in N_2$, and $r_i < 0$ for $i \in N_3$);

Find: a matrix of flows $x = [x_{ij}]$ that maximizes the flow $f$ in the partial network $P$, where

$$f = \Sigma_j x_{0j} = \Sigma_i x_{it} \quad \text{(maximize)}$$  \hfill (33)

subject to the restrictions

$$\text{if } \quad i j \not\in P, \quad \text{then} \quad x_{ij} = 0,$$  \hfill (34)

$$\text{if } \quad i j \in M, \quad \text{then} \quad x_{ij} \text{ must} = r_{ij},$$  \hfill (35)

$$\text{if } \quad i j \in Z, \quad \text{then} \quad 0 \leq x_{ij} \leq r_{ij},$$  \hfill (36)

and

$$0 \leq x_{ij} \leq r_{ij}, \quad \text{for } \quad i, j \in [0 \cup N \cup t],$$  \hfill (37)

and

$$\Sigma_j (x_{ij} - x_{j1}) = 0, \quad \text{for } i \in N, \quad j \in [0 \cup N \cup t].$$  \hfill (38)
and $\sum_j (x_{0j} - x_{0j}) = f$, for $i = 0, j \in N$, (39)

$\sum_j (x_{tj} - x_{tj}) = -f$, for $i = t, j \in N$, (40)

and $x_{i0} = r_{i0} - x_{ti} = r_{ti} = 0$, for $i \in N$. (41)

To solve this problem, we propose a simple Set-Labeling Procedure which uncovers paths from origin to terminal along which incremental flows $\Delta f$ may be made until the network becomes saturated and a maximum flow has been attained. (Examples of this procedure are contained in each cycle of the solution for the example transhipment problem given in Chapter IV.)

The procedure is a systematic method of determining paths in the restriction matrix along which partial flows $\Delta f$ may be made from 0 to t. The procedure is repetitive, and on each repetition the total flow through the network from 0 to t is increased by $\Delta f$.

In making each partial trial flow $\Delta f$, the capacity restrictions are reduced by $\Delta f$ in the forward direction (from 0 to t) of flow. To take care of the possibility that certain of the arc flows may not appear in the final maximum flow, we add $\Delta f$ in the backward direction of flow, thereby producing a fictitious capacity for flow at certain positions which were not defined in the original matrix of restrictions. When no further flows $\Delta f$ can be made, the matrix of flows $x = \begin{bmatrix} x_{ij} \end{bmatrix}$ is defined by the relation

$$0 \leq x_{ij} = \max \left[ r_{ij} - r'_{ij}, 0 \right] \leq r_{ij},$$

where the $r'_{ij}$ are the final entries in the restriction matrix. From the fact that $r_{ij}$ is only greater than zero if $ij \in P$ (and $ji \notin P$), and from equation (42), it can be seen that the restrictions (34), (36), and (37) are satisfied.
Because the partial flows $\Delta f$ are all made through a continuous path from $0$ to $t$, the flow into the network from $0$ equals the flow out of the network from $t$, and the flow into any intermediate node in $N$ must also flow out of this node. Therefore, (38), (39), and (40) will hold.

The set $M$ can only be non-empty when the procedure is used as a sub-routine in the Transhipment Algorithm. As will be seen, $ij$ can only become an element of $M$ if a flow $x_{ij} = r_{ij}$ has already been made on the previous cycle of the solution. In this case, the saturating flow has reduced the entry at $ij$ in the restriction matrix to zero ($r'_{ij} = 0$), and a positive entry will appear at $ji$ ($r'_{ji} = r_{ij}$) owing to the procedure of adding the flows in the backward flow direction. The Transhipment Algorithm will require that this positive entry be "bracketed." In the maximum flow computation procedure, bracketed entries are not included, thereby preventing a further flow from unsaturating the arc $ij \in M$.

Therefore, (35) is also satisfied.

It should also be noted here that any flow $x = \sum x_{ij}$ which satisfies (34) through (40) is a feasible flow and may be used as a starting point for the procedure of maximizing the flow in the partial network described by $P$, $M$, and $Z$. At each cycle of the Transhipment Algorithm, the composition of the sets $P$, $M$, and $Z$ will change, but, as will be seen, action will be taken which does not alter the flow matrix and yet ensures that (34) through (40) remain satisfied in passing from cycle to cycle. Therefore, when the flow procedure is used as a sub-routine in the transhipment problem, the reduced restriction matrix of the previous cycle will be taken as the starting point for making the maximum flow for the current cycle.
Equation (41) holds by definition (recall that these positions were \(X\)d out in the array which is used to carry out the solution). Therefore, it only remains to show that the flow (33) produced by the procedure is, in fact, a maximum flow. This will be done after the method for determining the flow (33) is described.

**The set-labeling procedure.** In order to determine the paths along which the partial trial flows \(\Delta f\) may be made from 0 to \(t\), enumerate the numbered sets \(S_0, S_1, \ldots, S_k, \ldots\), where

\[
S_0 = [0], \text{ (the origin—not a null set), by definition}, \quad (43)
\]

and for \(k > 0\), we have

\[
S_k = [ij; i \in S_{k-1}, j \notin [S_1 \cup S_2 \cup \ldots \cup S_{k-1}], \text{ unbracketed } r_{ij} > 0]. \quad (44)
\]

The last relation (44) can be put in words as follows: To enumerate the nodes \(j\) of a set \(S_k\) for \(k > 0\), systematically (or in numerical order) scan the rows \(i\) that were enumerated in the previous set and look for unbracketed positive entries. Where unbracketed positive entries \(r_{ij}\) occur, if \(j\) has not already been listed in a previous set or* in the present set \(S_k\), enter the node \(j\) as an element of the set \(S_k\). (To aid in tracing the flow path through the network, we will keep track of the "from" node \(i\) by placing \(i\) as a prefix-subscript on the "to" node \(j\).

* A double requirement is expressed here: an element \(j\) can appear at most once in a set, and it can appear in at most one set. The situation is clear if \(j\) has already been entered in a previous set—it is not entered again in the set \(S_k\). However, confusion can arise when there are two nodes of the previous set which have unbracketed positive entries in the same column \(j\). We can not enter \(j\) in the set \(S_k\) twice to show both of the "from" modes as prefix-subscripts on \(j\). We can choose either entry, but to be systematic we will select the first occurrence. To assure ourselves that the double requirement is satisfied, we can adopt some procedure (like checking-off the column heading \(j\) as the node \(j\) is entered in a set). This will help in making sure that \(j\) is entered at most once in the numbered sets.
Termination of set-labeling. The description of the Set-Labeling procedure ensures that each set \( S_k \) for \( k > 0 \) must either contain at least one new node, or the set must be empty (that is, \( S_k = \emptyset \)). Therefore, it is certain that, for some \( k \), where \( 0 < k < n+1 \), we will finally enumerate a set \( S_k \) such that we have one of the following two cases:

\[
\text{Case I. } t \in S_k, \tag{45}
\]

or

\[
\text{Case II. } S_k = \emptyset. \tag{46}
\]

In Case I, we have located a path from \( 0 \) to \( t \) with a positive capacity for flow, and an incremental flow \( \Delta f \) is made. Using the prefix-subscripts we trace back through the numbered sets to find the sequence of nodes in the flow path. Suppose the nodes in the flow path appear in the numbered sets as follows:

\[
\begin{align*}
S_0 &= [0], \\
S_1 &= [\ldots, 0_{j_1}, \ldots], \\
S_2 &= [\ldots, j_1j_2, \ldots], \\
&\quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\
S_{k-1} &= [\ldots, j_{k-2}j_{k-1}, \ldots], \\
S_k &= [\ldots, j_{k-1}t].
\end{align*}
\tag{47}
\]

Then the sequence of nodes in the flow path is

\[
0, j_1, j_2, \ldots, j_{k-1}, t. \tag{48}
\]

By (44), a node \( j_k \) is only entered in a set \( S_k \) if \( j_{k-1} \in S_{k-1} \) and there is an unbracketed \( r^f_{j_{k-1}j_k} > 0 \). Therefore, the capacity restrictions on all arcs in this flow path are positive, and a positive trial flow \( \Delta f \) can be made as large as possible and limited only by the smallest arc capacity in the flow path. Therefore,

\[
\Delta f = \min \{r^f_{0j_1}, r^f_{j_1j_2}, \ldots, r^f_{j_{k-1}t}\} > 0. \tag{49}
\]
In making the flow $\Delta f$, we reduce the capacity restrictions in the forward direction of flow. That is, subtract $\Delta f$ from
\[ r_{Oj_1}^f, r_{j_1j_2}^f, \ldots, r_{j_{k-1}j}^f. \] (50)

Also, since it is possible (and likely) that certain of the trial arc-flows will not appear in the final maximal flow, we allow for a later flow to cancel out these undesired flows by adding a fictitious capacity restriction in the backward flow direction*. That is, add $\Delta f$ to
\[ r_{j_{k-1}j_{k-2}}^f, \ldots, r_{j_2j_1}^f. \] (51)

After each trial flow $\Delta f$ is made, the set-labeling procedure is repeated in the resulting reduced restriction matrix. The reduced matrix is always a feasible starting point for initiating further flows because the procedure assures that (34) through (41) remain satisfied.

Now since $\Delta f \geq 1$, from (49) and the fact that all non-zero entries in the restriction matrices are positive integers, and since the maximum possible flow is bounded by (31) and (32), the repetition of the flow procedure must end in Case II where, for some $k > 0$, $S_k = \emptyset$, and no further flow $\Delta f$ is possible. Therefore, a finite number of repetitions of the set-labeling procedure results in
\[ S_k = \emptyset, \quad \text{for} \quad 0 < k < n+1. \] (52)

When Case II occurs, we define the set $S$ to be the set containing all the nodes enumerated in the numbered sets $S_1$ through $S_{k-1}$, that is, $S$ will contain all the nodes connected to the origin 0 by paths of unbracketed positive capacity restriction.

*Since $x_{i1} = 0$ for $i \in N_0$, and $x_{i0} = 0$ for $i \in N_1$, $\Delta f$ is not added at the positions $t_{j_{k-1}j}$ and $j_{10}$. Further, we add $\Delta f$ at a position $ij$ if the arc $ij$ appears in the backward flow direction even though the network may not be defined for the arc $ij$, that is, $ij \notin P$. 
\[ S = S_1 \cup S_2 \cup \ldots \cup S_{k-1}. \]  

From the definition of \( S \), when Case II occurs, we also have
\[ 0 \not\in S, \quad \text{and} \quad t \not\in S. \]  

It is not necessary to enumerate the set \( S \) in the solution of a maximum flow problem itself. However, when the procedure is used as a sub-routine in the Transhipment Algorithm, the set \( S \) is enumerated to indicate the rows and columns of the reduced cost matrix which must be examined on the next cycle of the procedure to improve the dual solution and to alter the sets \( M, Z, \) and \( P \) to permit a further increase in the flow through the partial network on the next cycle.

This terminates the set-labeling procedure and the flow algorithm. The flow defined by (13) and (14) at termination of the procedure is actually the maximum possible flow in the network defined by \( P \). It now remains to prove that the flow is in fact a maximum flow.

The max-flow min-cut theorem. Ford and Fulkerson [46] first proved the max-flow min-cut theorem. An outline of the development of this theorem is given in this section, both because of its historical importance in network-flow theory, and because it provides the proof that the flow as computed above is a maximum flow.

A "cut" is defined to be a set of arcs which, when removed from the network, separates or "cuts off" the origin 0 from the terminal \( t \). (The set of cuts is not empty, for clearly the set of all arcs is a cut.)

The value of a cut is the sum of all capacity restrictions on the arcs of the cut. Clearly, any flow \( f \), including the maximum flow \( f_{\text{max}} \), must flow completely through the arcs of each cut. (If not, then the cut has not "cut off" the origin from the terminal.) Therefore, any flow must
always be equal to or less than the value of any cut (that is, the sum of the capacities for flow in the arcs of the cut). Therefore,

\[ f_{\text{max}} \leq \text{value of each cut}. \]  

(55)

The maximum flow is a lower bound on the cut values, and the minimum cut value is an upper bound on flow values. Therefore, if we can produce any flow and cut such that the flow is equal to the value of the cut, then we must have both a maximum flow and a minimum cut. That is,

\[ f_{\text{max}} = \text{value of minimum cut}, \]  

(5c)

or \( f = \text{value of a cut}, \) then \( f \) is max and cut is min,

and \( \text{MAX FLOW} = \text{MIN CUT}. \)  

(57)

At termination of the Flow Algorithm, when no further flow can be made in the network, the set \( S \) is defined and consists of all the nodes which are connected to the origin \( 0 \) by paths of positive capacity restriction, and \( 0 \not\in S \) and \( t \not\in S \). For those arcs \( ij \in P \) and leading from \( i \in [0 \cup S] \) to \( j \not\in [0 \cup S] \), \( x_{ij} = r_{ij} \), because if the arc were not saturated, then \( j \) would also be in the set \([0 \cup S]\). Therefore,

\[ \text{if } x_{ij} < r_{ij} \text{ and } i \in [0 \cup S], \text{ then } j \in [0 \cup S], \]  

(58)

and \( \text{if } i \in [0 \cup S] \text{ and } j \not\in [0 \cup S], \text{ then } x_{ij} = r_{ij}. \)  

(59)

The set of saturated arcs \( ij \) with \( i \in [0 \cup S] \) and \( j \not\in [0 \cup S] \) is a cut which completely "cuts off" the origin from the terminal. Since the flow must pass completely through these arcs of the cut, and since the flow in these arcs saturates these arcs \( (x_{ij} = r_{ij}) \), the flow is equal to the sum of the capacities on these arcs. We have achieved equality of a flow and cut value, and therefore the flow is maximum and the cut is one of minimum value. Therefore, at termination of the Flow Algorithm, the flow defined by (13) and (14) is maximum.
The numerical value of \( f_{\text{max}} \) can be gotten from (13) or from the value of the minimum cut at termination of the procedure, that is

\[
f_{\text{max}} = \sum_{ij \in \mathcal{P}} (x_{ij} - x_{ji}).
\]

Algebraic derivation of maximum flow. We can produce a maximum flow which is equal to a minimum cut value more directly by an algebraic derivation. And for use in the proof of the Transhipment Algorithm, it will be necessary to make this derivation in terms of \( K, P, \) and \( S. \) *

Consider equations (38) and (39), reproduced here for convenience:

\[
\sum_{j} (x_{ij} - x_{ji}) = 0, \quad \text{for } i \in \mathcal{N}, \quad j \in \mathcal{O} \cup \mathcal{U} \cup \mathcal{T}, \tag{61}
\]

and

\[
\sum_{j} (x_{ij} - x_{ji}) = f, \quad \text{for } i = 0, \quad j \in \mathcal{O} \cup \mathcal{U} \cup \mathcal{T}. \tag{62}
\]

Adding (62) and (61) summed for all \( i \in \mathcal{O} \cup \mathcal{S} \) gives

\[
f = \sum_{j} (x_{0j} - x_{j0}) + \sum_{i \in \mathcal{S}} \sum_{j} (x_{ij} - x_{ji}), \quad \text{for } j \in \mathcal{O} \cup \mathcal{U} \cup \mathcal{T}. \tag{63}
\]

Expanding the index in (63) we have,

\[
f = \sum_{j} (x_{0j} - x_{j0}) + \sum_{i \in \mathcal{S}} (x_{i0} - x_{0i}) + \sum_{i \in \mathcal{S}} (x_{it} - x_{ti}) + \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{N}} (x_{ij} - x_{ji}). \tag{64}
\]

Because the flow can only be positive from the origin to the input nodes and from the output nodes to \( t, \) and because backward flow from the input nodes to \( 0 \) and from \( t \) to the output nodes is always zero, the first three major terms in (64) simplify to give us,

\[
f = \sum_{j \in \mathcal{N}_1} x_{0j} - \sum_{j \in \mathcal{N}_2} x_{0j} + \sum_{i \in \mathcal{S}} x_{it} + \sum_{j \in \mathcal{N}} \sum_{i \in \mathcal{S}} (x_{ij} - x_{ji}). \tag{65}
\]

*The footnote on page 24 especially applies to the set designations of this section. For example, the designation \([O \cup \mathcal{U} \cup \mathcal{T}]\) means the same as the designation \( \{[0] \cup \mathcal{U} \cup \mathcal{T}\}. \) And, since the range of an index does not include the nodes 0 or \( t \) unless the range is specified to include these nodes, the designation \( j \not\in \mathcal{S} \) means that \( j \not= 0, \) \( j \not= t, \) and \( j \in \mathcal{N} = \mathcal{S}. \)
Changing the dummy index from $i$ to $j$ in the second term of (65) and combining the first two terms, we have

$$f = \Sigma x_{0j} + \Sigma x_{it} + \Sigma \Sigma (x_{ij} - x_{ji}).$$  \hfill (66)

Now, by expanding the third term in (66) into two double-sums, the first taken over $i \in S$ and $j \notin S$, and the second taken over $i \in S$ and $j \in S$, and observing that the second double-sum vanishes because

$$\Sigma \Sigma (x_{ij} - x_{ji}) = \Sigma x_{ij} - \Sigma x_{ji} = 0,$$

we have from (66)

$$f = \Sigma x_{0j} + \Sigma x_{it} + \Sigma \Sigma (x_{ij} - x_{ji}).$$  \hfill (67)

In (67) we have included all the saturated arcs of the cut produced by the Flow Algorithm and defined in terms of $S$. For all of these arcs, $i \in [0 \cup S]$ and $j \notin [0 \cup S]$, and therefore $x_{ij} = r_{ij}$. We have once again produced a maximum flow equal to a minimum cut value. Therefore, we may rewrite (67) as a maximum flow

$$f_{\text{max}} = \Sigma x_{0j} + \Sigma x_{it} + \Sigma \Sigma (x_{ij} - x_{ji}).$$  \hfill (68)

Under the conditions of a maximum flow, the first two terms saturate their arcs and we have

$$x_{0j} = r_{0j} = r_j, \text{ for } j \in S \text{ and } j \notin S,$$

and

$$x_{it} = r_{it} = -r_i, \text{ for } i \in S \text{ and } i \notin S.$$

Substituting these last two expressions in (68) gives

$$f_{\text{max}} = \Sigma r_i + \Sigma -r_i + \Sigma \Sigma (x_{ij} - x_{ji}).$$  \hfill (69)
The third term in (69) requires further consideration. Let us consider this third term separately. There are only two ways in which the situation i \in S and j \notin S can arise in the Flow Algorithm.

Case I. A zero can appear at ij for i \in S and j \notin S. Either ij \in P or ji \in P, but not both:

\[
\begin{align*}
\text{if } ij \in P, & \text{ then } r_{ij} > 0, r'_{ij} = 0, r_{ji} = 0, r'_{ji} = r_{ij}, \\
& \text{and } x_{ij} = \max \left[ r_{ij} - 0, 0 \right] = r_{ij}, \\
& \text{and } x_{ji} = \max \left[ 0 - r_{ij}, 0 \right] = 0, \\
& \text{and } \sum_{i \in S} \sum_{j \notin S} (x_{ij} - x_{ji}) = \sum_{i \in S} \sum_{j \notin S} r_{ij}. \\
\text{if } ij \notin P, & \text{ then } r_{ij} = 0, r'_{ij} = 0, r_{ji} > 0, r'_{ji} = r_{ji}, \\
& \text{and } x_{ij} = \max \left[ 0 - 0, 0 \right] = 0, \\
& \text{and } x_{ji} = \max \left[ r_{ji} - r_{ji}, 0 \right] = 0, \\
& \text{and } \sum_{i \in S} \sum_{j \notin S} (x_{ij} - x_{ji}) = 0.
\end{align*}
\]

(70)

Case II. A positive bracketed entry can appear at ij for i \in S and j \notin S. However, in the procedure, the only time a positive entry r'_{ij} is bracketed is when ji \in M, 0 < r_{ji} < \infty, and r'_{ji} = 0. In this case,

\[
\begin{align*}
& r_{ij} = 0, \text{ and } r'_{ij} = r_{ji}, \\
& \text{and } x_{ij} = \max \left[ 0 - r_{ji}, 0 \right] = 0, \\
& \text{and } x_{ji} = \max \left[ r_{ji} - r_{ji}, 0 \right] = r_{ji}, \\
& \text{and } \sum_{i \in S} \sum_{j \notin S} (x_{ij} - x_{ji}) = -\sum_{i \notin S} \sum_{j \in S} r_{ji} = -\sum_{i \notin S} \sum_{j \in S} r_{ij}. \\
\end{align*}
\]

(71)

Since the third term in (69) is the sum of (70), (71), and (72), we can now write the maximum flow as

\[
\begin{align*}
f_{\text{max}} &= \sum_{i \in N} r_i + \sum_{i \notin N} -r_i + \sum_{i \in S} \sum_{j \notin S} r_{ij} - \sum_{i \notin S} \sum_{j \in S} r_{ij}.
\end{align*}
\]

(73)
The form of the maximum flow given by equation (73) will be used in the next chapter to prove the validity of the procedure to be developed for the solution of the transhipment problem.
CHAPTER III
THE TRANSHIPMENT PROBLEM

In this chapter the requirements and procedure are developed for solving the Primal Transhipment Problem by means of its Dual Problem, using the Flow Algorithm of the previous chapter as a sub-routine in the solution process. In the transhipment problem, it is desired to ship input quantities from the input nodes, through the network to the output nodes where requirements for these quantities exist. The input quantity \( r_i \) is given for each node \( i \), and \( r_i \) is a positive integer for the input nodes, zero for the transport nodes, and negative integer for the output nodes. For the arcs of the network, a matrix \( r = \{ r_{ij} \} \) of integral non-negative capacity restrictions is specified. The \( r_{ij} \) are the upper bounds on the flows \( x = \{ x_{ij} \} \) in the arcs. A cost-rate matrix, which will simply be referred to as a cost matrix, \( c = \{ c_{ij} \} \) of integral and non-negative values is also given, where \( c_{ij} \) is the cost of making a unit shipment from node \( i \) to node \( j \). It is desired to make the total shipment at a minimum of cost.

The problem. As before, the network consists of a set \( N \) of \( n \) nodes,

\[ N = [1, 2, \ldots, n]. \tag{1} \]

The \( n \) nodes consist of three subsets: \( N_1 \) of \( n_1 \) "input" nodes; \( N_2 \) of \( n_2 \) "transport" nodes; and \( N_3 \) of \( n_3 \) "output" nodes.

\[ N_1 = [1, 2, \ldots, n_1], \tag{2} \]
\[ N_2 = [n_1+1, \ldots, n_1+n_2], \tag{3} \]
\[ N_3 = [n_1+n_2+1, \ldots, n], \tag{4} \]

40
where \[ n = n_1 + n_2 + n_3, \] (5)

and \[ N = N_1 U N_2 U N_3. \] (6)

Once again we see from the definitions (1) through (6) that, without loss of generality, we have numbered the nodes so that the lower numbers designate the input nodes, the higher numbers are for the output nodes, and the transport nodes come in between.

Shipping is permitted along the arc \( ij \) from \( i \) to \( j \) of the network if and only if the cost \( c_{ij} \) is finite and the capacity restriction \( r_{ij} \) is positive. Therefore, when the matrices \( c = [c_{ij}] \) and \( r = [r_{ij}] \) are given, it is understood that the values are consistent, such that

if \( r_{ij} = 0 \), then make \( c_{ij} = \infty \),

and if \( c_{ij} = \infty \), then make \( r_{ij} = 0 \).

Also it is understood that the problem is in a directed network.

If the original problem is stated in such a way as to permit real flow in both directions in the arcs of the network, the network is reconstructed, as shown in the previous chapter, to permit flow in only one direction in the arcs. Therefore, we have the conditions that

if \( r_{ij} > 0 \), then \( r_{ji} = 0 \) and \( c_{ji} = \infty \),

and if \( c_{ij} < \infty \), then \( c_{ji} = \infty \) and \( r_{ji} = 0 \).

In the actual computations for the solution of a transhipment problem, it is suggested that an \( n \) by \( n \) matrix or array be laid out like the sample cost matrix shown in Figure 4. To prevent making an entry at the position \( ij \) where \( i = j \), the main diagonal of the matrix should be blanked out. Two such matrices are used in the problem solution. The first, called the original cost matrix \( c = [c_{ij}] \), contains the cost information given in the problem statement.
Directions: The cost information given in the statement of the problem is entered in the original cost matrix 
\[ c = \begin{bmatrix} c_{ij} \end{bmatrix} \text{ or } c' = \begin{bmatrix} c'_{ij} \end{bmatrix} \]
and saved until the solution is completed. In this matrix, all costs are entered for which \( 0 < c_{ij} < \infty \). Note that, if \( c_{ij} = \infty \), then \( c'_{ij} = \infty \). It is to be understood that, when no entry appears at the position \( ij \), the cost of shipping along the arc \( ij \) is infinite, and no entry will be made at this position in any of the reduced cost matrices which follow throughout the solution procedure.

The solution procedure is carried out in reduced cost matrices 
\[ c' = \begin{bmatrix} c'_{ij} \end{bmatrix} \]
by subtracting numbers \( \Delta m \) from the rows (and adding \( \Delta m \) to the corresponding columns) of the cost matrices. (These numbers are indicated to the right of, and below, the matrix.)

At each stage of the solution procedure, it will be convenient to keep track of the total amount \( n_i \), which has been subtracted from the row (or added to the column) \( i \), and the values \( D \) and \( \Delta D \) of the Dual.

Fig. 4.—Sample cost matrix.
The actual computations are carried out in the second cost matrix, along with the restriction matrix (described in the previous chapter). In Figure 4, a column "-Δn" and a row "+Δn" have been placed, respectively, to the right of and below the matrix. This is done, or shown, because, at every cycle of the solution procedure, numbers Δn (which will be defined later) are subtracted from the elements in certain of the rows and added to the elements in certain of the columns. (Adding or subtracting to the elements of a column or row will simply be referred to as adding or subtracting to the column or row.) When numbers Δn have been subtracted from rows (and added to columns) of the original cost matrix, certain of the elements will be greater or less than their original values. A matrix consisting of these "changed" elements will be called a reduced cost matrix $c' = [c'_{ij}]$ (where superscripts indicate stages or cycles of the solution procedure).

On the reduced cost matrix at each cycle, it will be convenient to keep a running tally on the total amount $n_i$ which has been added to the column $i$ at that stage of the solution procedure. (It will also be convenient to keep track of the Dual value and its change at each cycle--both of these terms will be defined later.)

The "input" quantities $r_i$ are given integers, such that

$$0 < r_i < \infty, \quad \text{for } i \in N_1,$$  \hspace{1cm} (11a)

$$r_i = 0, \quad \text{for } i \in N_2,$$  \hspace{1cm} (11b)

and

$$-\infty < r_i < 0, \quad \text{for } i \in N_3.$$  \hspace{1cm} (11c)

The transhipment problem will be completely specified, then, by giving a cost matrix as indicated in Figure 4 and a restriction matrix as indicated in the previous chapter. In the restriction matrix, the
ri are entered in row 0 and column t. The only other entries made in the original cost and restriction matrices are those for which c_{ij} and r_{ij} satisfy (7) through (10) and for which

\[ 0 \leq c_{ij} < \infty, \quad \text{and} \quad 0 < r_{ij} \leq \infty. \]  

(12)

For emphasis, we repeat this convention: OMIT "\( \infty \)" ENTRIES IN THE ORIGINAL COST MATRIX WHERE \( c_{ij} = \infty \), AND OMIT "0" ENTRIES IN THE ORIGINAL RESTRICTION MATRIX WHERE \( r_{ij} = 0. \) *

Feasibility considerations. There are two restrictions which must be satisfied in order for the transhipment problem to have a feasible or finite cost solution. Consider the network of Figure 5 where the arcs between nodes of the real network are not shown and are assumed to lie completely within the box which represents the real network. The total capacity for flow of the real network itself is given by \( F_2 \).

*See footnote on page 19.
The restrictions which must be satisfied for feasibility are

\[ \sum_{i \in N_1} r_i = \sum_{i \in N_3} -r_i, \tag{13} \]

and

\[ F_2 \geq \max \left[ \sum_{i \in N_1} r_i, \sum_{i \in N_3} -r_i \right]. \tag{14} \]

In the network of Figure 5, an additional fictitious node \( n' \) has been added to the augmented network. It is desired to make the flow from the fictitious origin node 0 to the fictitious terminal node \( t \). As in the previous chapter, for the arcs from 0 to the real network and from the real network to \( t \) we have

\[ c_{0j} = c_{it} = 0, \quad \text{for } i,j \in N, \tag{15} \]

\[ r_{0j} = r_j, \quad \text{for } j \in N_1, \tag{16} \]

and

\[ r_{it} = -r_i, \quad \text{for } i \in N_3. \tag{17} \]

For the new arcs associated with the fictitious node \( n' \) we have

\[ c_{On'} = c_{n't} = 0, \tag{18} \]

\[ r_{On'} = \max \left[ \left( \sum_{i \in N_3} r_i - \sum_{i \in N_1} r_i \right), 0 \right], \tag{19a} \]

and

\[ r_{n't} = \max \left[ \left( \sum_{i \in N_1} r_i - \sum_{i \in N_3} -r_i \right), 0 \right]. \tag{19b} \]

From (19a) and (19b), it is not possible for both the arcs \( On' \) and \( n't \) to appear in the network at the same time. Further, when (13) holds, neither \( On' \) nor \( n't \) will appear in the network. (Also, when (13) holds, (14) reduces to \( F_2 \geq \sum_{i \in N_1} r_i = \sum_{i \in N_3} -r_i \).)

There are also arcs from the input nodes to \( n' \) and from \( n' \) to the output nodes. For these arcs we have

\[ r_{in'} = r_{n'j} = \infty, \quad \text{for } i,j \in N, \tag{20} \]

and costs must be specified for these arcs. Shipping that is indicated as being made along these fictitious paths (outside the real network) is
never "really" made. Therefore, if the sum of the input quantities is less than the sum of the output quantities, n' acts as a fictitious input node, by (19a), and a shipment is made from 0 to n' to jcN₃ to t. In this case, cₙ'ₗ for jcN₃ represents the cost of a unit shortage at the output nodes jcN₃. If the sum of the input quantities is greater than the sum of the output quantities, n' acts as a fictitious output node, by (19b), and a shipment is made from 0 to icN₁ to n' to t. In this case, cₘ for icN₁ represents the cost of a unit surplus at icN₁. Therefore, we have

\[ 0 \leq c_{in} < \infty, \quad \text{for } icN₁, \quad \text{and } c_{in} \text{ is the cost of a unit surplus at } icN₁, \quad (21) \]

and \[ 0 \leq c_{n'j} < \infty, \quad \text{for } jcN₃, \quad \text{and } c_{n'j} \text{ is the cost of a unit shortage at } jcN₃. \quad (22) \]

There are two practical and common statements of the transhipment problem. In the first statement of the problem, the rᵢ are specified and no interest is displayed in unit surplus or shortage costs. The requirement is to ship the quantities through the real network. In this case, by taking the costs in (21) and (22) to be arbitrarily large but finite, the problem will be mathematically feasible although it is not necessarily "really" feasible. If either or both of (13) or (14) do not hold, a shipment through the fictitious arcs would be indicated, and the problem is not "really" feasible. In solving such a problem it is not necessary to express the values associated with n' in either the cost or restriction matrices—the solution procedure to be developed will determine infeasibility if it exists, and will make the maximum possible shipment at a minimum of cost.
In the second statement of the problem, interest is displayed in surplus and shortage costs, and in the possibility of not making the total shipment through the real network if this leads to an overall minimum cost solution. That is, the total shipment will not be made if it is cheaper to have some surplus and shortage than to make the total shipment. In this case, realistic values can be entered in the cost matrix for $c_{in}$ for $i \in N_1$ and $c_{rj}$ for $j \in N_2$. Surpluses and shortages will occur if either or both of (13) or (14) do not hold, or if this leads to an overall minimum cost solution. In this statement of the problem, a solution is both really and mathematically feasible.

An examination of all the possible situations that can arise when either or both of (13) or (14) do not hold will show that the convention indicated above and in Figure 5 will make the problem mathematically feasible in every case. In addition, because of the definitions (15) through (22), the correct costs will always be included in the overall optimal solution.

Conventions. In the development that follows, the matrices $c = \begin{bmatrix} c_{ij} \end{bmatrix}$ and $r = \begin{bmatrix} r_{ij} \end{bmatrix}$ without superscripts will always represent the original values given in the statement of the problem. Superscripts will denote the reduced or changed values at successive stages of the solution procedure. In the general development, it will be convenient to indicate reduced matrices by single and double primes for successive stages, and "*" for the optimal solution stage. In the example problem given in Chapter IV, successive stages will be denoted by superscript numbers, for example, $c_{ij}^1$, $c_{ij}^2$, $c_{ij}^3$. 
Formal statement of the problem. The Transhipment Problem may now be formally stated as follows:

Given: $r_i$ for $i \in \mathbb{N}$, where $r_i$ is a positive integer for $i \in \mathbb{N}_1$, zero for $i \in \mathbb{N}_2$, and a negative integer for $i \in \mathbb{N}_3$, and $\sum r_i = C$; and given the matrices of non-negative integers $c = [c_{ij}]$ and $r = [r_{ij}]$, where $r_i$, $c_{ij}$, and $r_{ij}$ satisfy (7) through (12),

Find: a matrix of flows $x = [x_{ij}]$ which minimizes $C(x)$, where

$$C(x) = \sum_i \sum_j c_{ij} x_{ij}^2$$

(minimize),

subject to the restraints

$$0 \leq x_{ij} \leq r_{ij}, \quad \text{for } i,j \in \mathbb{N},$$

and

$$\sum_j (x_{ij} - x_{ji}) = r_i, \quad \text{for } i,j \in \mathbb{N}.$$  

Finding the optimizing flow matrix is the primal problem. To minimize a linear form, such as $C(x)$, subject to linear inequality restrictions, Klein [70] has proposed the use of an extended Lagrange multiplier technique. The procedure is as follows.

To eliminate the left-hand inequality in (24) substitute

$$x_{ij}^2 = x_{ij}.$$  

(26)

To eliminate the right-hand inequality of (24) introduce the "slack" variable $z_{ij}^2$ so that we have, from (24), (25), and (26),

$$x_{ij}^2 + z_{ij}^2 - r_{ij} = 0,$$

and

$$\sum_j (x_{ij}^2 - x_{ji}^2) = r_i.$$  

(27)

Now $C(X)$ must be minimized subject to (27) and (28), where

$$C(X) = \sum_i \sum_j c_{ij} x_{ij}^2.$$  

(29)

With the equation (28) for each node $i$, associate the Lagrange multiplier $\lambda_i$; and with (27) for each arc $ij$, associate the multiplier $\lambda_{ij}$. (Note that one subscript is for the node and two for the arc.)
Now consider the expression
\[ \sum_i \eta_i \left[ \sum_j (x_{ij}^2 - x_{ij}^2) - r_i \right] + \sum_i \sum_j \eta_{ij} (x_{ij}^2 + z_{ij}^2 - r_{ij}). \]  
(30)
The expression (30) is zero for all \( x_{ij} \) and \( z_{ij} \) which satisfy (27) and (28). We may rearrange the subscripts of (30) and simplify to
\[ \left[ \sum_i \sum_j (\eta_i x_{ij}^2 - \eta_j x_{ij}^2 - \eta_i r_i) + \sum_i \sum_j \eta_{ij} (x_{ij}^2 + z_{ij}^2 - r_{ij}) \right] = 0. \]  
(31)
Now subtracting (31) from (29) we have
\[ C(\lambda, Z) = \sum_i \sum_j c_{ij} x_{ij}^2 = \sum_i \sum_j (\eta_i x_{ij}^2 - \eta_j x_{ij}^2 - \eta_i r_i) + \sum_i \sum_j \eta_{ij} (x_{ij}^2 + z_{ij}^2 - r_{ij}). \]  
(32)
Rearranging the terms of (32) we have
\[ C(X, Z) = \sum_i \sum_j \left[ (c_{ij} - \eta_i + \eta_j - \eta_{ij}) x_{ij}^2 - \eta_{ij} z_{ij}^2 + \eta_i r_i + \eta_j r_{ij} \right]. \]  
(33)

Minimizing the primal. Minimizing the original primal \( C(x) \) given by (23) subject to (24) and (25) is equivalent to minimizing (33). The necessary conditions for minimizing (33) are that the first partial derivatives of \( C(X, Z) \) with respect to \( X \) and \( Z \) vanish, and the second partials be non-negative. Dantzig [20] and Charnes, Cooper, and Henderson [10] have shown that these conditions are also sufficient conditions. Their argument is based on the nature of the convex sets which occur in the general linear programming case, and so the argument holds for the present problem as a special case of linear programming. By showing that there can be no local minima or "saddle points" in the solution space for a linear programming problem, they have shown that the conditions are both necessary and sufficient.

Therefore, the necessary and sufficient conditions for minimizing (33) are that the first partial derivatives of \( C(X, Z) \) with respect to \( X \) and \( Z \) are equal to zero, and the second partials are equal to or greater than zero. The partial derivatives are as follows.
Setting the first partials equal to zero and the second partials equal to or greater than zero, we have

\begin{align*}
\frac{\partial c(x, z)}{\partial x_{ij}} &= 2 \left( c_{ij} - \pi_i + \pi_j - \pi_{ij} \right) x_{ij}, \quad (34a) \\
\frac{\partial^2 c(x, z)}{\partial x^2_{ij}} &= 2 \left( c_{ij} - \pi_i + \pi_j - \pi_{ij} \right), \quad (34b) \\
\frac{\partial c(x, z)}{\partial z_{ij}} &= -2 \pi_i z_{ij}, \quad (34c) \\
\frac{\partial^2 c(x, z)}{\partial z^2_{ij}} &= -2 \pi_i. \quad (34d)
\end{align*}

In the Transhipment Algorithm being developed, the procedure is roughly as follows. Start out in the original cost matrix \( c = \begin{bmatrix} c_{ij} \end{bmatrix} \), and find the numbers \( \pi_i \) for each node \( i \) such that when \( \pi_i \) is subtracted from row \( i \) (and therefore from the element \( c_{ij} \)) and \( \pi_j \) is added to the column \( j \) (and therefore to the element \( c_{ij} \)), conditions (35) are always satisfied. At any cycle of the solution procedure, it will be useful to designate the elements that appear in the reduced cost matrices after \( \pi_i \) has been subtracted from row \( i \) and \( \pi_j \) has been added to column \( j \) as the reduced cost elements \( c'_{ij} \), where

\begin{equation}
\begin{aligned}
c'_{ij} &= c_{ij} - \pi_i + \pi_j. \\
\end{aligned}
\end{equation}

Therefore, at any stage of the solution procedure, the elements which appear in the reduced cost matrices \( c' = \begin{bmatrix} c'_{ij} \end{bmatrix} \) are given by (36) and equations (35) must be satisfied. Rewrite equations (35) as follows.
\[(c'_{ij} - \pi_{ij}) x_{ij} = 0,\]  \hspace{1cm} (37a)

\[(c'_{ij} - \pi_{ij}) \geq 0,\]  \hspace{1cm} (37b)

\[\pi_{ij} z_{ij} = 0,\]  \hspace{1cm} (37c)

and

\[\pi_{ij} \leq 0.\]  \hspace{1cm} (37d)

The implications in equations (37) give the requirements which must be met by the elements in the reduced cost matrices at each stage of the solution procedure. First recall that from (26) and (27),

\[
\begin{align*}
\text{if } x_{ij} = 0, & \text{ then } x_{ij} = 0, \quad (38) \\
\text{if } x_{ij} \neq 0, & \text{ then } 0 < x_{ij} \leq r_{ij}, \quad (39) \\
\text{if } z_{ij} = 0, & \text{ then } x_{ij} = r_{ij}, \quad (40) \\
\text{and } & \text{ if } z_{ij} \neq 0, \text{ then } 0 \leq x_{ij} < r_{ij}. \quad (41)
\end{align*}
\]

Now, in (37a), either \(c'_{ij} = \pi_{ij}\), or \(x_{ij} = 0\), or both. Therefore, from (37a), (37b), (38), and (39), we have

\[
\begin{align*}
\text{if } & c'_{ij} > \pi_{ij}, \text{ then } x_{ij} = 0, \quad (42) \\
\text{if } & c'_{ij} = \pi_{ij}, \text{ then } 0 < x_{ij} \leq r_{ij}, \quad (43) \\
\text{if } & 0 < x_{ij} \leq r_{ij}, \text{ then } c'_{ij} = \pi_{ij}, \quad (44) \\
\text{and } & \text{ if } x_{ij} = 0, \text{ then } c'_{ij} \geq \pi_{ij}. \quad (45)
\end{align*}
\]

In (37c), either \(\pi_{ij} = 0\), or \(z_{ij} = 0\), or both. Therefore, from (37c), (37d), (40), and (41), we have

\[
\begin{align*}
\text{if } & \pi_{ij} < 0, \text{ then } x_{ij} = r_{ij}, \quad (46) \\
\text{if } & \pi_{ij} = 0, \text{ then } 0 \leq x_{ij} \leq r_{ij}, \quad (47) \\
\text{if } & 0 \leq x_{ij} < r_{ij}, \text{ then } \pi_{ij} = 0, \quad (48) \\
\text{and } & \text{ if } x_{ij} = r_{ij}, \text{ then } \pi_{ij} \leq 0. \quad (49)
\end{align*}
\]

For later convenience, combine the information contained in the implications above to show the complete range \(x_{ij}, \pi_{ij}\), and \(c'_{ij}\).
From (48) and (45),
\[
\text{if } x_{ij} = 0, \text{ then } \pi_{ij} = 0 \text{ and } c'_{ij} > 0. \tag{50}
\]
From (48) and (44),
\[
\text{if } 0 < x_{ij} < r_{ij}, \text{ then } \pi_{ij} = 0 \text{ and } c'_{ij} = 0. \tag{51}
\]
From (49) and (44),
\[
\text{if } x_{ij} = r_{ij}, \text{ then } \pi_{ij} < 0 \text{ and } c'_{ij} = \pi_{ij} < 0. \tag{52}
\]
From (47), (44), and (45),
\[
\text{if } \pi_{ij} = 0, \text{ then } 0 \leq x_{ij} \leq r_{ij} \text{ and } c'_{ij} \geq 0. \tag{53}
\]
From (46) and (44),
\[
\text{if } \pi_{ij} < 0, \text{ then } x_{ij} = r_{ij} \text{ and } c'_{ij} = \pi_{ij} < 0. \tag{54}
\]
From (42), (37d), and (50),
\[
\text{if } c'_{ij} > 0, \text{ then } x_{ij} = 0 \text{ and } \pi_{ij} = 0. \tag{55}
\]
From (37b), (53), and (54),
\[
\text{if } c'_{ij} = 0, \text{ then } 0 \leq x_{ij} \leq r_{ij} \text{ and } \pi_{ij} = 0. \tag{56}
\]
From (37b) and (54),
\[
\text{if } c'_{ij} < 0, \text{ then } x_{ij} = r_{ij} \text{ and } \pi_{ij} = c'_{ij} < 0. \tag{57}
\]

The most important of these implications (50) through (57) can be stated as follows. **WHENEVER THERE IS A POSITIVE FLOW IN AN ARC, THE ENTRY IN THE REDUCED COST MATRIX MUST BE ZERO OR NEGATIVE, by (51) and (52).** **WHENEVER A NEGATIVE ENTRY APPEARS IN THE REDUCED COST MATRIX, THE FLOW MUST SATURATE THAT ARC, AND \( \pi_{ij} \) EQUALS THE NEGATIVE ENTRY, by (54) and (57).** **WHENEVER A POSITIVE ENTRY APPEARS IN THE REDUCED COST MATRIX, THE FLOW FOR THAT ARC MUST BE ZERO, by (55).**

Before considering how the instructions contained in the implications (50) through (57) can be used to develop the Transhipment Algorithm, we will first consider the Dual of the Primal problem, and show that maximizing the Dual is equivalent to minimizing the Primal.
The dual problem. The dual problem is gotten from (33) by taking those terms which do not involve the primal variables \( X \) or \( Z \), subject to (35b) and (35d). That is, it is desired to maximize \( D(n_i, p_{ij}) \) where

\[
D(n_i, p_{ij}) = \Sigma_i n_i \Sigma_j p_{ij}, \quad \text{(maximize)}
\]

subject to the restraints

\[
(c_{ij} - n_i + n_j - p_{ij}) \geq 0, \quad (59)
\]

and

\[
p_{ij} \leq 0. \quad (60)
\]

The same procedure is used to maximize \( (58) \) subject to (59) and (60) as was previously used to minimize the primal problem. To eliminate the non-positive restraint (60) let

\[
y_{ij}^2 = -p_{ij}. \quad (61)
\]

Then (59) becomes

\[
c_{ij} - n_i + n_j + y_{ij}^2 \geq 0. \quad (62)
\]

To eliminate the inequality in (62), introduce the real-valued "slack" variable \( Z_{ij} \), so that (62) becomes

\[
c_{ij} - n_i + n_j + y_{ij}^2 + Z_{ij}^2 = 0. \quad (63)
\]

Let the Lagrange multiplier \( x_{ij} \) be associated with each equation (63), and as before in the primal problem, we can write

\[
D(n_i, y_{ij}, Z_{ij}) = \Sigma_i n_i \Sigma_j y_{ij}^2 - \Sigma_i \Sigma_j y_{ij}^2 x_{ij} + [\Sigma_i \Sigma_j x_{ij} (c_{ij} - n_i + n_j + y_{ij}^2 - Z_{ij}^2)]. \quad (64)
\]

The term in brackets in (64) is zero for \( n_i, y_{ij}, \) and \( Z_{ij} \) which satisfy (63). Maximizing (64) is equivalent to maximizing (58) subject to (59) and (60). By the same argument previously used for minimizing the primal, both the necessary and sufficient conditions for maximizing (64) are that the first partial derivatives of \( D(n_i, y_{ij}, Z_{ij}) \) with respect to \( n_i, y_{ij}, \) and \( Z_{ij} \) vanish, and that the second partials be equal to or less than zero. The first and second partial derivatives are given as follows.
\[
\begin{align*}
\frac{\partial D}{\partial r_i} &= r_i - \sum_j x_{ij} + \sum_j x_{ji} = r_i - \sum_j (x_{ij} - x_{ji}), \\
\frac{\partial^2 D}{\partial r_i^2} &= 0, \\
\frac{\partial D}{\partial y_{ij}} &= -2 (r_{ij} - x_{ij}) y_{ij}, \\
\frac{\partial^2 D}{\partial y_{ij}^2} &= -2 (r_{ij} - x_{ij}), \\
\frac{\partial D}{\partial z_{ij}} &= -2 x_{ij} z_{ij}, \\
\frac{\partial^2 D}{\partial z_{ij}^2} &= -2 x_{ij}.
\end{align*}
\]

Setting the first partials equal to zero and the second partials equal to or less than zero, we have

\[
\begin{align*}
\sum_j (x_{ij} - x_{ji}) &= r_i, \\
(r_{ij} - x_{ij}) y_{ij} &= 0, \\
x_{ij} &\leq r_{ij}, \\
x_{ij} z_{ij} &= 0, \\
0 &\leq x_{ij}.
\end{align*}
\]

From (66a) we see that one of the restraints on the primal variable \( x_{ij} \) is satisfied by the conditions for maximizing the Dual. And from (66d) and (66f) we see that \( 0 \leq x_{ij} \leq r_{ij} \), and so the other restraint on the primal variable \( x_{ij} \) is satisfied by the conditions for maximizing the Dual. This leaves (66c) and (66e) to be accounted for.

From (66c) either \( x_{ij} = r_{ij} \), or \( y_{ij} = 0 \), or both. Therefore,

\[
\begin{align*}
\text{if } x_{ij} < r_{ij}, & \quad \text{then } \pi_{ij} = 0, \\
\text{if } x_{ij} = r_{ij}, & \quad \text{then } \pi_{ij} \leq 0.
\end{align*}
\]
if $\pi_{ij} < 0$, then $x_{ij} = r_{ij}$,  \hspace{1cm} (69)

and

if $\pi_{ij} = 0$, then $0 \leq x_{ij} \leq r_{ij}$. \hspace{1cm} (70)

From (66e), either $x_{ij} = 0$, or $Z_{ij} = 0$, or both. Therefore,

if $0 < x_{ij} \leq r_{ij}$, then $c_{ij}^f = \pi_{ij}$, \hspace{1cm} (71)

if $x_{ij} = 0$, then $c_{ij}^f > \pi_{ij}$, \hspace{1cm} (72)

if $c_{ij}^f > \pi_{ij}$, then $x_{ij} = 0$, \hspace{1cm} (73)

and

if $c_{ij}^f = \pi_{ij}$, then $0 \leq x_{ij} \leq r_{ij}$. \hspace{1cm} (74)

Therefore, the conditions for maximizing the Dual are the same as for minimizing the Primal, because, from (66) through (74), the restrictions on $x_{ij}$ and on the $c_{ij}^f$ and $\pi_{ij}$ are the same.

To show that the maximum value of the Dual function is equal to the minimum value of the Primal function, consider the equation in the implication (71) in the form

$$c_{ij} = \pi_i + \pi_j - \pi_{ij} = 0, \text{ when } x_{ij} > 0.$$ \hspace{1cm} (75)

The product $(c_{ij} - \pi_i + \pi_j - \pi_{ij})x_{ij}$ is always zero either because (75) holds or because $x_{ij} = 0$. Summing all such products we have

$$\sum_i \sum_j (c_{ij} - \pi_i + \pi_j - \pi_{ij}) x_{ij} = 0, \hspace{1cm} (76)$$

or

$$\sum_i \sum_j c_{ij} x_{ij} - \sum_i \pi_i [\sum_j (x_{ij} - x_{ij})] - \sum_i \sum_j \pi_{ij} x_{ij} = 0. \hspace{1cm} (77)$$

Substituting (23) and (25) into (77), we have

$$C(x) = \sum_i \pi_i r_i + \sum_i \sum_j \pi_{ij} x_{ij}. \hspace{1cm} (78)$$

But from (69), if $\pi_{ij} \neq 0$, then $x_{ij} = r_{ij}$, and so (78) becomes

$$C(x) = \sum_i \pi_i r_i + \sum_i \sum_j \pi_{ij} r_{ij} = D(\pi_i, \pi_{ij}). \hspace{1cm} (79)$$

Since the conditions under which (79) was derived are those for which $C(x)$ is a minimum and $D(\pi_i, \pi_{ij})$ is a maximum, we have

$$\min [C(x)] = \max [D(\pi_i, \pi_{ij})]. \hspace{1cm} (80)$$
The algorithm. Before giving a detailed description of the steps of the solution procedure, we will give a general description of the procedure. We have shown that the conditions for minimizing $C(x)$ are the same as for maximizing $D(n_i, n_{ij})$, and further, that the maximum value of the Dual is equal to the minimum value of the Primal. We will minimize the Primal by finding a maximum value of the Dual, given by

$$D(n_i, n_{ij}) = \sum_i n_i r_i + \sum_{ij} n_{ij} r_{ij}. \tag{31}$$

No special effort need be made to evaluate the numbers $n_{ij}$—they will always be zero unless a negative element $c'_{ij} < 0$ appears in the reduced cost matrix, and in this case $n_{ij} = c'_{ij} < 0$.

At each cycle of the iterative procedure, we will define changes $\Delta n_i$ in the numbers $n_i$ so that the change in the Dual value is $\Delta D \geq 1$. Since there is a finite lower bound for $C(x)$ and upper bound for $D(n_i, n_{ij})$, and since $\Delta D \geq 1$ at each cycle, the procedure will terminate with a maximum value of the Dual equal to the minimum value of the original Primal, and the flow matrix $x^* = \{ x^*_{ij} \}$ at termination provides an optimal solution to the transhipment problem.

If the values of the $n_i$ at any cycle are designated by $n_i^1$, then the entries that appear in $c' = \{ c'_{ij} \}$ are always of the form

$$c'_{ij} = c_{ij} - n_i^1 + n_j^1. \tag{82}$$

Let single-primes represent the entries at one cycle, and double-primes at the next cycle. Let $n_i^1$ change by $\Delta n_i$ in passing from one cycle to the next. Then we have

$$n_i^0 = n_i^1 + \Delta n_i, \tag{83}$$

$$c_{ij}^0 = c_{ij}^1 - (n_i^1 + \Delta n_i) + (n_j^1 + \Delta n_j), \tag{84}$$

and

$$c_{ij}^0 - c_{ij}^1 = - \Delta n_i + \Delta n_j. \tag{85}$$
From (86), if \( \pi_i \) changes by \( \Delta \pi_i \), then \( \Delta \pi_i \) is subtracted from the elements in row \( i \) and added to the elements in column \( i \). This suggests the following RULE: WHENEVER A NUMBER IS SUBTRACTED FROM (OR ADDED TO) A ROW OF THE COST MATRIX, THE SAME NUMBER IS ADDED TO (OR SUBTRACTED FROM) THE CORRESPONDING COLUMN OF THE MATRIX. (86)

On each cycle of the solution procedure, the attempt is made to change the numbers \( \pi_i \) in such a way that zero or negative elements are produced at certain positions \( ij \) in the reduced cost matrix, in order that flows may be initiated through these positions in the associated partial network \( P \) which is defined in terms of the sets \( K \) and \( Z \).

\[
K = [ij; c_{ij}^i = \pi_{ij} < 0],
\]

\[
Z = [ij; c_{ij}^i = \pi_{ij} = 0],
\]

and

\[
P = K \cup Z = [ij; c_{ij}^i = \pi_{ij} \leq 0].
\]

From (50) through (57) relating the values of \( c_{ij}^i \), \( \pi_{ij} \), and \( x_{ij} \),

\[\text{if } ij \in M, \quad \text{then } x_{ij} = r_{ij} \quad \text{and } \quad c_{ij}^i = \pi_{ij} < 0, \quad (90)\]

\[\text{if } ij \notin M, \quad \text{then } 0 \leq x_{ij} \leq r_{ij} \quad \text{and } \quad c_{ij}^i = \pi_{ij} = 0, \quad (91)\]

\[\text{if } ij \in Z, \quad \text{then } 0 \leq x_{ij} \leq r_{ij} \quad \text{and } \quad c_{ij}^i = \pi_{ij} = 0, \quad (92)\]

\[\text{if } ij \in P, \quad \text{then } 0 \leq x_{ij} \leq r_{ij} \quad \text{and } \quad c_{ij}^i = \pi_{ij} \leq 0, \quad (93)\]

and

\[\text{if } ij \notin P, \quad \text{then } x_{ij} = 0 \quad \text{and } \quad c_{ij}^i = \pi_{ij} \geq 0. \quad (94)\]

When \( ij \in K \), some notation must be used to designate "\( x_{ij} \) must be equal to \( r_{ij} \)". To show this we will "place brackets around" or "remove brackets from" certain entries. In this sense, we will sometimes refer to "bracketed" or "unbracketed" entries, or elements of the matrices.

The equations and inequalities relating \( c_{ij}^i \), \( \pi_{ij} \), and \( x_{ij} \) in terms of the sets \( M \), \( Z \), and \( P \), given in (90) through (94), must be preserved at each cycle of the solution procedure.
For successive cycles of the procedure, the Dual values and the change in the Dual value are given as follows.

\[ D' = \sum_i \pi_i^r + \sum_j \sum_{ij} \pi_{ij}^r r_{ij} \]  
\[ D'' = \sum_i (\pi_i^r + \Delta \pi_i^r) r_i + \sum_j \sum_{ij} (\pi_{ij}^r + \Delta \pi_{ij}^r) r_{ij} \]  
\[ \Delta D = D'' - D' = \sum_i \Delta \pi_i^r r_i + \sum_j \sum_{ij} \Delta \pi_{ij}^r r_{ij} \]

The Transhipment Algorithm is an iterative procedure which has a cycle consisting of two steps. In Step 1, operations or calculations are performed on the cost matrix (or reduced cost matrix) of the previous cycle. The operations consist in defining \( \Delta \pi_i \) in the previous cost matrix and performing the necessary additions and subtractions to reduce the matrix to a reduced cost matrix for the present cycle. The sets \( M, Z, \) and \( P \) are determined by the zero and negative elements produced in the reduced cost matrix for the present cycle. Modifications are performed in the restriction matrix of the previous cycle to make the partial network correspond to the sets \( M, Z, \) and \( P. \)

In Step 2 of a cycle, the Flow Algorithm is used as a sub-routine to maximize the flow in the partial network described by \( P = M \cup Z \) for the present cycle. Either the total shipment is made and the transhipment algorithm terminates, or the sub-routine produces the set \( S \) of all nodes connected to the origin by paths of positive capacity restriction.

If the procedure has not terminated with Step 2, a new cycle is begun with Step 1. The set \( S \) from the last cycle is used in the definition of \( \Delta \pi_i. \) Depending on the value determined for \( \Delta \pi_i, \) either the procedure terminates, or the operations described above are continued with a repeat of Step 2. In this way, the algorithm will eventually end on either Step 1 or Step 2.
The starting procedure. On the very first cycle, there is no set $S$ from a previous cycle to use to determine $\Delta n_i$. Therefore, on the first cycle, a special starting procedure is used to replace Step 1 of the iterative procedure. An initial feasible solution of the Dual is given by $\pi_i^0 = \pi_{ij}^0 = D^0 = 0$, and $c_{ij} \geq 0$. The set $M$ is empty, no flow has been made through the network, and, therefore, (90) through (94) are satisfied. Starting from this feasible solution, we will define $\Delta n_i$ so that (90) through (94) hold, and the value of the Dual increases.

Step A. In the original cost matrix $c = [c_{ij}]$, subtract the minimum value in each input node row from every element in the row, and add the value to every element in the input node column. The set $M$ is empty, and (90)-(94) hold. Actually we have defined the following terms.

$$\Delta n_i = \min_j c_{ij} \geq 0, \text{ for } i \in N_1,$$

$$\Delta n_i = 0, \text{ for } i \notin N_1,$$

and

$$\Delta n_{ij} = 0.$$ 

Therefore,

$$\pi_i^1 = \pi_i^0 + \Delta n_i = \Delta n_i,$$

$$\Delta D = \sum_{i \in N_1} \pi_i^1 r_i > 0,$$

$$D^1 = D^0 + \Delta D = \Delta D > 0,$$

and

$$c_{ij}^1 = c_{ij} - \pi_i^1 + \pi_j^1 \geq 0.$$

Step B. In the matrix $c^1 = [c_{ij}^1]$, subtract the minimum value in each output node column from every element in the column, and add the value to every element in the corresponding output node row. Since the values which are subtracted are the smallest of the (positive) values in the output node columns, once again no negative elements are produced, the set $M$ is empty, and the relations (90)-(94) remain satisfied. Actually we have defined the following terms.
\[
\Delta n_j = - \min_i c_{ij} \leq 0, \quad \text{for } j \in N_3, \tag{105}
\]
\[
\Delta n_j = 0, \quad \text{for } j \notin N_3, \tag{106}
\]
and
\[
\Delta n_{ij} = 0.
\]
Therefore,
\[
\pi_{ij}^2 = \pi_{ij}^1 + \Delta n_{ij}, \tag{107}
\]
\[
\Delta D = \sum_{i \in N_3} \Delta n_{ij} r_i > 0, \tag{108}
\]
\[
D^2 = \sum_{i \in N_1} \pi_{ij}^1 r_i + \sum_{i \in N_3} \pi_{ij}^2 r_i \geq D^1 > 0, \tag{109}
\]
and
\[
c_{ij}^2 = c_{ij} - \pi_{ij}^1 + \pi_{ij}^2 \geq 0. \tag{110}
\]

Although they need not be, Step A and Step B may be repeated until there is a zero in each input node row and each output node column. If Step A or Step B is repeated, (90)-(94) will remain satisfied and the Dual value will increase. (It should also be noted that neither Step A nor Step B is necessary. We could begin with Step 1 of the iterative procedure with only the input and output quantities entered in the restriction matrix and the definition \( S = \{i; i \in N_1\} \) at the start.)

Step G. Set up the restriction matrix for the starting cycle by entering \( r_1 \) for \( i \in N_1 \) in row "0" and \( -r_1 \) for \( i \in N_3 \) in column "t". Also enter \( r_{ij} \) at positions \( ij \) where zeros have appeared during Step A and Step B, that is, where \( ij \in P = \mathbb{M} U \mathbb{Z} = \mathbb{Z}. \)

This concludes the starting procedure which is a substitute for Step 1 on the first cycle. To conclude the first cycle we proceed to Step 2 of the iterative procedure where the Flow Algorithm is used as a sub-routine to maximize the flow in the partial network described by \( P \). Either the procedure terminates at Step 2, or we define the set \( S \) where
\[
S = S_1 U S_2 U ... U S_{k-1}, \tag{111}
\]
and begin the second cycle with Step 1 of the iterative procedure.
The iterative procedure. Step 1 of the iterative procedure. $S$ is the output of Step 2 of the previous cycle. The following operations or calculations are performed on the reduced cost matrix of the last cycle.

Place a minus sign $-$ at the right of the rows $i \in S$, and place a plus sign $+$ at the bottom of the columns $j \in S$. (In this sense we will give the instructions "minus the row" or "plus the column", and we will refer to "minused rows" or "plussed columns", of a reduced cost matrix.) Define $\Delta m$ as the smallest of the following values: the smallest of the positive values in $-\infty$ rows not in $+\infty$ columns; and the smallest absolute value of the negative values in $+\infty$ columns not in $-\infty$ rows.

$$\Delta m = \min \left[ \min_{i \notin P} c_{ij}, \min_{i \in S} c_{ij}^{'}, \min_{j \notin S} c_{ij}^{'}, -\infty \right] \geq 1. \quad (112)$$

For the first term in brackets in (112), if there is such an entry $i \notin P$, $i \in S$, and $j \notin S$ in the matrix, the entry is a positive integer; if there is no such entry, the first term is taken to be $\infty$. (Recall the convention of omitting "$-\infty$" entries where $c_{ij} = \infty$.) For the second term, if there is such an entry $i \in S$, $i \notin S$, and $j \in S$ in the matrix, the entry is a negative integer and its absolute value is positive; if there is no such entry, the second term is taken to be $\infty$.

If $\Delta m = \infty$, terminate the procedure—the original problem is not really feasible, and the flow from Step 2 of the previous cycle is the maximum (and optimum) flow permitted in the whole network.

If the procedure is to be continued ($\Delta m \neq \infty$), then

$$1 \leq \Delta m < \infty. \quad (113)$$

Now subtract $\Delta m$ from the $-\infty$ rows and add $\Delta m$ to the $+\infty$ columns.
In this way we have defined the following terms for this cycle.
\[ \Delta \pi_i = \Delta \pi, \quad \text{for } i \in S, \quad (114) \]
\[ \Delta \pi_i = 0, \quad \text{for } i \notin S, \quad (115) \]
\[ \Delta \pi_{ij} = -\Delta \pi, \quad \text{for } ij \in P, i \in S, j \notin S, \quad (116) \]
\[ \Delta \pi_{ij} = \Delta \pi, \quad \text{for } ij \in M, i \notin S, j \in S, \quad (117) \]
and \[ \Delta \pi_{ij} = 0, \quad \text{otherwise.} \quad (118) \]

Let us consider further the changes which occur in the elements of the reduced cost matrices during Step 1, keeping in mind (90)-(94).

From (65), (114), and (115),

if \( i \in S \) and \( j \notin S \), then \( c''_{ij} - c'_{ij} = -\Delta \pi + \Delta \pi = 0 \), \( (119) \)

if \( i \notin S \) and \( j \notin S \), then \( c''_{ij} - c'_{ij} = 0 + 0 = 0 \), \( (120) \)

if \( i \in S \) and \( j \notin S \), then \( c''_{ij} - c'_{ij} = -\Delta \pi + 0 = -\Delta \pi \), \( (121) \)

and if \( i \notin S \) and \( j \in S \), then \( c''_{ij} - c'_{ij} = 0 + \Delta \pi = \Delta \pi \). \( (122) \)

Since there is no change in the sets \( M, Z, \) or \( P \) in cases (119) and (120), restrictions (90)-(94) will hold for these cases. However, (121) and (122) must be examined in greater detail.

From (121), (122), and the definition of \( \Delta \pi \) given by (112), it can be seen that an element can not go from a positive value directly to a negative value, or from a negative value directly to a positive value. This is so because \( \Delta \pi \) has been taken to be the smallest absolute value of the positive and negative values which do undergo a change.

Also, changes of the form

\[ c'_{ij} > 0, \quad c''_{ij} = c'_{ij} \pm \Delta \pi > 0, \quad (123) \]

and

\[ c'_{ij} < 0, \quad c''_{ij} = c'_{ij} \pm \Delta \pi < 0, \quad (124) \]
do not produce any change in the composition of the sets \( M, Z, \) or \( P \), and so (90)-(94) will remain satisfied for these cases. Therefore, only the following four changes must be examined in detail.
Change I. \( c_{ij} > 0, \, i \notin S, \, j \notin S; \quad c''_{ij} = c'_{ij} - \Delta n = 0 \). \hspace{1cm} (125)

Change II. \( c_{ij} = 0, \, i \notin S, \, j \in S; \quad c''_{ij} = c'_{ij} + \Delta n > 0. \hspace{1cm} (126)

Change III. \( c_{ij} = 0, \, i \in S, \, j \notin S; \quad c''_{ij} = c'_{ij} - \Delta n < 0. \hspace{1cm} (127)

Change IV. \( c_{ij} < 0, \, i \notin S, \, j \in S; \quad c''_{ij} = c'_{ij} + \Delta n = 0. \hspace{1cm} (128)

In order to preserve the relations (90)-(94), when any of the four changes above occurs, the network described by \( P = M \cup Z \) changes, and some modification or action must be taken in the restriction matrix of the previous cycle. When reducing the cost matrix by subtracting \( \Delta n \) from the elements in "-" rows not in "+" columns and adding \( \Delta n \) to the elements in "+" columns not in "-" rows, some of the above changes will occur. To keep track of the positions \( ij \) where action must be taken for one of the above changes, we will put a "*" at \( ij \) in the cost matrix.

Change I. A POSITIVE ELEMENT HAS BECOME ZERO. Since \( c'_{ij} \) was greater than zero, there was no entry at \( ij \) in the restriction matrix, and \( x_{ij} = 0 \). But now \( c''_{ij} = 0, \) \( ij \) has become an element of \( Z \), and the flow \( x_{ij} \) must be permitted to become greater than zero. ACTION: ENTER \( r_{ij} \) AT \( ij \) IN THE RESTRICTION MATRIX.

Change II. A ZERO ELEMENT HAS BECOME POSITIVE. An entry \( r_{ij} \) was made in the restriction matrix when \( ij \) became an element of \( Z \). If any flow had been made through \( ij \), then there would be a positive (and unbracketed) entry at \( ji \) in the restriction matrix, in which case, if \( j \in S, \) then \( i \in S \) (by the set-labeling procedure). Therefore, since \( j \in S \) and \( i \notin S, \) \( x_{ij} = 0 \). Now, since \( c''_{ij} > 0, \) we must remove the entry at \( ij \) in the restriction matrix to prevent a flow from being made through this position. ACTION: REMOVE THE ENTRY AT \( ij \) IN THE RESTRICTION MATRIX.
Change III. A ZERO ELEMENT HAS BECOME NEGATIVE. An entry $r_{ij}$ was made in the restriction matrix when $ij$ became an element of $Z$. Now, since $icS$ and $jS$, a flow $x_{ij} = r_{ij}$ has been made so that the entry at $ij$ in the restriction matrix is zero. And, because of the backward flow, a positive entry (equal to $r_{ij}$) now appears at $ji$ in the restriction matrix. Since we now have $ij \in K$, $x_{ij} = r_{ij}$. To prevent a further flow from being made through the position $ji$ (which would unsaturate the arc $ij$), we bracket the positive entry at $ji$ in the restriction matrix. (Recall that bracketed entries are skipped in the set-labeling routine.)

ACTION: BRACKET THE POSITIVE ENTRY AT $ji$ IN THE RESTRICTION MATRIX.

Change IV. A NEGATIVE ELEMENT HAS BECOME ZERO. When $ij$ became an element of $N$, the positive entry at $ji$ in the restriction matrix was bracketed to preserve the mandatory flow $x_{ij} = r_{ij}$. Since we now have $ij \notin N$, it is no longer mandatory that $x_{ij} = r_{ij}$, and we must permit the arc $ij$ to become unsaturated. ACTION: REMOVE THE BRACKETS FROM THE POSITIVE ENTRY AT $ji$ IN THE RESTRICTION MATRIX.

With the action that has been taken for the four changes considered above, (90)-(94) remain satisfied at each cycle of the algorithm. The sub-routine may now be used to maximize the flow in the partial network.

Although it is not necessary, it will be useful to keep a running "tab" on the values $n_i$ and $D$ throughout the solution procedure, in order to check the dual value and the entries in the reduced cost matrices. (It is not necessary to keep track of the $n_{ij}$ because all non-zero values of $n_{ij}$ will appear as the negative elements in the reduced cost matrices at each cycle.) This terminates Step 1 of the cycle.

Proceed to Step 2 of the iterative procedure.
The sub-routine. Step 2 of the iterative procedure. With the action taken in Step 1 above, (90)-(94) hold, and it is assured that

\[ \text{if } i j \notin P, \text{ then } x_{ij} = 0, \quad (129) \]

and

\[ \text{if } i j \in K, \text{ then } x_{ij} = r_{ij}. \quad (130) \]

Therefore, the modified restriction matrix of the previous cycle is a suitable starting point for maximizing the flow in the network \( P = M \cup Z \) of the present cycle.

Apply the Flow Algorithm of the previous chapter to maximize the flow. The set-labeling procedure will terminate with an empty set.

If \( S_1 = \emptyset \), terminate the procedure---a maximum and optimum flow \( x^* = [x_{ij}^*] \) has been achieved, where

\[ x_{ij}^* = \max\{r_{ij} - r_{ij}^*, 0\}. \quad (131) \]

If \( S_1 \neq \emptyset \), the set-labeling procedure will terminate with

\[ S_k = \emptyset, \text{ for } 1 < k < n+1. \quad (132) \]

In the case where (132) occurs, define the set \( S \) by

\[ S = S_1 \cup S_2 \cup \ldots \cup S_{k-1}. \quad (133) \]

Return to Step 1 and begin a new cycle of the procedure.

This completes the explanation of the transhipment algorithm. Now is a suitable time to give an informal explanation of why the conditions for optimizing the primal and dual problems are both necessary and sufficient. At the start of the procedure, \( D = C = 0 \) and \( x_{ij} = 0 \).

On the first cycle, the cheapest cost paths in the network are selected and a maximum flow is made through these paths. If this flow had been made along any other paths, the cost of the partial shipment would have been at least as great. Therefore, on the first cycle, the dual value is a lower bound on the cost of making a partial shipment.
as great as the maximum flow of the first cycle, that is, \( D^1 \leq C^1 \). The procedure selects the flow such that \( D^1 = C^1 \). Therefore, the cost of making the partial shipment of the first cycle is a minimum.

If the procedure continues, on each successive cycle (90)-(94) are satisfied, and the procedure selects the next cheapest cost paths in the network. The flow is increased until a maximum flow has been achieved in the new partial network. Once again, we have \( D^i \leq C^i \). Since the procedure selects the flow such that \( D^i = C^i \), the cost of making the partial shipment of any cycle is a minimum. Therefore, at termination of the algorithm when the maximum flow for the whole network has been achieved, the cost of making the total shipment is a minimum.

**Increase in the dual value.** It remains to show that \( \Delta D \) is positive (that is, equal to or greater than one) for each application of Step 1.

From (97) we have

\[
\Delta D = \sum_i \Delta \pi_i r_i + \sum_j \Delta \pi_{ij} r_{ij}. \tag{134}
\]

Substituting for \( \Delta \pi_i \) and \( \Delta \pi_{ij} \), from (114) through (118) we have

\[
\Delta D = \Delta \pi \left[ \sum_{i \in S} r_i + \sum_{i \in M} \sum_{j \in P} r_{ij} - \sum_{i \in S} \sum_{j \notin S} r_{ij} \right]. \tag{135}
\]

Since \( 1 \leq \Delta \pi \leq \infty \) under the conditions of (113), it is only necessary to show that the term in brackets in (135) is equal to or greater than one. Consider the expression for \( f_{\text{max}} \) derived in Chapter II and repeated here for convenience.

\[
f_{\text{max}} = \sum_{i \notin S} r_i + \sum_{i \in S} -r_i + \sum_{i \in M} \sum_{j \in P} r_{ij} - \sum_{i \in S} \sum_{j \notin S} r_{ij}. \tag{136}
\]
Since the algorithm did not terminate on Step 2 of the previous cycle, a maximum flow for the whole network had not been achieved.

\[ f_{\text{max}} < \sum_{i \in N_1} r_i. \]  

(137)

From (136) and (137), we have

\[ \sum_{i \in S} \sum_{j \in P} r_{ij} - \sum_{i \in S} \sum_{j \in S} r_{ij} < \sum_{i \in N} r_i, \]

(138)

\[ \sum_{i \in S} \sum_{j \in P} r_{ij} - \sum_{i \in S} \sum_{j \in S} r_{ij} > 0, \]

(139)

\[ \sum_{i \in S} \sum_{j \in P} r_{ij} - \sum_{i \in S} \sum_{j \in S} r_{ij} > 0. \]

(140)

The \( r_i \) and \( r_{ij} \) in (140) are all integers, and therefore, the term in brackets in (135) is equal to or greater than one. Therefore,

\[ \Delta D \geq 1. \]  

(141)

Since the dual is bounded and increases by an integral amount at each cycle, the algorithm will terminate in a finite number of applications of the solution procedure.

**Summary.** The conditions for maximizing the Dual and minimizing the Primal Transhipment Problem have been derived and the proofs for the proposed solution procedure have been given.

In the next chapter, an informal statement of the Transhipment Algorithm, and the Flow Algorithm sub-routine, will be given. The procedure will be illustrated by an example transhipment problem.
CHAPTER IV
THE TRANSHIPMENT ALGORITHM

In the present chapter, an algorithm is presented, without proofs, for solving the transhipment problem (which includes as special cases both the capacitated and uncapacitated versions of the transportation problem, and the assignment problem). The Transhipment Algorithm uses the Flow Algorithm as a sub-routine for determining the maximum flow in the partial restricted network at each cycle of the procedure.

The Primal Problem is to find a matrix $x = [x_{ij}]$ of flows, so as to minimize the cost $C(x)$, which is given by

$$C(x) = \sum_{i,j} c_{ij} x_{ij}, \text{ (minimum)},$$

where the flow $x_{ij}$ is subject to the restraints

$$\sum_{j} (x_{ij} - x_{ji}) = r_i, \text{ for } i \in N,$$

and

$$0 \leq x_{ij} \leq r_{ij}, \text{ for } i,j \in N,$$

where $c = [c_{ij}]$ and $r = [r_{ij}]$ are given matrices of non-negative integers, and the $r_i$ are given positive integers for the input nodes $i \in N_1$, $r_i = 0$ for the transport nodes $i \in N_2$, and $r_i$ are given negative integers for the output nodes $i \in N_3$.

It will be assumed that the feasibility considerations of the previous chapters have been satisfied, namely that

$$\sum_{i \in N_1} r_i = \sum_{i \in N_3} r_i,$$

and that the maximum flow of the network will permit some feasible flow to be made at some finite cost.
The Dual Problem will be used to find the solution to the Primal Problem. The Dual Problem consists in finding the numbers \( \pi_i \) so as to maximize the Dual value \( D(\pi_i, \pi_{ij}) \), which is given by
\[
D(\pi_i, \pi_{ij}) = \sum_i \pi_i r_i + \sum_{ij} \pi_{ij} r_{ij},
\]
where the \( \pi_i \) reduce the elements of the cost matrix to
\[
c'_{ij} = c_{ij} - \pi_i + \pi_j,
\]
and where
\[
\pi_{ij} = c'_{ij}, \quad \text{when} \quad c'_{ij} < 0,
\]
and
\[
\pi_{ij} = 0, \quad \text{when} \quad c'_{ij} \geq 0.
\]

The restrictions on the flow are given by the following:
\[
\text{if} \quad c'_{ij} > 0, \quad \text{then} \quad x_{ij} = 0,
\]
\[
\text{if} \quad c'_{ij} = 0, \quad \text{then} \quad 0 \leq x_{ij} \leq r_{ij},
\]
and
\[
\text{if} \quad c'_{ij} < 0, \quad \text{then} \quad x_{ij} \text{ must} = r_{ij}.
\]

The set \( S \) (the output of the Maximum Flow sub-routine) will indicate the nodes for which the \( \pi_i \) change on each cycle of the algorithm. It will be convenient to keep track of the changing values of the \( \pi_i \) in order to check the overall solution. However, there is no need to keep track of the values \( \pi_{ij} \) since they are only non-zero when a negative value appears in the reduced cost matrix, and in this case the \( \pi_{ij} \) equals the negative value.

A transhipment problem. An example problem shown in Figure 6, will be worked out step-by-step to illustrate the solution procedure. The cost information is arranged in the original cost matrix \( c = [c_{ij}] \) and both the "input" restrictions on the nodes \( r_i \) and the restrictions on the arc flows \( r_{ij} \) are arranged in the original restriction matrix \( r = [r_{ij}] \).
<table>
<thead>
<tr>
<th>Costs $c_{ij}$</th>
<th>Restrictions $r_{ij}$</th>
<th>&quot;inputs&quot; $r_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{13} = 5$</td>
<td>$r_{13} = -$</td>
<td>$r_1 = 11$</td>
</tr>
<tr>
<td>$c_{14} = 7$</td>
<td>$r_{14} = -$</td>
<td>$r_2 = 8$</td>
</tr>
<tr>
<td>$c_{16} = 2$</td>
<td>$r_{16} = 1$</td>
<td>$r_3 = 0$</td>
</tr>
<tr>
<td>$c_{24} = 1$</td>
<td>$r_{24} = 2$</td>
<td>$r_4 = 0$</td>
</tr>
<tr>
<td>$c_{25} = 4$</td>
<td>$r_{25} = -$</td>
<td>$r_5 = 0$</td>
</tr>
<tr>
<td>$c_{34} = 1$</td>
<td>$r_{34} = 9$</td>
<td>$r_6 = -15$</td>
</tr>
<tr>
<td>$c_{36} = 20$</td>
<td>$r_{36} = -$</td>
<td>$r_7 = -4$</td>
</tr>
<tr>
<td>$c_{45} = 2$</td>
<td>$r_{45} = -$</td>
<td></td>
</tr>
<tr>
<td>$c_{46} = 6$</td>
<td>$r_{46} = 5$</td>
<td></td>
</tr>
<tr>
<td>$c_{47} = 3$</td>
<td>$r_{47} = 3$</td>
<td></td>
</tr>
<tr>
<td>$c_{56} = 5$</td>
<td>$r_{56} = -$</td>
<td></td>
</tr>
<tr>
<td>$c_{57} = 2$</td>
<td>$r_{57} = 5$</td>
<td></td>
</tr>
<tr>
<td>$c_{76} = 2$</td>
<td>$r_{76} = -$</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6.—Example transhipment problem.
Directions: The cost information given in the original statement of the problem is entered in this matrix and saved until the end of the solution procedure.
Original Restriction Matrix $r = \begin{bmatrix} r_{ij} \end{bmatrix}$

For the Complete Network

Directions: The restrictions on the nodes $r_i$ and on the arcs $r_{ij}$ are entered in this matrix and saved until the end of the solution procedure. The capacity for flow at those positions where no entry has been made is understood to be zero.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>$r_1=11$</td>
<td>$r_2=8$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>2</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>9</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$-r_6=15$</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$-r_7=4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
The transshipment algorithm. WHENEVER A NUMBER IS SUBTRACTED FROM (OR ADDED TO) A ROW OF THE COST MATRIX, THE NUMBER IS ADDED TO (OR SUBTRACTED FROM) THE CORRESPONDING COLUMN OF THE COST MATRIX. Keep track of the numbers \( n_i \) which are added to the columns of the cost matrix.

Also keep track of the value of the Dual D.

STARTING PROCEDURE: In the matrix \( c = \begin{bmatrix} c_{ij} \end{bmatrix} \), subtract (add) the smallest value in each input node row from (to) every element in the input node row (column). In the resulting matrix \( c^1 = \begin{bmatrix} c^1_{ij} \end{bmatrix} \), subtract (add) the smallest value in each output node column from (to) every element in the output node column (row). A matrix \( c^2 = \begin{bmatrix} c^2_{ij} \end{bmatrix} \) results.*

Set up a restriction matrix entering \( r_{ij} \) at those positions \( ij \) where a zero appears in \( c^2 = \begin{bmatrix} c^2_{ij} \end{bmatrix} \), and also enter the \( r_i \) for \( i \in N_1 \) in row 0 and the \( -r_i \) for \( i \in N_3 \) in column \( t \).

Proceed to Step 2 of the iterative procedure.

STEP 1 OF THE ITERATIVE PROCEDURE: This step begins a new cycle of the iterative procedure. \( S \) is the set of nodes which is determined in the Step 2 of the previous cycle of the procedure.

In the reduced cost matrix, place a "-" at the right of the rows \( i \in S \), and place a "+" at the bottom of the columns \( j \in S \).

Define \( \Delta n \) as the smaller of the following two terms: the smallest positive value in a "-" row and not in a "+" column—if there is no such term, take this first term to be \( \infty \); and the smallest absolute value of the negative values not in a "-" row and in a "+" column—if there is no such term, take this second term to be \( \infty \).

*These steps may be repeated, but they need not be.
If \( \Delta w = 0 \), TERMINATE THE PROCEDURE. The original problem is not feasible, and the maximum possible flow has already been made through the network in an optimal manner.

If \( 0 < \Delta w < \infty \), subtract \( \Delta w \) from the "-" rows and add \( \Delta w \) to the "+" columns. In this reduction process, where one of the following changes occurs, temporarily place a "*" in the position of the change to indicate that ACTION MUST BE TAKEN in the restriction matrix.

CHANGE I. A POSITIVE ELEMENT BECOMES ZERO. ACTION: ENTER \( r_{ij} \) AT THE POSITION \( ij \) IN THE RESTRICTION MATRIX.

CHANGE II. A ZERO ELEMENT BECOMES POSITIVE. ACTION: REMOVE \( r_{ij} \) FROM THE POSITION \( ij \) IN THE RESTRICTION MATRIX.

CHANGE III. A ZERO ELEMENT BECOMES NEGATIVE. ACTION: BRACKET \( r_{ij} \) AT THE POSITION \( ji \) (NOT \( ij \)) IN THE RESTRICTION MATRIX.

CHANGE IV. A NEGATIVE ELEMENT BECOMES ZERO. ACTION: REMOVE THE BRACKETS FROM \( r_{ij} \) AT \( ji \) (NOT \( ij \)) IN THE RESTRICTION MATRIX.

Proceed to Step 2 of the iterative procedure.

STEP 2 OF THE ITERATIVE PROCEDURE: The Flow Algorithm is used as a subroutine for maximizing the flow in the partial restriction matrix.

Define sets as follows: \( S_0 = [c] \), and for \( k > 0 \), \( S_k \) is given by

\[
S_k = [ij; i \in S_{k-1}, j \notin (S_1 \cup S_2 \cup \ldots \cup S_{k-1}) \text{, unbracketed } r_{ij} > 0].
\]

That is, to list the nodes of the set \( S_k \), scan the rows \( i \) of the previous set and look for unbracketed positive entries. If such an \( r_{ij} \) occurs, \( j \) is entered in \( S_k \) if it has not already been included in a previous set or in the present set. (To aid in tracing the flow path through the network, we will keep track of the "from" node \( i \) by placing \( i \) as a prefix-subscript on the "to" node \( j \).)
The set-labeling procedure will terminate in one of the following.

Case I. For some \( k \), \( tcS_k \). In this case a path for flow from 0 to \( t \) has been located. Using the prefix-subscripts, trace back through the numbered sets, and find the flow path through the network. A flow \( \Delta f \) is made along the path, and is as large as possible limited only by the minimum restriction in the flow path. Make the flow \( \Delta f \) by subtracting \( \Delta f \) from the restrictions in the flow path in the forward direction of flow (from 0 to \( t \)), and add \( \Delta f \) in the positions corresponding to a backward flow in the path. (Entries are never made at \( tj \) for \( jCN_3 \) or at \( i0 \) for \( iCN_1 \)—these positions are \( x'd \) out in the matrix.) Repeat the set-labeling procedure in the resulting restriction matrix making flows \( \Delta f \) until Case II or Case III occurs.

Case II. \( S_1 = \emptyset \), that is, \( S_1 \) is empty. In this case, TERMINATE THE PROCEDURE. The maximum and optimal flow has been attained, and the problem is solved by defining the optimizing flow matrix \( \mathbf{x}^* \) by

\[
x_{ij}^* = \max \{ r_{ij} - r_{ij}^*, 0 \}.
\]

Case III. \( S_k = \emptyset \) for \( 1 < k < n+1 \). In this case, define the unnumbered set \( S \) as the set of all nodes contained in the numbered sets \( S_1, S_2, \ldots, S_{k-1} \). That is,

\[
S = S_1 \cup S_2 \cup \ldots \cup S_{k-1}.
\]

Return to Step 1 of the iterative procedure.

Example solution. Following is the solution to the example problem given in Figure 6. A cumulative tally will be kept on the \( \pi_i, \Delta D \), and \( D \), in order to check the solution at the end of the procedure (at which stage, a "*" superscript is used to indicate termination).
The starting procedure (a substitute for Step 1 on the first cycle) is carried out in cost matrices $c$, $c^1$, and $c^2$. The zeros which appear in $c^2$ are entered in the partial network restriction matrix $r^{2a}$. Step 2 is performed by determining the maximum flow at this stage in the matrices $r^{2a}$ and $r^2$. In $r^2$ a maximum flow has been achieved, and the set $S$ is enumerated, thus completing the first cycle of the algorithm.

From this point on, each cost matrix begins a new cycle of the iterative procedure. In the example, the next cycle begins with $c^3$. However, it should be noted that the operations which reduce $c^2$ to $c^3$ are indicated on the previous cost matrix $c^2$. Also, the action which is taken in reducing $c^2$ to $c^3$ converts $r^2$ into $r^{3a}$, which is a suitable starting point for maximizing the flow in the new partial network.

Sufficient directions for following the procedure will be given in the beginning of the solution of the example problem.

The following relations hold at any cycle of the solution procedure and can be used to check the computations, and they are especially useful in checking the work when the overall solution does not check, that is, when the maximum value of the Dual does not equal the minimum Cost.

$$c^i_{ij} = c_{ij} - \pi_i^i + \pi_j^i.$$  
$$\pi_i^i = c^i_{ij} \text{ only if } c^i_{ij} < 0,$$
and
$$\pi_i^i = 0, \text{ otherwise.}$$

$$x^i_{ij} = \max [r_{ij} - r_{ij}^i, 0].$$

$$D(\pi_i, \pi_{ij}) = \Sigma_i \pi_i^i r_{ij} + \Sigma_j \pi_{ij}^i r_{ij}.$$  
$$C(x) = \Sigma_i \Sigma_j c_{ij} x_{ij}^i.$$  
$$C(x) = D(\pi_i, \pi_{ij}).$$
Directions: The starting procedure begins in the original cost matrix \( c = \begin{bmatrix} c_{ij} \end{bmatrix} \). Subtract the smallest value in each input node row from every element in the input node row (and add the smallest value in each input node row to every element in the input node column). The reduction is only indicated on this matrix. The results are shown in matrix \( c^1 \) which follows.

\[
c^0 = \begin{bmatrix} c_{ij}^0 \end{bmatrix} = \begin{bmatrix} c_{ij} \end{bmatrix} - \Delta \pi_1
\]

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & -\Delta \pi_1 \\
\hline
1 & X & 5 & 7 & & 2 & & -2 \\
2 & X & 1 & 4 & & & & -1 \\
3 & X & 1 & & 20 & & & \\
4 & X & 2 & 6 & 3 & & & \\
5 & & X & 5 & 2 & & & \\
6 & & & & & X & & \\
7 & & & & & & 2 & X \\
\end{array}
\]

\[
\begin{align*}
+\Delta \pi_1 & = +2 & +1 \\
\pi_0 & = 0 & 0 & 0 & 0 & 0 & 0 \\
\pi_1 & = 2 & 1 & 0 & 0 & 0 & 0 \\
\Delta D & = \Delta \pi_1 r_1 + \Delta \pi_2 r_2 = 2 \times 11 + 1 \times 8 = 30 \\
D^0 & = 0 \\
D^1 & = 30
\end{align*}
\]
Directions: Continue the starting procedure. Subtract (add) the smallest value in each output node column from (to) every element in the output node column (row). The reduction is only indicated on this matrix, and the results are shown in matrix $c^2$ which follows.

$\begin{align*}
c^1 &= \begin{bmatrix} c^1_{ij} \end{bmatrix} \\
\end{align*}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>$-\Delta \pi_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>3</td>
<td>5</td>
<td>0*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td>0*</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>1</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>X</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>X</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>X</td>
<td></td>
<td></td>
<td>$-(-2)$</td>
</tr>
</tbody>
</table>

$\begin{align*}
\Delta \pi_1 &= \Delta \pi_{77} = -2x - 4 = 8 \\
D^1 &= 30 \\
D^2 &= 38 \\
\end{align*}$

$\begin{align*}
\pi_1^1 &= 2, 1, 0, 0, 0, 0, 0, 0, 0 \\
\pi_1^2 &= 2, 1, 0, 0, 0, 0, 0, -2 \\
\end{align*}$
Directions: The zeros which appear in this matrix determine a partial network. For those positions where zeros appear in the matrix, enter the positive $r_{ij}$ in the restriction matrix $r_{2a}$ which follows. The flow algorithm is used to maximize the flow in the partial network. After the maximum flow is made in $r_{2a}$ and $r_{2}$, the set $S = \{1, 2, 4\}$ is enumerated, and the rows and columns of $S$ are marked with '-' and '+' in this matrix. The value $\Delta \pi = 1$ is determined and the reduction of $c^2$ to $c^3$ is indicated. The next cost matrix appears as $c^3$. 

$$c^2 = \begin{bmatrix} 1 & c_{1j}^2 & \end{bmatrix}$$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>-\Delta \pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>3</td>
<td>5</td>
<td>0*</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>0*</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>X</td>
<td>1</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td>2</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>X</td>
<td>5</td>
<td>0*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\Delta D = \Delta \pi (r_1 + r_2 - r_{16}) = 1 (11 + 8 - 1) = 18$

$D^2 = 38$

$D^3 = 56$
\[ r^{2a} = \begin{bmatrix} r_{11} & \ldots & r_{1n} \end{bmatrix} \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>-1</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>15</td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]

\[ s_1 = [0^1, 0^2] \]

\[ s_2 = [1^6, 2^4] \]

\[ s_3 = [6^t] \]

Flow Path: 0, 1, 6, t

\[ \Delta f = \min \{11, 1, 15\} = 1 \]

Directions: The zeros of \( c^2 \) indicate where the positive \( r_{1j} \) are entered in this matrix. Also, the \( r_{1j} \) are entered in their proper positions in row 0 and column t. A partial flow \( \Delta f \) is indicated on this matrix, and the result follows as \( r^{2a} \).
\[ r^2 = \| r^2_{ij} \| \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>10</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

- \( S_0 = [0] \)
- \( S_1 = [0^1, 0^2] \)
- \( S_2 = [2^4] \)
- \( S_3 = \emptyset \), that is, \( S_3 \) is empty
- \( S = [1, 2, 4] \)

Directions: The set-labeling procedure terminates, and the set \( S \) is enumerated. The set \( S \) is used to start a new cycle of the iterative procedure. The reductions are indicated on the previous cost matrix \( c^2 \), and the result is shown in the following cost matrix \( c^3 \).
Directions: This matrix begins a new cycle of the iterative procedure. In $r^2$ a set $S$ was determined, and in $c^2$ the value $\Delta \pi$ was found. The reduction required by Step 1 of the iterative procedure (that is, subtracting $\Delta \pi$ from the "-" rows of $S$ and adding $\Delta \pi$ to the "+" rows of $S$) is indicated on $c^2$. The results are shown in this matrix. In this matrix, the "*" indicates that ACTION must be taken to convert $r$ into $r^{3a}$ so that $r^{3a}$ is a suitable starting point for making the maximum flow in the new partial network.
Directions: The "X" entries in $c^3$ indicate where action must be taken in $r^2$ to make this matrix a suitable starting point for maximizing the flow in the partial network described by the zeros and negative values of $c^3$. A partial flow is indicated on $r^{3a}$ and carried out in $r^3$. 

$\begin{array}{cccccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & X & 10 & -2 & X & X & X & X & X \\
1 & X & X & & & & & & 0 \\
2 & X & X & & & -2 & & & \\
3 & X & & X & & & & & \\
4 & X & & +2 & X & & -2 & & \\
5 & X & & & X & & & & 5 \\
7 & -2 & & & & +2 & & X & \\
\end{array}$

$s_0 = [0]$  
s_1 = [0^1, 0^2]$  
s_2 = [2^4]$  
s_3 = [4^7]$  
s_4 = [7^t]$  
Flow Path: 0, 2, 4, 7, t  
$\Delta f = \min [8, 2, 3, 4] = 2$
$$r^3 = \begin{bmatrix} r_{ij}^3 \end{bmatrix}$$

<table>
<thead>
<tr>
<th>t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>10</td>
<td>6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td>2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>[1]</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

$$S_0 = [0]$$

$$S_1 = [0^1, 0^2]$$

$$S_2 = \emptyset$$

$$S = [1, 2]$$

**Directions:** Set-labeling terminates with the set $S$. Indicate the reductions on $c^3$ for reducing $c^3$ to the next cost matrix $c^4$. 
$$c^4 = \sum c^4_{ij}$$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>$-\Delta \pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>0*</td>
<td>3</td>
<td></td>
<td>-3</td>
<td></td>
<td></td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>-2*</td>
<td>0*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-4</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>2</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-4</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>-4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>5</td>
<td>0</td>
<td>-4</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>X</td>
<td></td>
<td>-4</td>
</tr>
</tbody>
</table>

$\Delta \pi = \Delta \pi (r_1 + r_2 + r_7 - r_{16}) = 4 (11 + 8 - 4 - 1) = 56$

$D^4 = 88$

$D^5 = 144$

**Directions:** The reductions indicated on $c^3$ produce $c^4$. The "*" in $c^4$ indicates where action must be taken in $r^3$ to produce $r^4$ which follows. The procedure should now be clear.
\[ r_{i,j}^{4a} = \emptyset \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>10</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>0</td>
<td>+2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>X</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td>+2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>[1]</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>-2</td>
<td></td>
<td>2</td>
<td>+2</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0^1, 0^2] \]
\[ s_2 = [1^3, 2^5] \]
\[ s_3 = [5^7] \]
\[ s_4 = [7^t] \]

Flow Path: 0, 2, 5, 7, t

\[ \Delta f = \min[6, \infty, 5, 2] = 2 \]
\[ r^4 = \{ r_{ij}^4 \} \]

\[
\begin{array}{ccccccccc}
\hline
& t & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & X & 10 & 4 & X & X & X & X & X \\
1 & X & X & - & - & 0 & - & 2 & - \\
2 & X & X & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & X & X & X & X & X & X & X & X \\
5 & X & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
6 & 14 & [1] & X & X & X & X & X & X \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
\]

\[ s_0 = [0] \]
\[ s_1 = [0^1, 0^2] \]
\[ s_2 = [1^3, 2^5] \]
\[ s_3 = [5^7] \]
\[ s_4 = [7^4] \]
\[ s_5 = \emptyset \]

\[ s = [1, 2, 3, 4, 5, 7] \]
\[
c^5 = \begin{bmatrix} c_{1j}^5 \end{bmatrix}
\]

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & \Delta \pi \\
\hline
1 & X & 0 & 3 & -7 & -1 \\
2 & X & -2 & 0 & -1 \\
3 & X & 2 & 16 & -1 \\
4 & X & 1 & 1 & 0 \\
5 & X & 1 & 0 & -1 \\
6 & X & & & & & & \\
7 & 0^* & X & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
+\Delta \pi & +1 & +1 & +1 & +1 \\
\pi_1^5 & 9 & 8 & 4 & 5 & 4 & 0 & 2 \\
\pi_1^6 & 10 & 9 & 5 & 5 & 5 & 0 & 2 \\
\end{array}
\]

\[
\Delta D = \Delta \pi \left( r_1 + r_2 + r_{16} + r_{24} - r_{57} \right) = 1 (11 + 8 - 1 - 2 - 5) = 11
\]

\[
D^5 = 144
\]

\[
D^6 = 155
\]
\[ r_{5a}^n = \prod_{ij} r_{ij}^{5a} \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>10</td>
<td>4(-3)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>(\infty)</td>
<td>0</td>
<td>-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>X</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>+3</td>
<td>2</td>
<td>-3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(-3)</td>
<td>14</td>
<td>[1]</td>
<td>X</td>
<td>+3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
<td>+3</td>
<td>-3</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ S_0 = [0] \]
\[ S_1 = [0^1, 0^2] \]
\[ S_2 = [1^3, 2^5] \]
\[ S_3 = [5^7] \]
\[ S_4 = [7^4, 7^6] \]
\[ S_5 = [6^t] \]

Flow Path: 0, 2, 5, 7, 6, t

\[ \Delta f = \min \{4, \infty, 3, \infty, 14\} = 3 \]
\[ r^5 \bullet \begin{array}{cccccccc}
& 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & X & X & X & X & X & X & X \\
1 & X & X & - & 0 & -5 & & \\
2 & X & X & 0 & -5 & & & \\
3 & X & X & & & & & \\
5 & X & 5 & X & 0 & & & \\
6 & 11 & [1] & X & 3 & & & \\
7 & 0 & 2 & 5 & -3 & X & & \\
\end{array} \]

\[ s_0 = [0] \]
\[ s_1 = [0^1, 0^2] \]
\[ s_2 = [1^3, 2^5] \]
\[ s_3 = \emptyset \]

\[ s = [1, 2, 3, 5] \]
\[ e^6 = \| e_{ij}^6 \| \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>(-\Delta n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>2</td>
<td>-8</td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>-3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>0*</td>
<td>-1*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \Delta D = \Delta n \left( r_1 - r_{1c} \right) = 1 \left( 11 - 1 \right) = 10 \]

\[ D^6 = 155 \]

\[ D^7 = 165 \]
\[ r^6_{ij} = \begin{bmatrix} r_{ij} \end{bmatrix} \]

<table>
<thead>
<tr>
<th></th>
<th>t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>10</td>
<td>-1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>-1</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>-1</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>-1</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>X</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>+1</td>
<td>X</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>[1]</td>
<td>+1</td>
<td>X</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
<td>[5]</td>
<td>-3</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0^1, 0^2] \]
\[ s_2 = [1^3, 2^5] \]
\[ s_3 = [5^6] \]
\[ s_4 = [6^t] \]

Flow Path: 0, 2, 5, 6, t

\[ \Delta f = \min [1, -, -, 11] = 1 \]
\[ r^6 = \left[ r^6_{ij} \right] \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>10</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td>6</td>
<td></td>
<td>X</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td></td>
<td></td>
<td>1</td>
<td>X</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>[5]</td>
<td>-3</td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0, 1] \]
\[ s_2 = [1, 3] \]
\[ s_3 = \emptyset \]
\[ s = [1, 3] \]
\[ c^7 = \{ c^7_{ij} \} \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>-(\Delta n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>-9</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>-3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>0</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>0</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+\(\Delta n\) = 1, 1, 1, 1

\[ \pi^7_1 = 11, 9, 6, 5, 5, 0, 2 \]

\[ \pi^8_1 = 12, 9, 7, 6, 5, 0, 2 \]

\[ \Delta D = \Delta n \left( r_{1} - r_{16} + r_{24} - r_{47} \right) = 1 \left( 11 - 1 + 2 - 3 \right) = 9 \]

\[ D^7 = 165 \]

\[ D^8 = 174 \]
\[ r_7^{a} = \sum r_{ij}^{7a} \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td></td>
<td>-1</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>0</td>
<td>-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>+1</td>
<td>X</td>
<td>-1</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>+1</td>
<td>X</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>6</td>
<td>X</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>[1]</td>
<td>1</td>
<td>X</td>
<td>3</td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>+1</td>
<td>[5]</td>
<td>-1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \Delta f = \min [10, -9, 1, -5, 10] = 1 \]

Flow Path: 0, 1, 3, 4, 7, 6, t

\[ S_0 = [0] \]
\[ S_1 = [0, 1] \]
\[ S_2 = [1, 3] \]
\[ S_3 = [3, 4] \]
\[ S_4 = [4, 7] \]
\[ S_5 = [7, 6] \]
\[ S_6 = [6, 5] \]
\[ r^7 = \begin{bmatrix} r_{ij} \end{bmatrix} \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>9</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>(-1)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>(-6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>6</td>
<td>X</td>
<td>(-1)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>[1]</td>
<td>1</td>
<td>X</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td>3</td>
<td>[5]</td>
<td>(-4)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0, 1] \]
\[ s_2 = [1, 3] \]
\[ s_3 = [3, 4] \]
\[ s_4 = \emptyset \]

\[ S = [1, 3, 4] \]
\[ e^8 = \begin{bmatrix} c_{ij}^8 \end{bmatrix} \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>-\Delta \pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td></td>
<td>-10</td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>-2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>1</td>
<td>0*</td>
<td>-1*</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

\[ \Delta \pi = \Delta \pi \left( r_{16} + r_{24} - r_{46} - r_{47} \right) = 1 \left( 11 - 1 + 2 - 5 - 3 \right) = 4 \]

\[ D^8 = 174 \]

\[ D^9 = 178 \]
\[ \mathbf{r}^{8a} = \left[ \mathbf{r}_{ij}^{8a} \right] \]

<table>
<thead>
<tr>
<th>t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>-5</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>-5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>-6</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>+5</td>
<td>X</td>
<td>8</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>1</td>
<td>[2]</td>
<td>X</td>
<td>5</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>6</td>
<td>X</td>
<td>X</td>
<td>-1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>[1]</td>
<td>+5</td>
<td>1</td>
<td>X</td>
<td>4</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>[3]</td>
<td>[5]</td>
<td>-4</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0 1] \]
\[ s_2 = [1 3] \]
\[ s_3 = [3 4] \]
\[ s_4 = [4 6] \]
\[ s_5 = [6 t] \]

Flow Path: 0, 1, 3, 4, 6, t

\[ \Delta f = \min \{ 9, -, 8, 5, 9 \} = 5 \]
\[
r^8 = \begin{bmatrix} \mathbf{r}^8_{ij} \end{bmatrix}
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>4</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>-6</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>0</td>
<td>-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>6</td>
<td>X</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>6</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>6</td>
<td></td>
<td>X</td>
<td>-1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>[1]</td>
<td>5</td>
<td>1</td>
<td>X</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td></td>
<td>[3]</td>
<td>[5]</td>
<td>-4</td>
<td>X</td>
</tr>
</tbody>
</table>

\[
S_0 = [0]
S_1 = [0, 1]
S_2 = [1, 3]
S_3 = [3, 4]
S_4 = \emptyset
\]
\[
S = [1, 3, 4]
\]
\[ c^9 = \begin{bmatrix} c^9_{1j} \end{bmatrix} \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>-(\Delta n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td></td>
<td>-11</td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>0</td>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>0*</td>
<td>-1*</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>0</td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+\(\Delta n\)  
9  
\[ \pi_1^9 = 13 \quad 9 \quad 8 \quad 7 \quad 5 \quad 0 \quad 2 \]
\[ \pi_1^{10} = 14 \quad 9 \quad 9 \quad 7 \quad 5 \quad 0 \quad 2 \]
\[ \Delta D = \Delta n (r_1-r_{15}-r_{34}) = 1 (11-1-9) = 1 \]
\[ D^9 = 178 \]
\[ D^{10} = 179 \]
\[ r^g_a = \{ r_{ij}^g \} \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>-3</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>-3</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>0</td>
<td>-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>+3</td>
<td>X</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>+3</td>
<td>X</td>
<td>-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td>+3</td>
<td>X</td>
<td>-3</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>-3</td>
<td>[1]</td>
<td>[5]</td>
<td>1</td>
<td>+3</td>
<td>X</td>
<td>4</td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0, 1] \]
\[ s_2 = [1, 3] \]
\[ s_3 = [3, 4] \]
\[ s_4 = [4, 5] \]
\[ s_5 = [5, 2, 5, 6] \]
\[ s_6 = [6, t] \]

Flow Path: 0, 1, 3, 4, 5, 6, t

\[ \Delta f = \text{min}[4, =, 3, =, =, 4] = 3 \]
\[ r^9 = \| r^9_{ij} \| \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>-2</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>9</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>[2]</td>
<td>9</td>
<td>X</td>
<td>-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>\epsilon</td>
<td>3</td>
<td>X</td>
<td>-4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>[1]</td>
<td></td>
<td></td>
<td>[5]</td>
<td>4</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>[3]</td>
<td>[5]</td>
<td>-4</td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0, 1] \]
\[ s_2 = [1, 3] \]
\[ s_3 = \emptyset \]

\[ s = [1, 3] \]
\[ c^{10} = c^* = \begin{bmatrix} c_{ij}^* \end{bmatrix} \]

Final Reduced Cost Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0*</td>
<td></td>
<td></td>
<td>-12</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td>-1*</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>0</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

\[ \pi_{i1}^{10} = \pi_{i1}^* = 14 \quad 9 \quad 9 \quad 7 \quad 5 \quad 0 \quad 2 \]

\[ D^{10} = D^* = 179 \]

\[ \begin{align*}
\pi_{16}^* &= -12 \\
\pi_{24}^* &= -1 \\
\pi_{34}^* &= -1 \\
\pi_{46}^* &= -1 \\
\pi_{47}^* &= -2 \\
\pi_{57}^* &= -1 \\
\end{align*} \]

Directions: \( c^{10} \) is the final reduced cost matrix because the flow determined in \( r^{10} \) and \( r^{10} \) is a maximum flow, and a solution matrix has been achieved. No further reductions can be made in the cost matrix. Notice that the \( \pi_{i1} \) are equal to the negative values in the positions \( ij \). All other \( \pi_{ij} \) are zero. \( D^* \) is the maximum value of the Dual.
\[ r^{10a} = [ ] r_{ij}^{10a} [ ] \]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>-1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>-6</td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>9</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>+1</td>
<td>[2]</td>
<td>[9]</td>
<td>X</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>ε</td>
<td>3</td>
<td>+1</td>
<td>X</td>
<td>-1</td>
<td>-4</td>
</tr>
<tr>
<td>6</td>
<td>l</td>
<td>[1]</td>
<td>[5]</td>
<td>4</td>
<td>X</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>[3]</td>
<td>[5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ s_0 = [0] \]
\[ s_1 = [0, 1] \]
\[ s_2 = [1, 3, 4] \]
\[ s_3 = [4, 5] \]
\[ s_4 = [5, 6] \]
\[ s_5 = [6, t] \]

Flow Path: 0, 1, 4, 5, 6, t

\[ \Delta f = \min [1, \infty, \infty, 1] = 1 \]
\[ r^{10} = r^* = \begin{bmatrix} r_{ij} \end{bmatrix} \]

Final Reduced Restriction Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>(-9)</td>
<td>(-1)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>(-\epsilon)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>9</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>1</td>
<td>[2]</td>
<td>[9]</td>
<td>X</td>
<td>(-4)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>6</td>
<td>4</td>
<td>X</td>
<td>(-5)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>[1]</td>
<td>[5]</td>
<td>5</td>
<td>X</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td>[3]</td>
<td>[5]</td>
<td>(-4)</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

\[ S_0 = [0] \]

\[ S_1 = \emptyset. \text{ Terminate.} \]

Directions: \( S_1 \) is empty and, therefore, the procedure terminates. A maximum and optimal flow is defined in the matrix which follows.
\[ x^* = \| x_{ij}^* \| \]

Optimizing Flow Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>11</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Directions: Determine this matrix by the relation

\[ x_{ij}^* = \max \left[ r_{ij} - r_{ij}^*, 0 \right] \]
\[
\begin{bmatrix}
  c_{ij} & x^*_{ij}
\end{bmatrix}
\]

\[
\begin{array}{cccccccc}
  \text{t} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
  0 & X & 0x11 & 0x8 & X & X & X & X \\
  1 & X & X & 5x9 & 7x1 & 2x1 & & & \\
  2 & X & X & 1x2 & 4x6 & & & & \\
  3 & X & X & 1x9 & & & & & \\
  4 & X & X & 2x4 & 6x5 & 3x3 & & & \\
  5 & X & X & 5x5 & 2x5 & & & & \\
  6 & 0x15 & & & & & X & & \\
  7 & 0x4 & & & 2x4 & X & & & \\
\end{array}
\]

\[\min\ C(x) = C(x^*) = \Sigma_i \Sigma_j c_{ij} x^*_{ij} = 179\]

\[D^* = \Sigma_i \pi^*_i r_i + \Sigma_j \pi^*_j r_{ij} = 179\]

\[C(x^*) = D^* = 179.\]

Directions: The minimum cost solution is gotten by multiplying the cost by the flow in each arc of the network. Note that the minimum cost is equal to the maximum value of the dual, and so the solution checks.
The Transhipment Algorithm and The Flow Algorithm developed in the previous chapters are efficient and simple hand-computation procedures for finding the maximum flow in a restricted network, and for solving the Assignment Problem, the Transportation Problem, and the Transhipment Problem. In addition, by making the costs equal to the distances between the various nodes of a network, and by having a unit input and output quantity, the Transhipment Algorithm will solve the problem of finding the Shortest Route Through a Network.

Generalizations. The usefulness of these algorithms can be extended by some generalizations on the procedure which should not be too difficult to achieve. Once the present procedure is understood, it is only necessary to present "understandable instructions" for generalizing the algorithms along some of the following lines.

By splitting each node $i$ into two fictitious nodes $i'$ and $i''$, the procedure can readily be generalized by treating a handling capacity and cost for the node $i$ as if it were specified for the fictitious arc $i'i''$. This might be more simply accomplished by entering the node handling capacity and cost along the main diagonal of the matrices.

To treat node handling costs, Paul Pepper has suggested that half of the handling cost for each node $i$ be added to the shipping cost for every arc connected to the node $i$. This would only require the addition of another fictitious origin $0'$ and terminal $t'$ to the augmented network.
A second generalization could be made by permitting two-way real flow in the arcs of the network. This could be accomplished by drawing a diagonal line (from upper-left to lower-right) in each "block" \(ij\) of the restriction matrix. When a flow \(\Delta f\) is made through the position \(ij\), the restriction "below the diagonal" could be reduced in the forward direction of flow, and the fictitious capacity \(\Delta f\) (for the backward flow direction) could be added "above the diagonal." This would leave the "block" \(ji\) open for making real flows in the backward direction. With this generalization, the unit shipping cost and the capacity restriction for the direction \(ij\) could be different from the values given for the direction \(ji\). It might also be possible to place an additional restriction on the total real flow in both directions in the arcs.

Other problems. Dantzig mentions several other problems which are "combinatorial in nature and easy to formulate, but that mathematicians have had only partial success in solving. These arise often in the form of discrete-programming problems..." [17, p 266]. One of these is a multi-stage machine scheduling problem, for which a graphical method of solution has been proposed by Johnson and Dantzig [67], and in which both the primal and dual problems are solved simultaneously by a method involving only intersections and rotations of straight lines.

Another of these fundamental but unsolved problems is the Fixed-Charge Problem, in which various activities have a fixed charge (for example, set-up time costs) if operating at a positive level. Although this is basically a non-linear problem, Hirsch and Dantzig [63] showed that, under special circumstances, a fixed-charge problem can be reduced to an ordinary linear programming problem.
Another of the problems mentioned by Dantzig is the travelling-salesman problem. This was solved by Robinson [90] in a special form, and again by Dantzig, Fulkerson, and Johnson [34]. The problem is often stated as follows: find the shortest route for a salesman starting from Washington, visiting all the state capitols, and then returning to Washington. This is a problem of finding the shortest route through a network and, as mentioned above, can be solved by the present method.

There are several general areas of linear programming where work is being done and where further results are needed, especially in "performing sensitivity analyses for the coefficients of the problem... [and in] ... development of methods that are particularly useful for dynamic problems..." [93, p 157]. And "perhaps one of the most exciting ideas to date is a possibility that the solution to a dynamic problem might be obtained as a by-product of an iterative procedure for solving a steady state problem" [23, p 18].

Mitten [81] has made progress on methods for performing sensitivity analyses for the coefficients of the problem. He considers the problem of determining how much the cost "estimates," given in the original statement of a problem, must change to affect the overall optimality of the solution. He treats two cases: first, if no flow has been made through the position $ij$, what decrease in $c_{ij}$ will require a flow to be made through $ij$; and second, if a flow has been made through $ij$, what increase in $c_{ij}$ will prohibit a flow through $ij$. Mitten's methods are not easily generalized to the case of network transhipment; however, they provide a starting point for further work. Some such sensitivity analyses is required for the Transhipment Algorithm.
Ford and Fulkerson [44] [45] have made progress on the computation of maximal dynamic flows in networks, and they have also suggested a computational procedure for multi-commodity [non-linear] networks [51]. Robacker [88] has generalized the max-flow min-cut theorem for multi-commodity networks, and Bellman, as early as 1951, made a computation for a very simple multi-commodity network [4].

Other problems which appear to be close to the area of present interest, are periodically appearing in the literature, for example, the Caterer Problem of Jacobs [66], a solution to which has been provided by Gaddum [58]. And the "Bottleneck" Problem of Fulkerson, Glicksberg, and Gross [57] seems as if it could be solved by the present method. The bottleneck problem is stated briefly as follows: suppose there are $n$ jobs to be performed in sequence on an item in a production line and we have $n$ men available to be assigned to the various jobs. If man $i$ in job $j$ can process $a_{ij}$ items per unit time, assign the men so as to maximize the rate of production of the end item.

**Summary.** In all of the above problems, the primary difficulty has been in the complication of the proposed solution procedures—complication sufficient to require the use of high-speed digital computers. In many areas, the efficiency and simplicity of the algorithms described in this work will permit hand-computation for these problems. For example, in a non-linear problem involving step-discontinuities or "price breaks" in the costs, the proposed algorithm is simple enough to enable the problem-solver to keep track of when to change the cost values. These algorithms should be useful for many such applications in addition to their use in solving transhipment and flow problems.
BIBLIOGRAPHY


12 Dantzig, George B. "Application of the Simplex Method to a Transportation Problem," Chapter XXIII of Activity Analysis of Production and Allocation [see 71], RAND R-193, pp. 359-373, June, 1951, or Wiley and Sons, New York, 1951.


"Maximization of a Linear Function of Variables Subject to Linear Inequalities," Chapter XXI of Activity Analysis of Production and Allocation [see 71], RAND R-193, 1951.


55 \textit{A Network-Flow Feasibility Theorem and Combinatorial Applications}, RAND RM-2159, April 21, 1958.


Kalaba, Robert E. "Linear Programming," RAND P-950, October 1, 1956.


Mitten, Loring G. *Programming Course Notes* (unpublished), The Ohio State University, 1958.


I, Edward A. Brown, was born in Brooklyn, New York, on October 4, 1928. I received my secondary school education at The Brooklyn Preparatory, New York, and The Cranwell Preparatory, Lenox, Massachusetts. My undergraduate training was taken in the Honors Program at Fordham University, New York, where I was granted the Bachelor of Science degree, Cum Laude, in 1951. My areas of specialization at Fordham were Physics and Philosophy.

From 1951 until 1958, I served on active duty with the United States Air Force in the grades Second Lieutenant through Captain. During that time, I spent three years as a Nuclear Supervisor in the Armed Forces Atomic Weapons Project, two years on flying duty and as a Jet Instructor Pilot, and two years as a Missile System Analyst and Operations Research Analyst at the Wright Air Development Center, Dayton, Ohio.

While on duty in Albuquerque, New Mexico, I completed the requirements for the Master of Science degree in Electronics, and was awarded the degree in 1955 by the University of New Mexico.

I began course work in Operations Research at The Ohio State Graduate Center at Wright Field. I was appointed "Fellow" Of The Ohio State University for the academic year 1958-1959, and held the Fellowship while completing the requirements for the degree Doctor of Philosophy.