CWI Tract 4

Minimal cost flow in processing networks, a primal approach

J. Koene

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

1980 Mathematics Subject Classification: 90B10, 90C35
ISBN 90 6196 270 6

PREFACE

In the past ten to fifteen years a vast amount of work has been done on the development of efficient algorithms and associated implementations for solving network flow problems. In these procedures both time and storage requirements are generally speaking much smaller than in the standard LP-approaches. This is accomplished by exploiting the network structure. The main advantage in this respect is of course the fact that there are much more possibilities to model large real-life situations.

But there is at least one other major reason for the increased interest in network flows. For both the OR-analyst and the model-user a network model is "much more visually informative and intuitively appealing than perhaps any other OR-model". (I quote Golden, Ball & Bodin (1981).) In my opinion the communication with model-users is for an OR-consultant as important as building the right model. By visualization of the basic concepts of a model (drawing network diagrams) the communication can be improved.

Knowledge of the structure is essential for getting insight in the problem at hand, exploiting it is important for the development of efficient computer codes, including input/output facilities such as matrixgenerators and report-writers.

In this book we will consider processing networks. Characteristic for a processing network is the possibility that a given flow splits up proportionally in a number of components (a refining process), or conversely, that a number of components is blended in given proportions (a blending process).

Processing networks are hardly considered in the literature in spite of a vast amount of possible applications. This book is intended as a first in-depth treatment of this type of problems. It can be divided into four main parts.

The first two chapters are meant as background information. An overview is presented of the historical developments in network flow programming. Moreover, an outline is presented of the primal simplex algorithm for general linear programming problems, as well as specializations of this algorithm for pure and generalized network problems. Many of the basic ideas in these

procedures are essential in (understanding) the development of processing network algorithms.

The main issue of this monograph is the presentation of primal simplex based solution procedures for several kinds of processing network problems (chapters 3, 4 and 6). These algorithms exploit the special basis structure in such a way that many of the simplex computations can be performed by graph theoretic means.

The above mentioned procedures have been set up from the viewpoint that processing networks are generalizations of pure and generalized networks. On the other hand we might consider processing network problems as LP problems with a special structure. The relation between processing networks and general LP's will be discussed in chapter 5.

Finally the potential applicability of processing networks is outlined in chapter 7.

At the time this book was written nothing had happened with respect to the implementation of the algorithms described. Meanwhile processing network codes are being developed at the Eindhoven University of Technology under supervision of Prof.dr. J.F. Benders. Other research activities in this field have been reported by a.o. Dr. D.L. Adolphson, Dr. R.D. McBride and Dr. M. Enquist.

There are several people who contributed to a large extent to the realization of this book and to whom I would like to express my sincere graditude. First of all I would like to thank Prof.dr. J.F. Benders, my first promotor. To consider processing networks as an important and fascinating topic for research was his suggestion. And right from the start of my research efforts I experienced his continuous guidance and encouragements. In a later stage I received fruitful comments from Prof.dr. J. Wessels, my second promotor, Prof.dr. A.H.G. Rinnooy Kan, Prof.dr. G.J. Veltkamp and Prof.dr. F. Lootsma. I want to express my thanks to the Mathematical Centre for the opportunity to publish this work as a CWI Tract.

CONTENTS

1. INTRODUCTION, SURVEY AND CONCLUSIONS

1.1. Introduction

Many managerial and industrial problems encountered in practice show a total or partial network flow character. Most of them can be modelled adequately as linear models, in which both continuous and integer activities may play a role.
With respect to the continuous case such models are Linear Programming models which of course can be solved by standard LP-programs. However, such programs do not take full advantage of the network structure. This is one of the reasons why in the past decades much research has been done on how a specific network structure can be employed more efficiently in solving such problems.
Knowing structure is essential for getting insight in the problem at hand. Exploiting structure is important, not only for the development of solution procedures which are faster or require less memory capacity than the present day standard procedures, but also for the design of a proper data base and for adequate manipulation and reporting instructions of LP-based decision support systems.

This monograph is concerned with an important type of network problems often encountered in practice. They are called *processing network* problems. Before explaining in Subsection 1.1.2 what processing networks are, where they arise and how we intend to analyze and solve processing network problems, the history of network problems is briefly sketched, focussing primarily on so-called pure and generalized networks. These two types play an important role in the subsequent discussions.

1.1.1. Historical background

The real interest in network models started from the work of KANTOROVICH [1939], HITCHCOCK [1941] and KOOPMANS [1947] who studied transportation problems. The more general transshipment problem was stated somewhat later, in fact already by KANTOROVICH & GAVURIN [1949].

In the 1950's and 1960's the emphasis lay on solution techniques to solve such problems and on the development of more general network models and associated solution procedures.
Three classes of models are:

A. Pure Networks

DANTZIG [1951] presented a specification of the Simplex algorithm for the transportation problem, in which the basis structure is exploited. ORDEN [1956] extended these results to the transshipment problem. Only slightly different from the transshipment problem is the so-called minimal cost flow problem in a pure network (see LAWLER [1976]). The latter, often just called a pure network problem, can be stated as follows:

Given a network, consisting of nodes and directed arcs between certain pairs of nodes,
the cost for transporting a unit of flow along each arc,
the demands and supplies in each node,
determine flows in the network such that they satisfy the demands from the supplies at minimal total cost,
whenever

1. the flow is conserved throughout the network, that is to say, both in nodes and on arcs (no losses or gains in transporting flow along arcs);
2. the flow in each arc is in between given lower and upper bounds for that arc (capacity bounds).

A well-known and useful property of pure networks is total unimodularity, which guarantees that basic solutions are integer valued, provided that the demands, supplies and capacity bounds are integers.
In its most general setting, pure network problems can be seen as LP-problems in which the coefficient matrix has at most two nonzero elements in each column, with the additional requirement that the column sum of each column with two nonzero entries equals zero.

Some relevant solution procedures developed in this period are:

primal-dual : FORD & FULKERSON [1957],
out-of-kilter : FULKERSON [1961],
dual : BALAS & HAMMER [1962],
negative cycle: KLEIN [1967].

B. Generalized networks

Generalized networks are also known as networks with gains. They differ in only one aspect from pure networks: in transporting flow through the network flow may be lost or gained. Usually one considers networks where flow is conserved in nodes, but not on arcs. Associated with each arc is a so-called multiplier or gain. In physical processes mainly losses occur (leakage, damage), whereas true gains are found in certain business applications (e.g. cash flow models). Among the pioneers in this field are KANTOROVICH [1939], FERGUSON & DANTZIG [1954], MARKOWITZ [1954], EISEMANN [1964] and BALAS [1966]. They considered generalized transportation problems. JEWELL [1962] proposed a primal-dual approach for the general case, allowing positive as well as negative multipliers. In its most general setting generalized network problems can be considered as LP-problems in which the coefficient matrix has at most two nonzero entries in each column.

C. Multicommodity networks

Multicommodity networks arise when several items (commodities) share capacitated arcs in a network. They can be regarded as pure or generalized networks with generalized upper bounds.
Some of the solution procedures for multicommodity network problems are:

decomposition : ROBACKER [1956],
FORD & FULKERSON [1958],
TOMLIN [1966],
primal-dual : JEWELL [1966],
primal basis partitioning: SAIGAL [1967].

In the 1970's and early 1980's much work has been done on:

(a) implementation and computational testing of known algorithms,
(b) exploring the field of applicability,
(c) new theoretical developments,
(d) problems with embedded pure or generalized network structure.

These aspects are discussed next in some more detail.

(a) implementation and computational testing of known algorithms.

With respect to pure networks in the early 1970's codes were developed by a.o. BENNINGTON [1972], BARR, GLOVER & KLINGMAN [1974], out-of-kilter / primal-dual, GLOVER, KLINGMAN & NAPIER [1972], dual, and GLOVER, KARNEY & KLINGMAN [1974], primal. Computational comparisons, described a.o. in the latter reference, led to a quite general believe that primal Simplex solution procedures are superior to other approaches, both with respect to time and storage requirements. Until then out-of-kilter / primal-dual procedures were thought to perform best. The "Primal Revolution" had begun.
Primal Simplex codes for generalized networks were developed as well: MAURRAS [1972], GLOVER, KLINGMAN & STUTZ [1973].
In implementing such algorithms much attention was paid to finding efficient datastructures a.o. to store the basis, finding good starting bases, pivot selection criteria, the use of mirror arcs, distance labels, etc. References are: GLOVER, KARNEY & KLINGMAN [1974], BRADLEY, BROWN & GRAVES [1977], GLOVER & KLINGMAN [1978a], GLOVER, HULTZ, KLINGMAN & STUTZ [1978], ELAM, GLOVER & KLINGMAN [1979].
The current primal codes for pure and generalized network problems have several appealing advantages over standard LP approaches (see the just mentioned papers):

1. they perform much faster, for pure networks up to 200 times, for generalized networks about 50 times faster than APEX III;
2. they require much less storage capacity;
3. because of the special basis structure they work with the original data, thus eliminating or reducing round-off errors.

(b) exploring the field of applicability

In itself the applicability potential of pure and generalized networks has been known for a long time, but the success of the primal codes opened up the possibility to consider many real-life, large size problems. Currently systems are developed which challenges one's imagination, see e.g. BARR & TURNER [1981] who consider a file merging solution system designed to accommodate problems with up to 50.000 constraints and 65 million activities.
To mention some other fields of applicability:

Pure networks: transportation of goods, design of communication and pipeline systems, assignment of men to jobs, bid evaluation, production planning.
Generalized networks: the "multiplier facility" is capable to model two types of situations (see GLOVER, HULTZ, KLINGMAN & STUTZ [1978]):

1. to modify the amount of flow of some item. In this way situations involving evaporation, seepage, deterioration, breeding, interest rates, sewage treatment, purification processes, machine efficiencies and structural strength design can be modelled.
2. to transform the flow from one type of good to another: processes of manufacturing, conversions of fuel to energy, blending, crew scheduling, allocating manpower to job requirements, currency exchanges, production.

For a further discussion of the applicability of pure and generalized networks, see e.g. JEWELL [1962] and GLOVER & KLINGMAN [1977, 1978a].

By now, both pure and generalized network models are more or less accepted as fundamental modelling tools. This is not only due to the advantages mentioned under (a) but to a large extent also because "network models are more visually informative and intuitively appealing than other OR-models", GOLDEN, BALL & BODIN [1981], see also GLOVER & KLINGMAN [1975, 1977].

(c) new theoretical developments

Just a few new theoretical developments are mentioned.
EDMONDS & KARP [1972] discussed the pure network problem from a computational complexity point of view. Moreover, they proposed the first polynomial algorithm for the maximal flow problem in a pure network. For further developments on max flow problems, see GLOVER & KLINGMAN [1980].
BALACHANDRAN, SRINIVASAN & THOMPSON (see - [1981]) developed an "operator" theory of parametric programming for pure and generalized transportation problems.
In pure and generalized networks degeneracy was taken into consideration. CUNNINGHAM [1976, 1979] and ELAM, GLOVER & KLINGMAN [1979] presented "pivot row" selection rules which prevent cycling in pure networks and generalized networks with positive multipliers, respectively. Implementation of such rules in actual codes show some reduction in required solution times.
ADOLPHSON [1980], building on the work of FONG & SRINIVASAN [1977], recently proposed a nondegenerate primal Simplex method for pure networks. Although degenerate steps are excluded, the steps of this algorithm require shortest path information and are therefore more time consuming than in the usual procedures.

It is stressed that these degeneracy considerations are not only of theoretical importance. Degeneracy is a severe practical problem: up to 90% of the Simplex steps in large scale applications are degenerate in the current codes.

(d) problems with embedded pure or generalized network structure

The success of pure and generalized networks led to a general belief that for LP's as well as for (mixed) integer LP's with embedded pure or generalized network structure good computational results could be obtained by extending the ideas on which the primal approaches for pure and generalized network problems are based. An increasing interest can be observed for the following questions:

1. how to exploit embedded pure or generalized network structure.
Basis partitioning, rather than decomposition or other approaches, seems to be the right way to do this (cf. KENNINGTON [1978]). Primal basis partitioning procedures were suggested for different types of problems:
Multicommodity networks, HARTMAN & LASDON [1972], KENNINGTON [1977].
Pure networks with side constraints: KLINGMAN & RUSSELL [1975], CHEN & SAIGAL [1977].
Generalized networks with side constraints: HULTZ & KLINGMAN [1976].
Pure networks with side constraints and side activities: GLOVER & KLINGMAN [1981]. As they put it: "Side constraints arise for instance from economies of scale, limitations on shared resources, multiple criteria or from the outputs of subdivisions to meet overall demands. Side activities (columns) arise from activities which involve different time periods, production alternatives (e.g. refinery activities) or which involve different subdivisions (e.g. assembly)."

REMARK 1.1.1. In the above lines words as "subdivisions, refinery activities and assembly" are underlined because such type of processes fall exactly within the scope of this monograph. □

It is characteristic for these approaches that the pure or generalized network part is extracted from the basis. In each step of the Simplex algorithm there is an interaction between this "transportation" part and the so-called working basis. Sometimes the size of this working basis is fixed, at other times it varies dynamically and then one tries to keep it as small as possible.

Since in solving (mixed) integer problems, the continuous LP-formulation plays an essential role as a subproblem (e.g. Branch & Bound, BENDERS' [1962] decomposition) there is a great interest in network formulations and network solution techniques, see e.g. GEOFFRION & GRAVES [1974], GLOVER & KLINGMAN [1978a], GLOVER & MULVEY [1980], VAN NUNEN & BENDERS [1981]. Preliminary computational results on these embedded network problems are encouraging, but much work has to be done before general conclusions can be drawn.

2. how to detect hidden pure or generalized network structure, see BIXBY [1981], BROWN & WRIGHT [1981], GUNAWARDANE, HOFF & SCHRAGE [1981] and SCHRAGE [1981].

3. how to create pure or generalized network structure, GLOVER [1981].

Future research directions in network optimization are indicated by CHARNES, KARNEY, KLINGMAN & STUTZ [1975] and GOLDEN, BALL & BODIN [1981]. Finally, it is remarked that surveys on networks are written by ELMAGHRABY [1970] and BRADLEY [1975].

1.1.2. Scope of this monograph

With the above mentioned developments in mind, we consider an important class of network problems, called *processing network* problems. They carry this name because they are able to model certain refining and blending processes which a.o. arise in production planning environments in the process industry. Processing networks are more general than pure or generalized networks in these two respects:

1. they allow the possibility that a given flow splits up in several components in given proportions. For quite obvious reasons such a process is called a *refining* process. Schematically it is depicted in Figure 1.1.1.

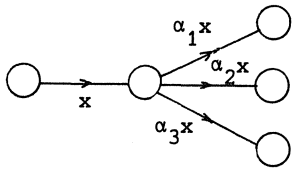


Figure 1.1.1. A refining process ($\sum_i \alpha_i = 1;\ \alpha_i > 0,\ \forall_i$).

2. they allow the possibility that several components are blended in given proportions. This is called a *blending* process; it is depicted in Figure 1.1.2.

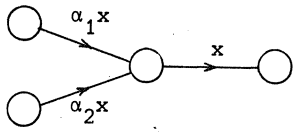


Figure 1.1.2. A blending process ($\sum_i \alpha_i = 1;\ \alpha_i > 0,\ \forall_i$).

Arcs in a processing network which do not take part in some proportionality requirement can be seen as describing a simple "transportation process". So there are three types of processes in a processing network: refining, blending and transportation.

Two classes of processing networks are distinguished:

a. Pure Processing Networks, where the same conditions hold as in pure networks: conservation of flow and capacity bounds on arcs.
b. Generalized Processing Networks, where the same conditions hold as in generalized networks: conservation of flow in nodes, but not necessarily on arcs, and capacity bounds on arcs.

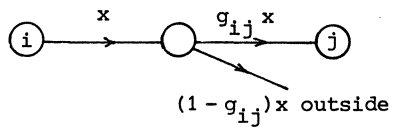
The processing network structure comes up in quite a number of situations:

1. in production planning in the process industry. In the petrochemical industry both refining (destillation) and blending "on receipt" takes place. Also reference is made to the milk industry where, e.g., raw milk is split in proportional amounts of consumption milk, butter and cheese, GEURTS [1980].
2. in assembly models the fact that parts are "blended" in given proportions is essential. STEINBERG & NAPIER [1980] describe a mixed integer network model for a lot sizing problem in material requirements planning (MRP).
3. in energy models not only conversion processes (generalized networks) take place, but also blending (for instance, different types of gas must be mixed in given proportions) and refining (oil sector!) occur. Examples of network energy models are BOONEKAMP, KOENDERS & VAN OOSTVOORN [1979], model SELPE and the models PIES and BESOM, a.o. described in MANNE, RICHELS & WEYNANT [1979].

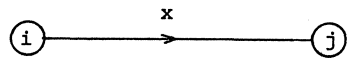
4. in economic models, such as input/output models, the outputs from each industry are directly proportional to its inputs.

It is remarked that generalized networks with positive multipliers can readily be seen as a special type of pure processing networks. This observation is already described in SCHAEFER [1978].
Let (i,j) denote an arc from node i to node j in a generalized network, the associated multiplier is given by $g_{ij} > 0$. Three cases with corresponding processes can be distinguished:

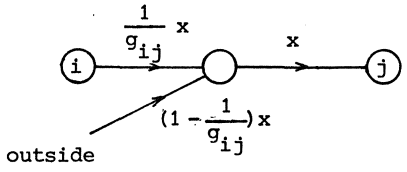
(a) $0 < g_{ij} < 1$, refining process



(b) $g_{ij} = 1$, pure transportation process



(c) $g_{ij} > 1$, blending process



In many of the sketched practical situations (relatively few) additional requirements must be satisfied, which lead to additional linear constraints (side constraints) in the model (cf. Subsection 1.1.1).

From the description of processing networks given thus far it is immediately clear that they can be seen as pure or generalized networks with side constraints, which arise from the proportionality requirements of the refining and blending processes. Therefore, procedures of CHEN & SAIGAL [1977] and HULTZ & KLINGMAN [1976] can be used to solve them, thus exploiting the embedded pure or generalized network structure.

Another possibility is to view processing networks as pure or generalized networks with side activities, which represent the refining and blending processes (cf. Remark 1.1.1). Pure processing network problems formulated in this way can be solved by the recent Simplex SON approach of GLOVER & KLINGMAN [1981], which exploits again the embedded pure network structure. In doing this, in general, a smaller working basis would be required than in applying CHEN & SAIGAL's algorithm to the side-constraints-formulation. For generalized network problems with side activities (and side constraints) no special algorithms are known.
Here the side-activities-formulation will be used in developing solution procedures for processing network problems. It appears that these procedures are related to the Simplex SON appraoch. Similarities and differences will be discussed in Chapter 6. The only aspect emphasized here is that the typical feature of processing networks, i.e., proportionality of flow in certain subsets of the arc set, is not considered in the above mentioned procedures of CHEN & SAIGAL, HULTZ & KLINGMAN and GLOVER & KLINGMAN.

It is quite surprising that the processing network structure is hardly analyzed quantitatively in the literature. Some work has been done in the economic field. SCHAEFER [1978] studied the maximal flow problem in pure processing networks with only refining processes or only blending processes. His main intention was to solve input/output type problems and the approach he used was an extension of FORD & FULKERSON's [1962] labeling approach for maximal flow problems in pure networks. Before 1978 graph theoretic analysis of economic models were presented by, e.g., PETER [1954] and CZAYKA [1972], but these studies dealt with qualitative rather than quantitative aspects. In the Operations Research oriented literature no special studies on processing networks are known. It should be said, however, that processing networks are closely related to so-called networks with homologous arcs. Such problems were posed by BERGE & GHOUILA-HOURI [1965] and MAYEDA [1968]. Special solution procedures for such problems are not known, only GHOUILA-HOURI [1960] studied a special case. Of theoretical importance is ITAI's [1978] work. He proved that the problem of finding a maximal flow in a pure network with homologous arcs is polynomially equivalent to general LP. Processing networks can be seen as more general structures than pure and generalized networks. On the other hand they can be considered (at least at first sight, cf. Chapter 5) as more special problems than general LP's.
In view of the historical developments this thesis aims to extend the known

results for pure and generalized networks by using primal basis partitioning approaches. An important aspect is that the typical processing network structure is analyzed and exploited.
Three types of processing network problems are taken into consideration:
1. pure,
2. generalized,
3. pure or generalized with additional linear constraints.
We will call the solution procedures, developed for these types of problems, Simplex PRON procedures (from PROcessing Networks).

1.2. Survey

In order to make this thesis self-contained and to make it possible to describe formulations and results in a unified format, some background information is given in Chapter 2. The backbone of all procedures considered is the primal Simplex algorithm for LP-problems with simple upper bounds. It is briefly summarized in Section 2.3. Moreover, an overview is given of well-known results on pure and generalized network problems.
The statements:
"a basis in a pure network is a rooted spanning tree"
and
"a basis in a generalized network is a forest of quasi-trees"
are proved in a quite unusual fashion, namely by using a condition much alike or the same as one which arises in a theorem due to HALL [1935], which deals with sets of distinct representatives. This is done because HALL's theorem plays an important role in Chapters 3 and 4.

Chapter 3 is concerned with pure processing networks. In Section 3.2 two mathematical formulations are given for the minimal cost flow problem. The first one states the problem as a pure network with additional linear constraints. The second one is more compact and can be viewed as a pure network with side activities, where each of the side activities represent either a refining process or a blending process. This compact formulation is used for the solution procedure.
In Section 3.3 the basis structure is analyzed and described in terms of the so-called basis graph, that is the subgraph of the original network which corresponds to a basis matrix. The basis structure is exploited in a specification of the primal Simplex algorithm (Section 3.4). The main characteristics of this approach are:

1. the transportation part of the basis is extracted. In each iteration there is an interaction between this transportation part and the so-called working basis.
2. the size of the working basis varies dynamically and is equal to the number of basic refining and blending processes.
3. a simple labeling procedure determines which basic processes can take part at a nonzero level in the representation of the process which enters the basis.
4. a certain substructure of the basis graph, namely some specific spanning tree, is kept stored and is updated after each basis change by means of the previously given labels.
5. the labeling procedure provides a block triangular form of the working basis (with two blocks on the main diagonal).

A somewhat different view on solving pure processing network problems is presented in Section 3.5. Perhaps this approach is intuitively less appealing then the one in Section 3.4, but it has certain advantages.
Some remarks, for instance on implementation considerations, are made in Section 3.6. Here also the relation between HALL's theorem, the exploited structure of the basis graph and the possibility to block triagularize the working basis further by applying an algorithm of TARJAN [1972] is pointed out. See also DUFF & REID [1978a].

Chapter 4 considers generalized processing networks. It appears possible to generalize the results of Chapter 3 to generalized processing networks, except for some details.

Where in the previous two chapters processing networks were considered as more general structures than pure and generalized networks, Chapter 5 looks in the other direction: What about the relation between processing networks and general LP's?
It appears that any LP-problem can readily be interpreted as a generalized processing network problem in which both positive and negative multipliers may be present. So the procedure of Chapter 4 can in principle be applied to general LP's, leading to an approach in which the (working) basis is block triangularized. It stands to reason that this approach is the most efficient for generalized network problems with relatively few side activities. The relation between this approach and other sparse matrix approaches known in the literature will be discussed.

Furthermore, it is possible to give an "almost" pure processing network interpretation to general LP's. The approaches of Chapter 3 can easily be adapted to solve general LP-problems, although some of the properties which hold for pure processing networks are no longer valid.
It is important to observe that any LP can be transformed to a pure processing network, possibly at the expense of blowing up the size of the problem in a polynomial way. The relevance of this result is not as much that a transformation yields a problem which can be solved easier but rather

1. it shows that a (pure) processing network structure is not as special as it seems at first sight.
2. it gives a certain reassurance that the problem structure is indeed exploited adequately in the procedures presented in Chapters 3 and 4.
3. it gives the opportunity to visualize the structure of certain LP's by drawing processing network diagrams.

Finally it is shown that there are classes of problems which can right away be interpreted as pure processing networks or generalized processing networks with positive multipliers, for instance, the multicommodity network problem.

Chapter 6 deals with pure or generalized processing networks with additional linear constraints. In applying the approaches of Chapters 3 and 4 to such problems the embedded single commodity network structure would not be exploited fully. That is why here a different basis partitioning approach is proposed to solve these problems. In a broader context this approach can be used to solve general LP/embedded network problems and, as a matter of fact, the pure case is an alternative for the Simplex SON approach of GLOVER & KLINGMAN [1981]. It appears that both procedures use similar ideas at some points, but at other points they are different.

Finally, in Chapter 7, the applicability and expected computational results are discussed.

1.3. Conclusions

The first result of this study is deeper insight in the processing network structure itself, in the basis structure, and in the relation to LP. Insight also in the way this structure can be exploited in primal basis partitioning solution procedures.

The solution procedures developed have several desirable properties: they use the embedded pure or generalized network structure, they employ special labeling and updating procedures to accelerate computations and they maintain a block triangular version of the working basis.

Furthermore, the theory developed in this study provides a bridge between pure and generalized networks at one hand and (sparse matrix) LP at the other.

Processing networks have a wide range of applicability. They may become efficient real-world modelling tools. The fact that their structure can be completely pictured in network diagrams may tend to increase the nonanalyst's (management's) level of acceptance.

This monograph provides a complete theory on processing networks. However, it is stressed that much work has to be done on implementation and subsequent computational testing of our methods before conclusions can be drawn on their efficiency.

2. PRELIMINARIES

2.1. Introduction

In order to make this monograph self-contained and to make it possible to describe formulations in a unified format some background information is given in this chapter.
The backbone of all solution procedures considered in the subsequent chapters is the primal Simplex algorithm for LP-problems with simple upper bounds. It is briefly described in Section 2.3.
Furthermore, many of the results known in pure and generalized networks will be used as basic tools in dealing with processing networks. Pure networks are considered in Section 2.4, generalized networks in Section 2.5.

2.2. Notation and definitions

In this section some remarks are made concerning the notation used. Furthermore, the most important concepts which arise in network flow programming are defined.

Matrices and sets will be denoted by uppercase Roman characters (A, B, etc.), vectors and scalars by lowercase Roman or Greek characters (a, b, α, β, etc.). The transpose of a matrix A is given by A'. All vectors considered are assumed to be column vectors.
Finally we denote by:

e_i , the i-th unit vector,
e , a vector with all elements equal to 1,
$|S|$, the number of elements in some set S,
$r(A)$, the rank of a matrix A.

A *directed graph* G(N,A) consists of a set $N = \{1,2,\ldots,m\}$, of which the elements are called *nodes* and a set $A \subseteq N \times N$ of ordered pairs (i,j), $i,j \in N$, called *arcs*.

Arc $(i,j) \in A$ is *directed from* node i *to* node j.
An arc (i,i), $i \in N$, is called a *self-loop*.
Arc $(i,j) \in A$ is said to be *incident* to nodes i and j.
In reverse: both nodes i and j are said to be *incident* to arc $(i,j) \in A$.

REMARK 2.2.1. Note that the above definition of a directed graph does not allow the existence of more than one arc from node i to node j, where $i,j \in N$ (so-called *multiple arcs*). However, the exclusion of multiple arcs is:

- not restrictive, since the occurrence of multiple arcs can always be circumvented by introducing dummy nodes and arcs,
- not essential: the ideas developed in the sequel remain valid when a broader definition of a directed graph is used in which multiple arcs are allowed.

The reason for adopting the present definition of a directed graph is, that it gives rise to a convenient notation (i,j) to denote an arc from node i to node j. □

Nodes i and j are said to be *adjacent* iff $(i,j) \in A$ (so a node i is adjacent to itself iff the self-loop $(i,i) \in A$).
A *network* is a directed graph with one or more real valued functions defined on the arc set.
The *after set* A(i) and the *before set* B(i) of a node $i \in N$ are defined as:

2.2.1. $\quad A(i) := \{j \in N \mid (i,j) \in A\}$,

2.2.2. $\quad B(i) := \{j \in N \mid (j,i) \in A\}$.

Suppose that $\{i_k \mid i_k \in N,\ k = 1,2,\ldots,\ell\}$, with $\ell \geq 2$, is a set of distinct nodes and w_k is either arc $(i_k,i_{k+1}) \in A$ or arc $(i_{k+1},i_k) \in A$, $k = 1,2,\ldots,\ell-1$, then the sequence

2.2.3 $\quad i_1,w_1,i_2,w_2,\ldots,i_{\ell-1},w_{\ell-1},i_\ell$

is called a *path* from i_1 to i_ℓ. The arcs (i_k,i_{k+1}) are called *forward* arcs, arcs (i_{k+1},i_k) *backward* arcs.
If $i_1,\ldots,i_{\ell-1}$ are distinct nodes and $i_1 = i_\ell$, then sequence 2.2.3 is called a *cycle* ($\ell \geq 2$). Note that a self-loop is a cycle.
If in G(N,A) a path exists from every node to every other node, G(N,A) is said to be *connected*.

A *tree* is a connected directed graph which contains no cycles.
Some arbitrary node i_0 of the node set of a tree is designated as the *root* of the tree. If the *root arc* (i_0,i_0), which is a self-loop, is attached to a tree, we speak of a *rooted tree*.
The unique path from node i to node j in a tree will be denoted by P_{ij}.
A *spanning tree* in G(N,A) is a tree with node set N and arc set $\subseteq$ A.
A *quasi-tree* is a connected graph with exactly one cycle.
A *forest* of trees (respectively quasi-trees) is a set of disjunct trees (respectively quasi-trees).
A *spanning forest* of (quasi-) trees in G(N,A) is a forest of (quasi-) trees with arc set $\subseteq$ A, such that each node of N belongs to this forest.

2.3. The Simplex algorithm for LP-problems with upper bounds

The Revised Simplex algorithm for LP-problems with (simple) upper bounds provides the backbone for all network flow algorithms considered in the sequel. Only a brief description is presented here. A more elaborate treatment can be found in, e.g., DANTZIG [1963], LASDON [1970], and BAZARAA & JARVIS [1977].
By introducing artificial variables, any LP-problem can be cast into the so-called canonical form:

2.3.1. $\quad$ minimize $c'x$

2.3.2. $\quad Ax = b$

2.3.3. $\quad 0 \leq x \leq u$,

where $c,u,x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and A is an $m \times n$ matrix. In the literature sometimes the constraint

2.3.4. $\quad \ell \leq x \leq u$,

with $\ell \geq 0$ is used instead of 2.3.3. In that case by using the transformation $\bar{x} := x - \ell$ the form 2.3.1 - 2.3.3 is obtained. In the rest of this monograph we always consider lower bounds equal to zero.
The dual problem of 2.3.1 - 2.3.3 is

2.3.5. maximize $b'\pi - u'\nu$

2.3.6. $A'\pi - \nu \leq c$

2.3.7. $\nu \geq 0$

where $\pi \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^n$.

ASSUMPTION 2.3.1. *The rank of matrix* A *equals* m.

This assumption is standard and not restrictive since in practice artificial variables are added such that the extended coefficient matrix has full row rank.
Let B be a square nonsingular submatrix of A of order m; then B is called a *basis*.
Suppose that matrix A, after permuting the columns, is written as:

2.3.8. $A = [B, N_1, N_2]$.

Let $x' = [x_B', x_{N_1}', x_{N_2}']$ be the partitioning of x' compatible with 2.3.8 (u and c are partitioned similarly), which satisfies:

2.3.9. $x_{N_1} = 0$

2.3.10. $x_{N_2} = u_{N_2}$

2.3.11. $x_B = B^{-1}b - B^{-1} N_2 u_{N_2}$,

then x is said to be a *basic solution*; x_B denotes the basic variables, x_{N_1} the nonbasic variables at their lower bound, x_{N_2} the nonbasic variables at their upper bound.
If, in addition, x satisfies 2.3.3, x is called a *basic feasible solution*.
The value of the objective function will be denoted by z.
The Simplex algorithm is discussed next.

Simplex algorithm for LP-problems with upper bounds

Initialization

As starting basis the identity matrix, corresponding to artificial variables, can be chosen. By applying the "Big-M" method or Phase I of the "Phase I, Phase II" method, a basic feasible solution is determined if it exists. If no (basic) feasible solution exists the algorithm stops.

1. Determine the Simplex multipliers

The Simplex multipliers, also called dual variables or shadow prices, are obtained from:

2.3.12 $\pi' = c_B' B^{-1}$.

2. Calculate the reduced costs

This operation is sometimes called pricing. The reduced cost vector $\bar{c}$ can be found from:

2.3.13 $\bar{c}' = \pi' A - c'$,

where, according to 2.3.12: $\bar{c}_B' = \pi' B - c_B' = 0$.

3. Perform the optimality test

If $\bar{c}_j \leq 0$ for all nonbasic variables x_j at their lower bound,
and $\bar{c}_j \geq 0$ for all nonbasic variables x_j at their upper bound,
then the current solution is optimal and the algorithm stops.

4. Choose the nonbasic variable to enter the basis

Let I denote the index set of all nonbasic variables which violate the optimality test in step 3. As variable to enter the basis can be chosen any x_j, $j \in I$.
Suppose x_k is chosen. In Simplex tableau terms $a_{\cdot k}$ is the pivot column.

5. Find the representation of the entering column in terms of the basis

The *representation vector* y_k of $a_{\cdot k}$ in terms of the basis is calculated from

2.3.14. $y_k = B^{-1} a_{\cdot k}$.

6. Perform the minimal ratio test

Consider the two possible cases:

(a) x_k is at its lower bound. Define Δ_k as:

$$2.3.15.\quad \Delta_k := \min \begin{cases} \min_i \left\{ \dfrac{x_{B_i}}{y_{ik}} \;,\; y_{ik} > 0 \right\} \\ \min_i \left\{ \dfrac{u_{B_i} - x_{B_i}}{-y_{ik}} \;,\; y_{ik} < 0 \right\} \\ u_k \\ \infty \end{cases}$$

(b) x_k is at its upper bound. Define Δ_k as:

$$2.3.16.\quad \Delta_k := \min \begin{cases} \min_i \left\{ \dfrac{x_{B_i}}{-y_{ik}} \;,\; y_{ik} < 0 \right\} \\ \min_i \left\{ \dfrac{u_{B_i} - x_{B_i}}{y_{ik}} \;,\; y_{ik} > 0 \right\} \\ u_k \\ \infty \end{cases}$$

If $\Delta_k = \infty$, the solution is unbounded and the algorithm stops.
Otherwise, choose a row index s for which the minimum is obtained. Row s is said to be the pivot row.

7. Update the activity levels and the basis inverse

In updating the objective function value and the activity levels, again the two cases of step 6 are considered.

(a) x_k is at its lower bound zero.

$$2.3.17.\quad x_k := \Delta_k$$

$$2.3.18.\quad x_B := x_B - y_k \Delta_k \;,$$

other activity levels remain what they are.

$$2.3.19.\quad z := z - \bar{c}_k \Delta_k \;.$$

(b) x_k is at its upper bound u_k.

2.3.20. $x_k := u_k - \Delta_k$

2.3.21. $x_B := x_B + y_k \Delta_k$,

other activity levels remain what they are.

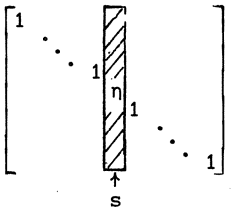
2.3.22. $z := z + \bar{c}_k \Delta_k$.

If $\Delta_k = u_k$, variable x_k shifts from its lower bound to its upper bound (or the other way round) and the basis remains the same. In this case proceed with step 3.
Otherwise the basis inverse is updated by:

2.3.23. $\hat{B}^{-1} = E.B^{-1}$,

where E is an elementary matrix given by:

2.3.24. $E =$



with η a vector with elements:

2.3.25. $\eta_s = \frac{1}{y_{sk}}$

2.3.26. $\eta_j = -\frac{y_{jk}}{y_{sk}}$, $\quad j \neq s$.

Matrix E describes the pivot operation. Continue with step 1.

As can be seen from this description, the basis inverse B^{-1} plays an essential role in steps 1 and 5. In actual implementations B^{-1} is usually stored either in product form or in elimination form (see e.g., BASTIAN [1980]) and reinverted after a number of iterations in order to reduce cumulative round-off errors and storage requirements.
Furthermore, it is quite usual to replace the nonbasic variables, which are at their upper bound, by their complement $\bar{x}_{N_2} := u_{N_2} - x_{N_2} = 0$. This transformation makes the computation somewhat easier, since one only has to deal with nonbasic variables which are at their lower bound zero.

2.4. *Pure network flow problems*

The theory of pure networks plays an important role in Chapter 3, which deals with pure processing networks. Several relevant aspects of pure networks are discussed here.

Let G(N,A) denote a directed and connected graph, with N the set of nodes and A the set of arcs. The number of nodes is m, the number of arcs n. If self-loops (i,i), $i \in N$, are present in G(N,A) they can be replaced by common arcs (i,m+1), where (m + 1) is an additional node (cf. BAZARAA & JARVIS [1977, pp. 419, 420]).

ASSUMPTION 2.4.1. G(N,A) *does not contain any self-loop.*

The LP-formulation of the minimal cost flow problem in a pure network is:

2.4.1. $$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

2.4.2. $$-\sum_{j\in A(i)} x_{ij} + \sum_{j\in B(i)} x_{ji} = b_i , \qquad i \in N$$

2.4.3. $$0 \le x_{ij} \le u_{ij} , \qquad (i,j) \in A$$

Equations 2.4.2 are the conservation of flow equations, where b_i ($i \in N$) denotes:
- the external demand ($b_i > 0$),
- the external supply ($b_i < 0$), or
- no external demand or supply ($b_i = 0$).

Capacity bounds are given by 2.4.3, where u_{ij} is not necessarily finite. The coefficient matrix of the left-hand sides of 2.4.2 is denoted by $A = [a_{\ell,ij}]$.
The dual problem of 2.4.1-2.4.3 is given by

2.4.4. $$\text{maximize} \sum_{i\in N} b_i \pi_i - \sum_{(i,j)\in A} u_{ij} \nu_{ij}$$

2.4.5. $$-\pi_i + \pi_j - \nu_{ij} \le c_{ij} , \qquad (i,j) \in A$$

2.4.6. $$\nu_{ij} \ge 0 , \qquad (i,j) \in A .$$

Properties of matrix A

Row $a_{\ell\cdot}$ of A is associated with node $\ell \in N$, column $a_{\cdot ij}$ of A is associated with arc $(i,j) \in A$ and has exactly two nonzero elements, namely
-1 in row i, and
$+1$ in row j.
The column sum of each column in A is zero: $e'A = 0$.

REMARK 2.4.2. As noted before any LP-problem with a coefficient matrix A, in which
- each column has at most two elements $\neq 0$,
- each column with two nonzero elements has column sum zero,
can be regarded as a pure network problem.

By using positive column scales it can be accomplished that all nonzero elements of such a matrix A are equal to ± 1.
A column of A with only one nonzero element in some row i, which is equal to -1, can be thought to represent a self-loop or an arc from node i to outside the network (see e.g., BAZARAA & JARVIS [1977]).
A column of A with only one nonzero element in some row i, which is equal to $+1$, can be thought to represent an arc from outside the network to node i. □

THEOREM 2.4.3. *The rank of* A *equals* $m - 1$.

PROOF. Because $e'A = 0$, the rank of A must be smaller than or equal to $m - 1$. Since G(N,A) is connected, a submatrix of A can be constructed which corresponds to a spanning tree in G(N,A). It can easily be shown that this matrix has rank $m - 1$, see e.g., BAZARAA & JARVIS [1977]. □

We introduce a single artificial variable $x_{i_0 i_0}$ with $a_{\cdot i_0 i_0} = -e_{i_0}$ (i_0 arbitrarily chosen from $\{1,\ldots,m\}$). It is easy to prove that matrix

2.4.7. $$A^* = [-e_{i_0}, A]$$

has rank m.

Properties of a basis

Let B denote a basis of A^*. Column $a_{\cdot i_0 i_0}$ always belongs to B and can be thought to represent the self-loop (i_0,i_0). Assume column $a_{\cdot i_0 i_0}$ to be the first column of B.

Let $\underline{B}$ denote the $m \times (m-1)$ matrix, consisting of the last $(m-1)$ columns of B. So:

2.4.8. $\quad B = [a_{\cdot i_0 i_0}, \underline{B}]$.

We define the *basis graph* associated with matrix B as the subgraph of G(N,A) with node set N and arcs in A which correspond to the columns in $\underline{B}$, and the self-loop (i_0,i_0).
Next a lemma is stated, which is generalized in a certain sense in Chapter 3. To avoid notational difficulties we denote the elements of $\underline{B}$ by $\underline{b}_{\ell p}$ (instead of $\underline{b}_{\ell,ij}$).
Let S be a nonempty subset of $\{1,2,\ldots,m-1\}$, associated with the columns of $\underline{B}$. Furthermore, let R(S) be defined by:

2.4.9. $\quad R(S) := \{\ell \mid \ell \in \{1,2,\ldots,m\}, \exists_{p \in S}: \underline{b}_{\ell p} \neq 0\}$.

So R(S) is related to those rows of $\underline{B}$ which have at least one nonzero element in the columns associated with S.

LEMMA 2.4.4. *Given a collection of* $|S|$ *columns of matrix* $\underline{B}$ *there are at least* $|S|+1$ *rows in* $\underline{B}$ *which have a nonzero element in these columns:*

2.4.10. $\quad |R(S)| \geq |S| + 1$.

PROOF. Suppose that $|R(S)| \leq |S|$. Then, because $e'\underline{B} = 0$, the columns of $\underline{B}$ associated with S would clearly be linearly dependent. This contradicts the fact that B denotes a basis. □

REMARK 2.4.5. Note that the only argument used in proving Lemma 2.4.4 is that $\underline{B}$ is an $m \times (m-1)$ matrix of rank $(m-1)$ with the property that $e'\underline{B} = 0$. □

We can use Lemma 2.4.4 to prove the well-known theorem:

THEOREM 2.4.6. *A basis graph in a pure network is a rooted spanning tree.*

PROOF. Suppose the basis graph contains a cycle besides the self-loop (i_0,i_0). Let S denote the columns in $\underline{B}$ associated with the arcs in this cycle. Then R(S) corresponds to the set of nodes incident to the arcs in the cycle. In a cycle the number of nodes equals the number of arcs, so $|R(S)| = |S|$. This is in contradiction with Lemma 2.4.4. Since the basis graph contains $(m-1)$ "real" arcs these arcs form the arc set of a spanning

tree in G(N,A). The self-loop (i_0,i_0) is usually called the root-arc, and node $i_0 \in N$ the root of this spanning tree.
This completes the proof. □

The reverse of Theorem 2.4.6 is true too:

THEOREM 2.4.7. *Every rooted spanning tree with arc set* $\subseteq A$ *is a basis graph.*

PROOF. See BAZARAA & JARVIS [1977]. □

A square matrix is said to be (upper) *triangular* if the rows and columns can be permuted such that all elements below the main diagonal are zero.

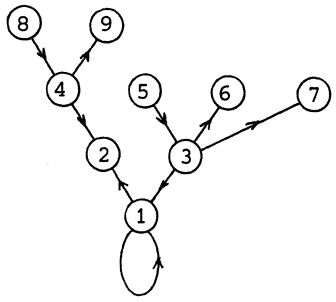
THEOREM 2.4.8. B *is (upper) triangular*

PROOF. A constructive proof is given. The permuted B matrix will be denoted by B^*.

1. Take $a_{\cdot i_0 i_0}$ as the first column of B^* and row i_0 as the first row of B^*.
   Put $W = \{i_0\}$.
2. If $W = N$ then stop, B^* is found.

Otherwise, let (i,j) be an arc in the basic spanning tree, such that either $i \in W$ or $j \in W$. Such an (i,j) always exists, since a spanning tree is a connected graph which contains no cycles.
Take $a_{\cdot ij}$ as the next column in B^*.
If $i \notin W$ make row i the next row of B^*, set $W = W \cup \{i\}$ and goto 2.
If $j \notin W$ take row j as the next row of B^*, set $W = W \cup \{j\}$ and goto 2.
It is obvious that this constructive scheme provides a matrix B^* with all elements below the main diagonal equal to zero. □

EXAMPLE 2.4.9. For the rooted spanning tree in Figure 2.4.1, B^* is a possible realization.



$B^* =$

11	12	42	31	53	49	36	84	37	
-1	-1		1						1
	1	1							2
		-1			-1		1		4
			-1	1		-1		-1	3
				-1					5
					1				9
						1			6
							-1		8
								1	7

Figure 2.4.1. A rooted spanning tree and an associated triangularized basis.

The properties of a basis and associated basis graph, mentioned in Theorems 2.4.6 and 2.4.8, make it possible to perform the steps of the Simplex algorithm by using the basis graph (a rooted spanning tree) instead of the basis inverse B^{-1}. The advantages of such an approach are already mentioned in Subsection 1.1.1.
Before we give an outline of the Simplex algorithm for pure network problems a clarification of some of the calculations, which have to be carried out, is presented.

Solving $\pi' B = c_B'$

In order to determine the Simplex multipliers π the system

2.4.11 $\quad \pi' B = c_B' \ ,$

must be solved (cf. 2.3.12). In network terms this can be done in the following way (cf. the constructive proof of Theorem 2.4.8):

1. Take $\pi_{i_0} = 0$ (it can be assumed that $c_{i_0 i_0} = 0$). Set $W = \{i_0\}$.
2. If $W = N$, stop.

Otherwise, take an arc (i,j) such that either $i \in W$ or $j \in W$.
If $i \in W$ then π_i has already been determined and π_j can be found from

2.4.12. $\quad -\pi_i + \pi_j = c_{ij} \ .$

Make $W = W \cup \{j\}$ and goto 2.
If $j \in W$, π_j is known and π_i can be evaluated from 2.4.12. Set $W = W \cup \{i\}$ and goto 2.

Informally speaking the Simplex multipliers are determined in some sequence "from the root towards the *leaves* (i.e., those nodes of N which are incident to only one arc in the basic spanning tree)".
It is noted that, in the subsequently discussed Simplex algorithm, the Simplex multipliers are evaluated in this way only in the initialization step. In all other steps they can be found by updating the previous vector π.

Solving $Bx = b^*$

In order to find the activity levels of the basic variables, the system:

2.4.13. $Bx_B = b^*$,

where $b^* = b - N_2 u_{N_2}$ (formula 2.3.11), must be solved.
In a similar way as the Simplex multipliers are evaluated, these activity levels (flow levels in the basic arcs) are calculated "from the leaves towards the root".
They are determined in this way only in the initialization step. In all other steps they can be found, as usual, by updating the previous vector x.

Equations of the type $Bx = b^*$ must also be solved in determining the representation $y_{k\ell}$ of the entering column, say $a_{\cdot k\ell}$, in terms of the basis:

2.4.14. $By_{k\ell} = a_{\cdot k\ell}$.

This can be done in an easier way than indicated above, simply because the right-hand side of 2.4.14 has a special form.
Associated with column $a_{\cdot k\ell}$ is arc (k,ℓ).
Let $C_{k\ell}$ denote the set of arcs in the basic (rooted) spanning tree, which belong to the unique cycle induced in this spanning tree by the entering arc (k,ℓ). $C_{k\ell}$ is given an orientation consistent with (k,ℓ). Denote by $C^f_{k\ell}$ the set of forward arcs in $C_{k\ell}$, by $C^b_{k\ell}$ the set of backward arcs. It is easy to observe that $a_{\cdot k\ell}$ can be written as:

2.4.15. $$a_{\cdot k\ell} = - \sum_{(i,j)\in C^f_{k\ell}} a_{\cdot ij} + \sum_{(i,j)\in C^b_{k\ell}} a_{\cdot ij} ,$$

or in words: in the representation of $a_{\cdot k\ell}$ in terms of the basic columns, the columns associated with forward arcs in $C_{k\ell}$ have coefficient -1, the columns associated with backward arcs in $C_{k\ell}$ have coefficient + 1, and all other basic columns have coefficient 0.

For obvious reasons, vector $y_{k\ell}$ will be called the *cycle vector*, induced by arc (k,ℓ) in the basic spanning tree.

In view of the theory of generalized networks (Section 2.5), it is instructive to consider the representation of $a_{\cdot k\ell}$ in terms of B in a slightly different fashion.
Denote by P^f_{ij} the set of forward arcs on the unique path from node i to node j in the basic spanning tree and by P^b_{ij} the set of backward arcs. Then:

2.4.16. $$a_{\cdot k\ell} = -e_k + e_\ell$$

2.4.17. $$e_k = -\sum_{P^f_{ki_0}} a_{\cdot ij} + \sum_{P^b_{ki_0}} a_{\cdot ij} + e_{i_0}$$

2.4.18. $$e_\ell = -\sum_{P^f_{\ell i_0}} a_{\cdot ij} + \sum_{P^b_{\ell i_0}} a_{\cdot ij} + e_{i_0} \ .$$

Using these formulae, one observes that the root arc plus all arcs which belong to $P_{ki_0} \cap P_{\ell i_0}$ have a zero coefficient in the representation and in fact 2.4.15 results (see Figure 2.4.2).
Vector $B^{-1} e_j$ is called the *root-path* vector of node j since it describes the path from node j to the root of the basic spanning tree.

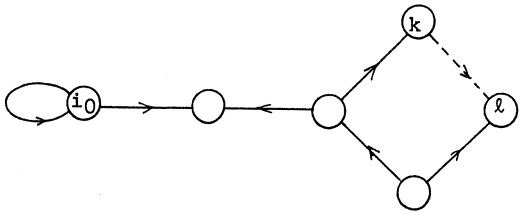


Figure 2.4.2. Illustration of the representation of $a_{\cdot k\ell}$ in terms of B.

Observe from 2.4.15 that one of the arcs in $C_{k\ell}$ must leave the basis graph, consequently a new basic rooted spanning tree arises.

In the following specification of the Simplex algorithm it is assumed that the basic rooted spanning tree is stored and updated in some convenient way.

Simplex algorithm for the minimal cost flow problem in a pure network

Initialization

A simple way to find a starting basis is: introduce an additional node $(m+1)$ and arcs $(i,m+1)$ if $b_i \leq 0$, $i \in N$, and $(m+1,i)$ if $b_i > 0$, $i \in N$. These added arcs form the arc set of a spanning tree in the extended network. Take an arbitrary root i_0 with root arc (i_0,i_0). Let B denote the matrix representing the rooted spanning tree. B is taken as a starting basis. Take all nonbasic variables at their lower bound zero. Determine the flow levels x_B and the Simplex multipliers π as indicated above. Use the Big-M method or Phase I of a two phase method to find a basic feasible solution (if it exists). Alternative ways to determine a starting basis can be found in BAZARAA & JARVIS [1977] and in GLOVER, KARNEY & KLINGMAN [1974].

1. Determine the Simplex multipliers

The Simplex multipliers can be evaluated as described above. However, after each basis change it is possible to update the previous vector π. This is discussed at the end of step 7.

2. Calculate the reduced costs

The reduced costs are determined from:

2.4.19. $$\bar{c}_{ij} = -\pi_i + \pi_j - c_{ij} \qquad (i,j) \in A \ .$$

3. Perform the optimality test

This is standard (see Section 2.3).

4. Choose the nonbasic variable to enter the basis

See Section 2.3. Suppose $a_{\cdot k\ell}$ is selected to enter the basis (arc (k,ℓ) enters the basis graph).

5. Find the representation of $a_{\cdot k\ell}$ in terms of B

Determine the cycle vector $y_{k\ell}$ from

$$By_{k\ell} = a_{\cdot k\ell} \ ,$$

as explained above.

6. Perform the minimal ratio test

See Section 2.3. Suppose $a_{\cdot st}$ leaves the basis.

7. Update

Updating the objective function value and flow levels is standard.
By dropping arc (s,t) in the previous basic spanning tree, two subtrees, say T_1 and T_2, remain with $s \in T_1$ and $t \in T_2$. The Simplex multipliers can easily be updated:

2.4.20. $\pi_i := \pi_i \quad , \quad i \in T_1 \ ,$

2.4.21. $\pi_i := \pi_i - \bar{c}_{st} \ , \quad i \in T_2 \ .$

Adding subsequently arc (k,ℓ) results in the basic rooted spanning tree for the new situation.
Continue with step 2.

2.5. *Generalized network flow problems*

Generalized networks play an important role in solving generalized processing network problems (Chapter 4). Some relevant aspects of generalized networks are discussed here.

Suppose G(N,A) is a directed and connected graph, with node set N and arc set A. The number of nodes is m, the number of arcs n. Self-loops are allowed to be present.
The essential difference with pure network flow problems (Section 2.4) is that flow is not necessarily conserved in transporting it along arcs. In every arc $(i,j) \in A$ it is assumed that, whenever the flow in (i,j) is x_{ij}, upon arrival in node j the flow has value $g_{ij} x_{ij}$. The factor g_{ij}, $(i,j) \in A$ is called the *multiplier* or *gain* of arc (i,j).

The multipliers are assumed to be arbitrary real numbers.
However, negative multipliers are intuitively not as appealing as positive ones. Nevertheless the following interpretation can be given:
If $g_{ij} < 0$ and the flow in arc (i,j) is x_{ij}, necessarily a flow of magnitude $-g_{ij} x_{ij}$ must arrive at node j. (See also Section 5.5.)

The LP formulation of the minimal cost flow problem in a generalized network is stated as:

2.5.1. $$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

2.5.2. $$-\sum_{j\in A(i)} x_{ij} + \sum_{j\in B(i)} g_{ji} x_{ji} = b_i \,, \qquad i \in N$$

2.5.3. $$0 \le x_{ij} \le u_{ij} \qquad (i,j) \in A \,.$$

Equations 2.5.2 are the conservation of flow equations in the nodes of the network, where b_i, if unequal to zero, denotes the external demand ($b_i > 0$) or supply ($b_i < 0$) in node i.
The coefficient matrix of the left-hand sides of 2.5.2 is denoted by $A = [a_{\ell,ij}]$.
The dual problem of 2.5.1-2.5.3 is given by:

2.5.4. $$\text{maximize} \sum_{i\in N} b_i \pi_i - \sum_{(i,j)\in A} u_{ij} \nu_{ij}$$

2.5.5. $$-\pi_i + g_{ij}\pi_j - \nu_{ij} \le c_{ij} \,, \qquad (i,j) \in A$$

2.5.6. $$\nu_{ij} \ge 0 \,, \qquad (i,j) \in A \,.$$

Before discussing some properties of matrix A, an important concept, the cycle factor of a cycle, is introduced. This cycle factor plays a role both in theoretical and computational considerations.
Let C denote a cycle in G(N,A) with arbitrary orientation. C_f is the set of forward arcs in C, C_b the set of backward arcs. The *cycle factor* $\alpha(C)$ is defined as:

2.5.7. $$\alpha(C) := \prod_{(i,j)\in C_f} g_{ij} \Big/ \prod_{(i,j)\in C_b} g_{ij} \,.$$

Properties of matrix A

Row $a_{\ell\cdot}$ of A is associated with node $\ell \in N$, column $a_{\cdot ij}$ of A is associated with arc $(i,j) \in A$ and has either two nonzero elements, namely
- 1 in row i, and
g_{ij} in row j,
or only one nonzero element, namely
$-g_{ii}$ in row i, if i = j (so (i,j) is a self-loop).

Note that an arc (i,j) with multiplier $g_{ij} = 0$ has the same representation in matrix A as self-loop (i,i) with multiplier $g_{ii} = 1$. Therefore the following assumption is not restrictive.

ASSUMPTION 2.5.1. *In* G(N,A) *no arcs are present with a multiplier equal to zero.*

REMARK 2.5.2. Any LP-problem with a coefficient matrix A in which each column has at most two elements $\neq 0$, can be regarded as a generalized network problem. If we replace 2.5.2 by

$$2.5.8. \qquad -\sum_{j\in A(i)} h_{ij}x_{ij} + \sum_{j\in B(i)} g_{ji}x_{ji} = b_i , \qquad i \in N ,$$

2.5.1, 2.5.8 and 2.5.3 formulate such an LP-problem.
By using positive column scales it can always be accomplished that h_{ij} in 2.5.8 equals ± 1. □

THEOREM 2.5.3. *The rank of matrix* A *equals* $(m-1)$ *or* m.

The proof of this theorem is similar to that of Theorem 2.4.3. See also Figure 2.5.1.

Under strong conditions a generalized network problem can be reduced to a pure network problem by means of scaling. In this respect the following theorem is valid:

THEOREM 2.5.4. *Let* G(N,A) *denote a connected generalized network. Problem* 2.5.1-2.5.3 *can be scaled to a pure network problem iff one of the following equivalent conditions is valid:*

(a) $r(A) = m - 1$
(b) $\alpha(C) = 1$, *for every cycle* C *in* G(N,A) *which is not a self-loop.*

PROOF. See GLOVER & KLINGMAN [1973] and TRUEMPER [1976]. □

Both GLOVER & KLINGMAN and TRUEMPER developed simple scaling procedures. Scaling generalized networks to networks with positive multipliers is discussed in TRUEMPER [1976]. Scaling generalized networks to networks in which all multipliers g_{ij} satisfy $0 < g_{ij} \leq 1$ (so-called lossy networks) or $g_{ij} \geq 1$ (gainy networks) is discussed in KOENE [1979b].

Theorem 2.5.4 implies that, if $r(A) = m - 1$, the generalized network problem can be solved as a pure network problem, after a suitable scaling has been performed.
In the remaining part of this chapter the following assumption holds:

ASSUMPTION 2.5.5. *The rank of* A *equals* m.

Properties of a basis

Let B denote a basis of A.
Define the *basis graph* associated with matrix B as the subgraph of G(N,A) with node set N and arc set the arcs associated with the columns in B.
A similar lemma as Lemma 2.4.4, which deals with pure networks, is stated.
Suppose $B = [b_{\ell p}]$.
Let S be a nonempty subset of $\{1,\ldots,m\}$, associated with the columns in B.
Similarly as in 2.4.9, R(S) is defined as:

2.5.9. $\quad R(S) = \{\ell \mid \ell \in \{1,\ldots,m\}, \exists_{p\in S}: b_{\ell p} \neq 0\}$.

LEMMA 2.5.6. *Given a collection of* $|S|$ *columns of* B *there are at least as many rows in* B *which contain a nonzero element in these columns:*

2.5.10. $\quad |R(S)| \geq |S|$.

PROOF. If $|R(S)| \leq |S| - 1$ the columns of B associated with S are linearly dependent. This contradicts the fact that B denotes a basis. □

REMARK 2.5.7. Note that the only argument used in proving this lemma is the fact that B is a square nonsingular matrix. □

It is remarked that the relation 2.5.10 also arises in a theorem due to HALL [1935] in dealing with systems of distinct representatives, see also FORD & FULKERSON [1962, p. 67]: Let $V = \{V_1,\ldots,V_m\}$ be a family of subsets of a given set $W = \{w_1,\ldots,w_q\}$. A list of distinct elements of W, say $W^* = \{w_{\ell_1},\ldots,w_{\ell_m}\}$ is a system of distinct representatives for V if $w_{\ell_j} \in V_j$; w_{ℓ_j} is said to represent V_j.

THEOREM 2.5.8 (HALL). *A system of distinct representatives for* $V = \{V_1,\ldots,V_m\}$ *exists iff every union of* $|S|$ *sets of* V *contains at least* $|S|$ *distinct elements,* $|S| = 1,\ldots,m$.

The condition in this theorem is the same as that in Lemma 2.5.6, only a different terminology is used.

An immediate consequence of Lemma 2.5.6 and Theorem 2.5.8 is (cf. remark 2.5.7):

COROLLARY 2.5.9. *The rows (or the columns) of a square nonsingular matrix can be permuted such that the main diagonal of this permuted matrix is zero-free.*

Hall's theorem will appear to play an important role in Chapters 3 and 4.

THEOREM 2.5.10. *A basis graph in a generalized network is a (spanning) forest of quasi-trees.*

PROOF. Consider a connected component of the basis graph. Suppose this component contains q arcs, then Lemma 2.5.6 shows that this component contains at most q nodes.
Since the number of arcs in the basis graph equals the number of nodes this implies that each connected component must be a quasi-tree. □

The reverse of Theorem 2.5.10 is in general not true (compare with the situation in pure networks: Theorem 2.4.7):

THEOREM 2.5.11. *A forest of quasi-trees with node set* N *and arc set* $\subseteq$ A *is a basis graph iff* $\alpha(C) \neq 1$ *for every cycle* C, *which is not a self-loop.*

PROOF. See TRUEMPER [1976] and also Figure 2.5.1. □

Whether a subgraph of G(N,A) is a basis graph or not does not only depend on the topology of this subgraph but also on the values of the multipliers.

A square matrix is said to be *one-triangular* if the rows and columns can be permuted such that all elements below the first lower diagonal are zero.

THEOREM 2.5.12. B *is one-triangular.*

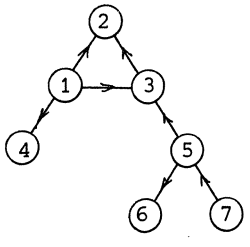
Before proving this theorem it is remarked that B has a block diagonal form:

2.5.11. $$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_q \end{bmatrix}$$

Each block B_i corresponds to a quasi-tree Q_i in the basis graph (each quasi-tree has as many nodes as arcs). Therefore, it is sufficient to show that each block B_i has a one-triangular structure. A constructive proof is given.

PROOF of Theorem 2.5.12. Consider quasi-tree Q_i of the basis graph and its associated block B_i of the basis.
If the cycle in Q_i is a self-loop, B_i is triangular as shown in the previous section.
Consider the case that the cycle C_i of Q_i is not a self-loop and suppose arc (v,w) belongs to the arc set of C_i. Omitting (v,w) from Q_i turns Q_i into a tree.
First, sequence the rows and columns which correspond to the nodes and arcs in C_i (except the column associated with arc (v,w)) in the way nodes and arcs are passed in traversing the unique path from v to w in the tree.
Next, add the column corresponding to arc (v,w). The submatrix of B_i which now has been obtained is one-triangular.
The remaining part of Q_i has a tree structure and, as shown in Section 2.4, can be written in triangular form. □

EXAMPLE 2.5.13.



	12	32	13	14	53	75	56
1	-1		-1	-1			
2	g_{12}	g_{32}					
3		-1	g_{13}		g_{53}		
4				g_{14}			
5					-1	g_{75}	-1
7						-1	
6							g_{56}

Figure 2.5.1. A quasi-tree and a corresponding one-triangular matrix representation.

The properties of a basis and associated basis graph, mentioned in Theorems 2.5.10 and 2.5.12, make it possible to perform the steps of the Simplex algorithm by using the basis graph (a forest of quasi-trees) instead of the basis inverse B^{-1}.
A clarification of the calculations, to be carried out in the distinct steps of the Simplex algorithm, is presented.

Solving $\pi' B = c_B'$

In order to determine the Simplex multipliers the system

2.5.12 $\quad \pi' B = c_B'$

must be solved (cf. 2.3.12). In network terms this can be done in the following way (cf. the constructive proof of Theorem 2.5.12):
Consider each quasi-tree in the basis graph separately and distinguish the two cases:

(a) The cycle of the quasi-tree is a self-loop, say arc (v,v). Then $\pi_v = - c_{vv}/g_{vv}$.
The remaining part is dealt with as described in Section 2.4, where for each basic arc (i,j) the following relation holds:

2.5.13. $\quad - \pi_i + g_{ij}\pi_j = c_{ij}$.

(b) The cycle of the quasi-tree contains two or more arcs. Suppose arc (v,w) belongs to the cycle. P_{vw} denotes the path from v to w in the quasi-tree in which arc (v,w) is not contained. First, calculate all π_i for all nodes $i \in P_{vw}$ in terms of π_v using 2.5.13. Next, π_v is found from

$$- \pi_v + g_{vw}\pi_w = c_{vw}$$

and the π_i ($i \in P_{vw}$) are known too.
The remaining part of the quasi-tree has a tree structure and the Simplex multipliers of nodes in that part are determined as in Section 2.4, using 2.5.13 instead of 2.4.12.
The close relationship with the pure network situation is obvious. Here the Simplex multipliers are determined in some sequence "from the cycle towards the leaves". In actual implementations the cycle factors are used to speed up these calculations.
The Simplex multipliers are determined in this way in the initialization step of the Simplex algorithm. In all other steps the previous vector π is updated.

Solving $Bx = b^*$

The activity levels of the basic variables (the flow levels in the basic arcs) can be found by solving the system:

2.5.14. $Bx_B = b^*$,

where $b^* = b - N_2 u_{N_2}$ (formula 2.3.11).
In a similar way as the Simplex multipliers are evaluated these activity levels are calculated "from the leaves towards the cycle".
They are determined in this way only in the initialization step. In all other steps they can be found, as usual, by updating the previous vector x.

Equations of the form $Bx = b^*$ must also be solved in determining the representation of the entering column, say $a_{\cdot k\ell}$, in terms of the basis:

2.5.15. $By_{k\ell} = a_{\cdot k\ell}$.

This can be done in an easier way than indicated above, because the right-hand side of 2.5.15 has a special form. Associated with $a_{\cdot k\ell}$ is arc (k,ℓ). Since $a_{\cdot k\ell}$ can be written as

2.5.16. $a_{\cdot k\ell} = - e_k + g_{k\ell} e_\ell$,

if (k,ℓ) is not a self-loop, or as

2.5.17. $a_{\cdot k\ell} = - g_{kk} e_k$

if (k,ℓ) is a self-loop ($k = \ell$), the essential question is to find the representation of e_k (and e_ℓ) in terms of B (vectors $B^{-1} e_k$ and $B^{-1} e_\ell$). Denote by W_i the set of arcs, which belong to the path from node $i \in N$ to the cycle in the quasi-tree in which node i is contained, plus all arcs in this cycle.

THEOREM 2.5.14. *All columns of* B *which are not associated with arcs in* W_i *have a zero coefficient in the representation* $B^{-1} e_i$ *of* e_i *in terms of the basic columns* ($i \in N$).

PROOF. See ELAM, GLOVER & KLINGMAN [1979]. □

For this reason the vector $B^{-1} e_i$ is called the *cycle-path* vector of node i. An explicit formula for the representation $B^{-1} e_i$ of e_i, $i \in N$, in terms of

the basic columns can be found in ELAM, GLOVER & KLINGMAN [1979]. Regarding 2.5.15-2.5.17, vector $y_{\cdot k\ell}$ is found from:

2.5.18. $$y_{k\ell} = - B^{-1} e_k + g_{k\ell} B^{-1} e_\ell$$

if (k,ℓ) is not a self-loop, or from

2.5.19. $$y_{k\ell} = - g_{kk} B^{-1} e_k$$

if (k,ℓ) is a self-loop $(k = \ell)$.

Theorem 2.5.14 does not imply that all columns of B associated with arcs in $W_k \cup W_\ell$ have a nonzero coefficient in the representation vector $y_{k\ell}$. This is illustrated in Figure 2.5.2 where the set of arcs in $W_k \cup W_\ell$ is depicted by heavy lines for one of the cases which may occur. In case $g_{0k}\, g_{k\ell} / g_{0\ell} = 1$ only arcs $(0,k)$ and $(0,\ell)$ have a nonzero coefficient (cf. the situation in pure networks and theorem 2.5.11).

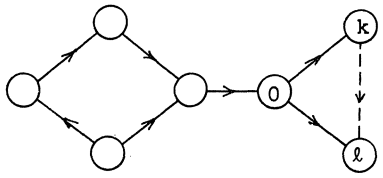


Figure 2.5.2. A possible union of W_k and W_ℓ.

Finally, an outline is given of the Simplex algorithm for generalized network problems.

Simplex algorithm for the minimal cost flow problem in a generalized network.

Initialization

The same starting basis as described in Section 2.4 can be used. Alternatives can be found, e.g., in GLOVER, HULTZ, KLINGMAN & STUTZ [1978].

1. Determine the Simplex multipliers

The Simplex multipliers can be evaluated as described above. However, after each basis change it is possible to update the previous vector π. This is discussed in step 7.

2. Calculate the reduced costs

The reduced costs are found from

2.5.20. $\bar{c}_{ij} = -\pi_i + g_{ij}\pi_j - c_{ij} \qquad (i,j) \in A$.

3. Perform the optimality test

This is standard (see Section 2.3).

4. Choose the nonbasic variable to enter the basis

Standard. Let $a_{\cdot k\ell}$ enter the basis.

5. Find the representation of $a_{\cdot k\ell}$ in terms of B

Determine the vector $y_{k\ell}$ from

$$By_{k\ell} = a_{\cdot k\ell}$$

as indicated above.

6. Perform the minimal ratio test

See Section 2.3. Suppose $a_{\cdot st}$ leaves the basis.

7. Update

Updating the objective function value and flow levels is standard.
The new basis graph is obtained from the previous one by omitting arc (s,t) and adding arc (k,ℓ). The Simplex multipliers associated with nodes in unchanged quasi-trees or cycles remain the same.
Only in case a new cycle is formed or tree parts are attached to another quasi-tree, the associated Simplex multipliers must be calculated in the way described before.
Continue with step 2.

3. PURE PROCESSING NETWORKS

3.1. Introduction

This chapter is concerned with pure processing networks. In Section 3.2 we discuss two distinct LP-formulations of the minimal cost flow problem in such a network.
The basis structure will be explained in terms of the network (Section 3.3) and subsequently exploited in a specification of the primal Simplex algorithm, discussed in Section 3.4.
A somewhat different specification of the primal Simplex algorithm for pure processing networks is presented in Section 3.5.
Finally, in Section 3.6, some remarks are made.

3.2. Mathematical formulation

A verbal description of a processing network is given in Subsection 1.1.2. The present section provides two distinct LP-formulations of the minimal cost flow problem in a pure processing network.
The first formulation is that of a pure network problem with side constraints.
The second one is more compact and can be seen as a pure network problem with side activities.
Consider a directed and connected graph G(N,A), with node set N, containing m nodes and arc set A, consisting of n arcs.
If self-loops (i,i), $i \in N$, are present in G(N,A) they can be replaced by arcs (i,m+1), where (m + 1) is an additional node.

ASSUMPTION 3.2.1. G(N,A) *does not contain any self-loop.*

It is convenient, and in many practical situations natural, to assume that G(N,A) satisfies some special topological properties (cf. the discussion at the end of this section). These properties will show up in the subsequent discussion and are summarized in Remark 3.2.2.

The node set N can be partitioned into three subsets:

RN: refining nodes
BN: blending nodes
TN: transportation nodes.

A *refining node* i ($i \in RN$) is a node with one incoming arc and at least two outgoing arcs. The flow on each outgoing arc (i,j), $j \in A(i)$, is required to be a given fraction α_{ij} of the total flow entering node i (see Figure 3.2.1a).
It is assumed that

3.2.1. $\quad 0 < \alpha_{ij} < 1 , \quad j \in A(i) , \quad i \in RN$

and

3.2.2. $\quad \sum_{j \in A(i)} \alpha_{ij} = 1 , \quad i \in RN .$

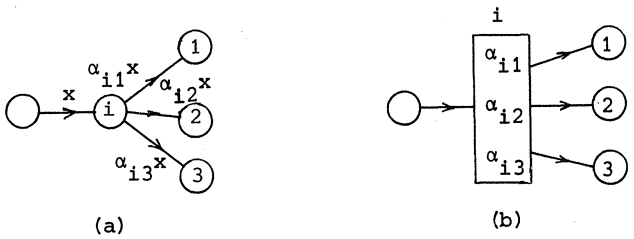


Figure 3.2.1. A refining node i.

A *blending node* i ($i \in BN$) is a node with at least two incoming arcs and only one outgoing arc. The flow on each incoming arc (j,i), $j \in B(i)$, is required to be a given fraction α_{ji} of the total flow leaving node i (see Figure 3.2.2a).
In analogy with refining nodes we have:

3.2.3. $\quad 0 < \alpha_{ji} < 1 , \quad j \in B(i) , \quad i \in BN$

and

3.2.4. $\sum_{j \in B(i)} \alpha_{ji} = 1 , \qquad i \in BN .$

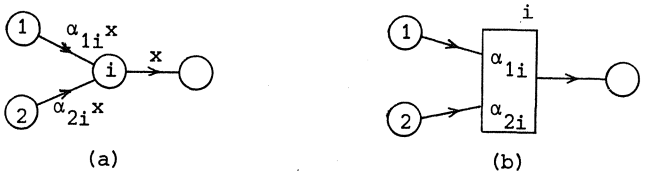


Figure 3.2.2. A blending node i.

All nodes of N which are neither refining nodes nor blending nodes are called *transportation nodes*.

It is assumed that, if i is a refining node or a blending node, all nodes $j \in A(i) \cup B(i)$ are transportation nodes.

REMARK 3.2.2. Note that in G(N,A) the following topological properties hold:

(a) if node $i \in N$ is a refining node (blending node) proportionality of flow is assumed on <u>all</u> outgoing (incoming) arcs of i;

(b) if node $i \in N$ is a refining node (blending node) there is exactly <u>one</u> incoming (outgoing) arc of node i;

(c) if node $i \in N$ is a refining node or blending node all nodes $j \in A(i) \cup B(i)$ are transportation nodes. □

Finally, it is convenient to introduce the set of *processing nodes* PN:

3.2.5. $PN := RN \cup BN .$

A *refining process* i is formed by the outgoing arcs of a refining node i.
Such arcs are called *refining arcs*.
The set of refining arcs contained in A is denoted by RA.
A *blending process* i is formed by the incoming arcs of a blending node i.
Such arcs are called *blending arcs*.
The set of blending arcs $\subseteq$ A is denoted by BA.
All arcs in A which are neither refining arcs nor blending arcs are called *transportation arcs*.
The set of transportation arcs $\subseteq$ A is denoted by TA.

We will say that a transportation arc (i,j) $\in$ A describes a *transportation process* (i,j).
Note that the incoming (outgoing) arc of a refining (blending) node describes a transportation process.
Also note that the arc set A is truely partitioned into the subsets RA, BA and TA.

The set of *processing arcs* PA is defined by:

3.2.6. PA := RA $\cup$ BA .

The coefficients α_{ij} in 3.2.1 or 3.2.3 are called *processing coefficients*.
A network with at least one processing node is called a *processing network*.
Conservation of flow is assumed in every node i $\in$ N. If, in addition, flow is conserved on the arc set A (no losses or gains in transporting flow along arcs), the network is addressed as a *pure* processing network.
Otherwise it is called a *generalized* processing network. Such networks will be discussed further in Chapter 4.

Before passing over to the mathematical formulations of the minimal cost flow problem in a pure processing network, some other notation and definitions are introduced.
PA(i) denotes the set of processing arcs incident to node i $\in$ PN.
In other words: PA(i) describes the set of arcs which correspond to refining or blending process i.
N(i) is the set of nodes which are incident to the arcs in PA(i), i $\in$ PN.
The number of arcs in PA(i) is called the *order* of process i and is denoted by n_i. Note that $n_i \geq 2$, $\forall$ i $\in$ PN.
Finally, it is remarked that in drawing diagrams of processing networks, it is convenient to distinguish the three types of nodes. Refining nodes and blending nodes will be represented as in Figure 3.2.1b and Figure 3.2.2b, transportation nodes are given by a small circle. An example of a processing network is presented in Figure 3.2.3.

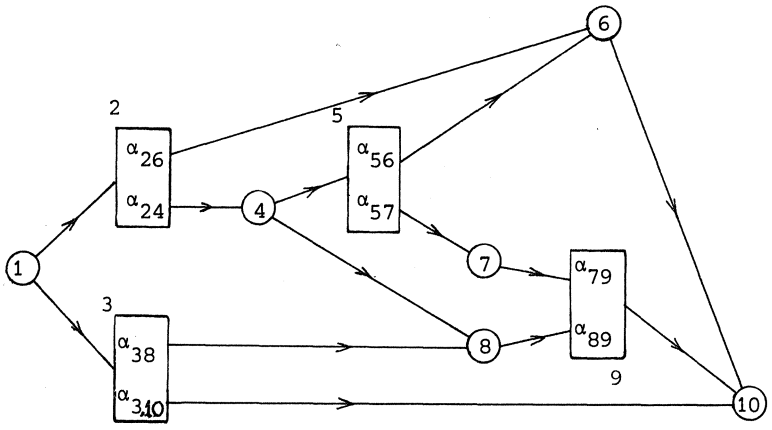


Figure 3.2.3. An example of a processing network.

Formulation I

The proportionality requirements in a refining or blending process can be stated in several ways.

Consider a refining process i with its corresponding refining node i.

A quite natural way to capture the proportionality requirements would be to express the flows on the outgoing arcs of node i in terms of the flow on the incoming arc. However, in view of the subsequent discussions, the following way will appear to be more appropriate:

Choose an arbitrary outgoing arc (i,r), $r \in A(i)$, of node i. It is clear that if the flow in (i,r) is known, flows on all outgoing arcs of node i are known too. For this reason arc (i,r) is called the *representative arc* of process i (or also of the set PA(i)).

The flows on all other outgoing arcs of i can be expressed in terms of the flow on arc (i,r):

$$\frac{x_{ij}}{x_{ir}} = \frac{\alpha_{ij}}{\alpha_{ir}} , \qquad r \in A(i) , \quad j \in A(i) \setminus \{r\}$$

or

3.2.7. $$\frac{\alpha_{ij}}{\alpha_{ir}} x_{ir} - x_{ij} = 0 , \qquad r \in A(i) , \quad j \in A(i) \setminus \{r\}$$

where all α_{ij}'s satisfy 3.2.1 and 3.2.2.

Perhaps it would be more appropriate to write r(i) instead of r, but for notational simplicity this has not been done.
In a similar way the proportionality requirements in a blending process can be stated:
Consider a blending process i with its corresponding blending node i.
Choose a representative arc (r,i) of process i ($r \in B(i)$) and formulate the blending requirements as:

3.2.8. $$\frac{\alpha_{ji}}{\alpha_{ri}} x_{ri} - x_{ji} = 0 , \qquad r \in B(i) , \quad j \in B(i) \setminus \{r\}$$

where all the α_{ji}'s satisfy 3.2.3 and 3.2.4.
The LP-formulation of the minimal cost flow problem in a pure processing network G(N,A) is:

3.2.9. $$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

3.2.10. $$- \sum_{j\in A(i)} x_{ij} + \sum_{j\in B(i)} x_{ji} = b_i , \quad i \in N$$

3.2.11. $$\frac{\alpha_{ij}}{\alpha_{ir}} x_{ir} - x_{ij} = 0 , \quad i \in RN , \; r \in A(i) , \; j \in A(i) \setminus \{r\}$$

3.2.12. $$\frac{\alpha_{ji}}{\alpha_{ri}} x_{ri} - x_{ji} = 0 , \quad i \in BN , \; r \in B(i) , \; j \in B(i) \setminus \{r\}$$

3.2.13. $$0 \le x_{ij} \le u_{ij} , \quad (i,j) \in A .$$

Equations 3.2.10 are the conservation of flow equations in which $b_i > 0$ denotes the external demand and $b_i < 0$ denotes the external supply in node i.
Formulae 3.2.11 and 3.2.12 are the refining and blending requirements.
Capacity bounds are given by 3.2.13. The next assumption is not restrictive.

ASSUMPTION 3.2.3. *For each refining process* i*:*

$$\frac{u_{ij}}{u_{ir}} = \frac{\alpha_{ij}}{\alpha_{ir}} , \qquad r \in A(i) , \quad j \in A(i) \setminus \{r\}$$

and for each blending process i*:*

$$\frac{u_{ji}}{u_{ri}} = \frac{\alpha_{ji}}{\alpha_{ri}} , \qquad r \in B(i) , \quad j \in B(i) \setminus \{r\} .$$

Formulation 3.2.9-3.2.13 is in fact one of a pure network problem (3.2.9, 3.2.10 and 3.2.13) with side constraints 3.2.11 and 3.2.12. Therefore CHEN & SAIGAL's algorithm [1977] can be used to solve 3.2.9-3.2.13; see also Subsection 1.1.1. For this problem a working basis of fixed size: $\Sigma_{i \in PN}(n_i - 1)$, i.e., the number of constraints in 3.2.11 and 3.2.12, would be required. However, the solution procedures, developed in Sections 3.4 and 3.5 (based on formulation II), use a working basis of variable size q, with $0 \leq q \leq \Sigma_{i \in PN} 1$ $(= |PN|)$, which is in general much smaller than $\Sigma_{i \in PN}(n_i - 1)$. The structure of the coefficient matrix of the left hand sides of 3.2.10-3.2.12 is clarified in Figure 3.2.4. In this figure p_R and p_B denote the number of refining and blending processes. Moreover, the matrices R_i $(i = 1,\ldots,p_R)$ and B_i $(i = 1,\ldots,p_B)$ are $(n_i - 1) \times n_i$ matrices with the following structure:

$$R_i = \begin{bmatrix} \frac{\alpha_{ij_1}}{\alpha_{ir}} & -1 & & \\ \frac{\alpha_{ij_2}}{\alpha_{ir}} & & \ddots & \\ \vdots & & & -1 \end{bmatrix} , \quad B_i = \begin{bmatrix} \frac{\alpha_{j_1 i}}{\alpha_{ri}} & -1 & & \\ \frac{\alpha_{j_2 i}}{\alpha_{ri}} & & \ddots & \\ \vdots & & & -1 \end{bmatrix} .$$

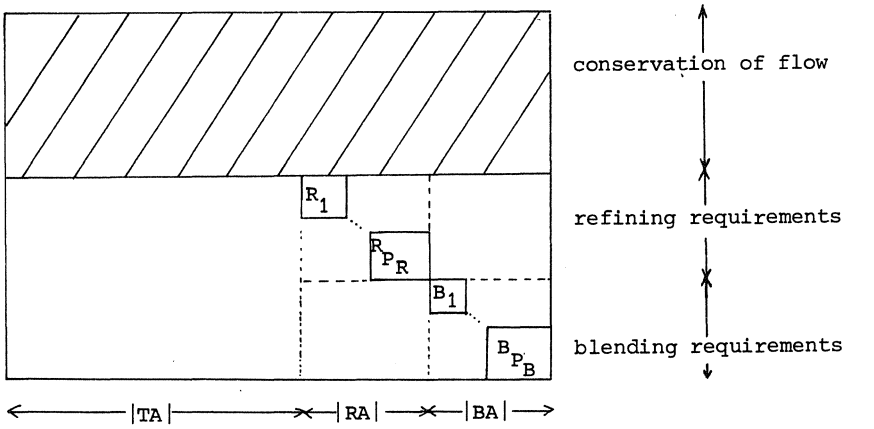


Figure 3.2.4. Structure of the coefficient matrix.

Note that in this formulation each refining or blending process i has n_i associated columns in the coefficient matrix.

Formulation II

An alternative, more compact, formulation is obtained from formulation I if the expressions for x_{ij} (formula 3.2.11):

$$x_{ij} = \frac{\alpha_{ij}}{\alpha_{ir}} x_{ir} , \qquad i \in RN , \quad r \in A(i) , \quad j \in A(i) \setminus \{r\}$$

and for x_{ji} (formula 3.2.12):

$$x_{ji} = \frac{\alpha_{ji}}{\alpha_{ri}} x_{ri} , \qquad i \in BN , \quad r \in B(i) , \quad j \in B(i) \setminus \{r\}$$

are substituted into 3.2.10.
Then each refining process and each blending process is represented by a single column in the resulting coefficient matrix A. Of course, the variable associated with this column in A describes the flow level in the representative arc of this process.
Matrix A has m rows and each row i of A can be identified by node i in the network. Each column of A describes one of the three possible types of processes (a column of A associated with refining or blending process i is denoted by $a_{\cdot i}$, a column of A associated with transportation process (i,j) is denoted by $a_{\cdot ij}$):

(a) refining process i. The elements in column $a_{\cdot i}$ are:

$-1/\alpha_{ir}$	in row i,
α_{ij}/α_{ir}	in row j, $j \in A(i)$,
0	otherwise.

(b) blending process i. The elements in column $a_{\cdot i}$ are:

$1/\alpha_{ri}$	in row i,
$-\alpha_{ji}/\alpha_{ri}$	in row j, $j \in B(i)$,
0	otherwise.

(c) transportation process (i,j). The elements in column $a_{\cdot ij}$ are:

-1	in row i,
1	in row j,
0	otherwise.

Figure 3.2.5 clarifies the structure of A.

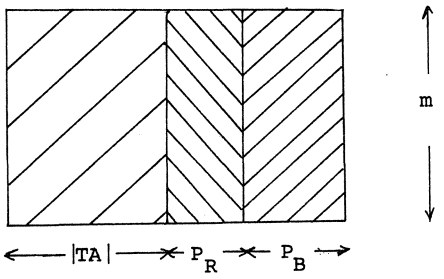


Figure 3.2.5. Structure of coefficient matrix A.

REMARK 3.2.4. It can easily be observed that matrix A has the following properties:

1. the sum of elements of each column in A is zero;
2. if there is more than one positive (negative) element in some column of A, there is only one negative (positive) element. □

Note that a column $a_{\cdot i}$ as meant under (a) or (b) has a unique representation except for some scaling factor (this scaling factor depends on the choice made for the representative arc).
In the rest of this monograph it is assumed that all columns of A are scaled such that the only negative (or positive) element in a column is equal to -1 (+1, respectively). So a refining process i has elements:

-1 in row i.
α_{ij} in row j, $j \in A(i)$,
0 otherwise,

and a blending process i has elements:

$+1$ in row i,
$-\alpha_{ij}$ in row j, $j \in B(i)$,
0 otherwise.

After scaling the variable associated with column $a_{\cdot i}$ describes the total throughput of process i.

The solution procedures of this chapter will use the compact formulation:

3.2.14. minimize $c'x$

3.2.15. $Ax = b$

3.2.16. $0 \leq x \leq u$,

where A satisfies the properties mentioned above.
Note that this formulation is one of a pure network problem with side activities. Therefore 3.2.14-3.2.16 can be solved by the Simplex SON approach of GLOVER & KLINGMAN [1981]. Their procedure would employ a working basis of the same size as the Simplex PRON procedures of Sections 3.4 and 3.5. However, Simplex SON does not make any specific use of the typical processing network structure.
The dual problem of 3.2.14-3.2.16 is given by:

3.2.17. maximize $b'\pi - u'\nu$

3.2.18. $$-\pi_i + \pi_j - \nu_{ij} \leq c_{ij} , \qquad (i,j) \in TP$$

3.2.19. $$-\pi_i + \sum_{j \in A(i)} \alpha_{ij} \pi_j - \nu_i \leq c_i , \qquad i \in RP$$

3.2.20. $$\pi_i - \sum_{j \in B(i)} \alpha_{ji} \pi_j - \nu_i \leq c_i , \qquad i \in BP$$

$$\nu \geq 0$$

where TP denotes the set of transportation processes, RP represents the set of refining processes and BP the set of blending processes.

It is emphasized that formulation 3.2.14-3.2.16 should merely be considered as a compact way of writing 3.2.8-3.2.13. The network interpretation remains the same:
Column $a_{\cdot i}$, associated with refining process i, can be written as

3.2.21. $$a_{\cdot i} = \sum_{j \in A(i)} \alpha_{ij} a^*_{\cdot ij} ,$$

where $a^*_{\cdot ij}$ denotes the vector representation of arc (i,j) (see Section 2.4). Formula 3.2.21 makes clear that the set of processing arcs PA(i) can still be associated with refining process i.
Of course, a similar statement can be made for a blending process i.

Note that a pure network (Section 2.4) can be considered as a "degenerate" case of a pure processing network (with processes of order 1).

The remaining part of this section discusses the possibility to define processing networks in a more general way, namely as a network in which the following three properties hold:

1. For generalized processing networks: conservation of flow in nodes, for pure processing networks: conservation of flow, both in nodes and on arcs.
2. Proportionality of flow in subsets of the arc set A. In each such a subset the arcs are incident to one common node and they are all directed either from or towards this node.
3. Capacity bounds on arcs.

An example of a processing network in this more general sense is presented in Figure 3.2.6. The subsets of arcs on which proportionality of flow is required are: $S_I = \{(1,4),(1,6)\}$, $S_{II} = \{(1,8),(1,10)\}$, $S_{III} = \{(4,6),(4,7)\}$ and $S_{IV} = \{(7,10),(8,10)\}$. The α_{ij}'s beside the arcs in S_I, S_{II}, S_{III} and S_{IV} are the proportionality coefficients.

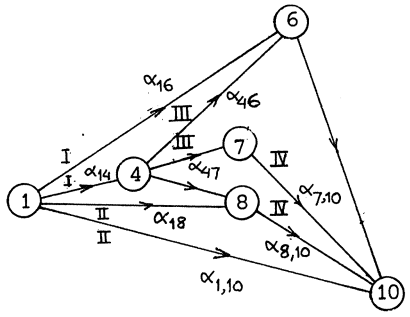


Figure 3.2.6. A processing network in the more general sense.

It is remarked that, after adapting the α_{ij}'s in an obvious way, the processing network in Figure 3.2.6 is in fact equivalent to the one drawn in Figure 3.2.3.

If we would define:

a refining node i as a node of N for which a subset of the set $\{(i,j) \mid j \in A(i)\}$ exists with proportionality of flow on the arcs in this subset,

a blending node i as a node of N for which a subset of the set $\{(j,i) \mid j \in B(i)\}$ exists with proportionality of flow on the arcs in this subset

and

a transportation node i as a node of N which is not a refining or blending node,

then it is immediately clear that none of the properties mentioned in Remark 3.2.2 have to hold.

Yet, in the remaining part of this monograph it is assumed that a processing network satisfies the properties mentioned in Remark 3.2.2.

The motivation for doing this is:

1. In many practical situations it is natural to assume <u>one</u> incoming (outgoing) arc of a refining (blending) node and proportionality of flow on <u>all</u> outgoing (incoming) arcs (for instance, a destillation column in an oil refinery).
2. The network diagrams which can be drawn have a simpler structure (compare Figure 3.2.3 with Figure 3.2.6) and are therefore easier to interpret. Visualization is an important aspect which will be discussed further in Chapter 5.
3. The assumptions simplify the way to think about processing networks as well as the notation.

We emphasize the following facts:

1. It is <u>in no way restrictive</u> to assume that the properties in Remark 3.2.2 are satisfied in a processing network: by introducing additional transportation nodes and/or transportation arcs an arbitrary processing network can be cast into this framework.
2. The special topological properties will <u>not</u> be <u>used in an essential way</u> in the subsequent discussions (only for notational convenience). Consequently, the solution procedures developed in the sequel can easily be adapted to suit processing network problems in the more general sense.

As shown above, a pure processing network problem can be formulated as in 3.2.14-3.2.16, where matrix $\hat{A}$ has the properties mentioned in Remark 3.2.4. Conversely, a LP-problem 3.2.14-3.2.16 in which $\hat{A}$ has the properties described in Remark 3.2.4 can be considered as a pure processing network problem in the more general sense.

EXAMPLE 3.2.5. Let $\hat{A}$ be given by:

$$\hat{A} = \begin{bmatrix} -1 & -1 & & & & & \\ \alpha_{14} & & -1 & -1 & & & \\ \alpha_{16} & & & \alpha_{46} & & -1 & \\ & & & \alpha_{47} & -\alpha_{7,10} & & \\ & \alpha_{18} & 1 & & -\alpha_{8,10} & & \\ & \alpha_{1,10} & & & 1 & 1 & \end{bmatrix} \begin{matrix} 1 \\ 4 \\ 6 \\ 7 \\ 8 \\ 10 \end{matrix}$$

where all α_{ij}'s are positive and $e'\hat{A} = 0$.
Then $\hat{A}$ reflects the pure processing network structure of Figure 3.2.6.

3.3. Basis structure

The basis structure will be explained in terms of the pure processing network. Both formulations of Section 3.2 can be used, leading to essentially the same results. Here we use the compact formulation (formulation II) considering the fact that this formulation will also be used for the solution procedures of Sections 3.4 and 3.5. An explanation of the basis structure in terms of the first formulation can be found in KOENE [1981a].

The rank of matrix $\hat{A}$ in 3.2.15 is obviously smaller than or equal to $(m-1)$, since $e'\hat{A} = 0$.

ASSUMPTION 3.3.1. *The rank of* $\hat{A}$ *equals* $(m-1)$.

REMARK 3.3.2. A sufficient (not necessary) condition, to let $\hat{A}$ have rank $(m-1)$, is that the directed graph with node set N and as arc set the set of transportation arcs $TA \subseteq A$, is connected. In that case a spanning tree, containing transportation arcs only, can be constructed in G(N,A). As mentioned in Section 2.4 the submatrix of $\hat{A}$ which describes such a spanning tree has rank $(m-1)$. Note that dummy transportation arcs can always be introduced such that this is the case. □

As in pure networks (Section 2.4) we introduce a single artificial variable $x_{i_0 i_0}$ with $a_{\cdot i_0 i_0} = -e_{i_0}$ (i_0 arbitrarily chosen from $\{1,\ldots,m\}$. Again it is easy to prove that matrix

3.3.1 $A^* = [-e_{i_0} \; A]$

has rank m.
Let B denote a basis of A^*. Column $-e_{i_0}$ always belongs to B and can be thought to represent the self-loop (i_0,i_0).
Suppose B contains q processing columns (i.e., the columns of A associated with a refining or blending process), $0 \leq q \leq m-1$, and, consequently, $m-q$ transportation columns (columns of A associated with a transportation process), including the slack column $-e_{i_0}$.
Matrix B can be partitioned as:

3.3.2. $B = [-e_{i_0} \; B_T \; B_P]$

where B_T is an $m \times (m-q-1)$ matrix denoting the structural basic transportation processes, and B_P is an $m \times q$ matrix representing the basic refining and blending columns.
Let $\underline{B}$ denote the matrix:

3.3.3. $\underline{B} = [B_T \; B_P]$.

The set of basic refining and blending processes is given by BAP.

Define the *basis graph* associated with B as the directed graph with node set N and as arc set:
the self-loop (i_0,i_0),
the transportation arcs $\subseteq$ A associated with the columns in B_T, and
all processing arcs $\subseteq$ A associated with the columns in B_P, i.e., all arcs in PA(i), $i \in$ BAP (cf. formula 3.2.21 and the definition of PA(i) given in the previous section).

The purpose of the subsequently stated lemmas is to explain the structure of the basis graph.
The arc set of a basis graph consists of a number of transportation arcs and a number of processing arcs.

LEMMA 3.3.3. *Except for the self-loop* (i_0,i_0) *the basis graph contains no cycle with only transportation arcs.*

PROOF. This fact follows immediately from the theory of pure networks (Section 2.4). □

Knowing this, consider for a moment the basis graph in which all processing arcs and the self-loop (i_0,i_0) are left out. This graph consists of a number of connected components, each of which cannot contain any cycle. Such a connected component must therefore have a tree structure and is called a *transportation tree*. A transportation tree may consist of a single node.
The next lemma gives a relation between the number of basic refining and blending processes (i.e., the number of elements in BAP) and the number of transportation trees contained in a basis graph. The number of basic refining and blending processes is given by q ($0 \leq q \leq m-1$).

LEMMA 3.3.4. *A basis graph contains* $(q+1)$ *transportation trees iff the number of basic refining and blending processes equals* q.

PROOF. If there are q basic refining and blending processes the basis graph must contain $m-(q+1)$ transportation arcs apart from the self-loop (i_0,i_0). Considering the following two facts:
- the number of arcs in a tree is exactly one less than the number of nodes in a tree,
- each of the m nodes of N belongs to some transportation tree,

it is immediately clear that the basis graph must contain $(q+1)$ transportation trees.
The other part of the proof is obtained by reversing the argument. □

According to Remarks 2.4.5 and 3.2.4 Lemma 2.4.4 of the previous section is also valid for pure processing networks, with $\underline{B}$ as in 3.3.3.
In addition, it is also possible to state a lemma closely related to Lemma 2.4.4.
Suppose $BAP \neq \emptyset$.
Let S_p be a nonempty subset of BAP.
If a node of the set N(i), $i \in$ BAP, belongs to some transportation tree T_ℓ, process i and transportation tree T_ℓ are said to be *incident* to each other.
Let $T(S_p)$ denote the set of transportation trees which are incident to the processes $i \in S_p$.

LEMMA 3.3.5. *Any set* S_p *of basic refining and blending processes is incident to at least* $|S_p|+1$ *transportation trees:*

3.3.4. $\quad |T(S_p)| \geq |S_p|+1$

PROOF. Suppose the transportation trees in $T(S_p)$ contain in total t nodes and, consequently, $t - |T(S_p)|$ arcs. These $t - |T(S_p)|$ arcs plus the $|S_p|$ refining and blending processes in S_p correspond to $t - |T(S_p)| + |S_p|$ columns in the basis B. According to the definition of $T(S_p)$, these columns contain only nonzero elements in the t rows of B which correspond to the nodes in $T(S_p)$.
Since the columns in a basis must be linearly independent and the column sum of each of the $t - |T(S_p)| + |S_p|$ columns is zero, a necessary condition is:

$$t - |T(S_p)| + |S_p| \leq t - 1$$

or

$$|T(S_p)| \geq |S_p| + 1 \ .$$ □

According to Lemma 3.3.4 equality in formula 3.3.4 clearly holds if S_p = BAP, but there may also be proper (nonempty) subsets of BAP for which equality holds too.
An immediate consequence of Lemma 3.3.5 is that every transportation tree must be incident to at least one process i, i ∈ BAP. For suppose there is some transportation tree for which this is not true, then the q processes in BAP would be incident to the remaining q transportation trees, which is impossible because of Lemma 3.3.5.
Lemma 3.3.5 can be used to prove that the representative arcs of the basic refining and blending processes can be chosen in a special way:

LEMMA 3.3.6. *The representative arcs of the basic refining and blending processes can be chosen in such a way that the basic transportation arcs associated with matrix* B_T *plus these representative arcs form the arc set of a spanning tree in* G(N,A).

Such a spanning tree will be called a *representative spanning tree* of the basis graph.

PROOF of LEMMA 3.3.6. A simple induction argument will be used.
If BAP = ∅ the statement is trivially true.
Suppose BAP ≠ ∅.
Let S_0 be a nonempty subset of BAP such that:

3.3.5. $|T(S_0)| = |S_0| + 1$,

and such that for every subset $S_1 \subseteq S_0$, $S_1 \neq \emptyset$, $S_1 \neq S_0$:

3.3.6. $|T(S_1)| > |S_1| + 1$.

Such a subset S_0 of BAP must clearly exist: it is either BAP itself or a proper subset of BAP.
Furthermore, consider a subset S_2 of BAP such that $S_0 \cap S_2 = \emptyset$.
Since $S_0 \cap S_2 = \emptyset$ Lemma 3.3.5 implies

3.3.7. $|T(S_0 \cup S_2)| \geq |S_0| + |S_2| + 1$.

Consequently, using formula 3.3.5:

3.3.8. $|T(S_0 \cup S_2)| - |T(S_0)| \geq |S_2|$.

Verbally stated: the set S_2 must be incident to at least $|S_2|$ transportation trees which are not contained in $T(S_0)$. Because S_1 is a subset of S_0 the following is also true (using 3.3.6):

3.3.9. $|T(S_1 \cup S_2)| \geq |T(S_1)| + |S_2| > |S_1| + |S_2| + 1$.

After these preparatory observations consider an arbitrary process i^* in S_0 and suppose the associated processing node i^* belongs to transportation tree T_k. Lemma 3.3.5 guarantees there is at least one node, say j^*, in $N(i^*)$ which belongs to some transportation tree T_ℓ ($\ell \neq k$).
Take the arc incident to i^* and j^* as the representative arc of process i^*.
This representative arc transforms the (transportation) trees T_k and T_ℓ into one new tree.
Leave process i^* out of the set BAP and consider the new situation:
there is one basic refining or blending process less and one tree less.
The statements in 3.3.6, 3.3.8 and 3.3.9 guarantee that in the new situation every nonempty subset of basic refining and blending processes again satisfies the condition of Lemma 3.3.5.
By induction the statement in Lemma 3.3.6 follows. □

Lemmas 3.3.3-3.3.6 provide the following essential theorem.

THEOREM 3.3.7. *A basis graph in a pure processing network* G(N,A) *consists of a rooted spanning tree formed by the self-loop* (i_0,i_0), *the basic*

transportation arcs in A *and the properly chosen representative arcs of the basic refining and blending processes, plus all nonrepresentative arcs of the basic refining and blending processes.*

An illustrative example is given in Figure 3.3.1, where the representative spanning tree is drawn with heavy lines. The associated basis matrix is given by:

3.3.10.
$$B = \begin{array}{c} \begin{matrix} 11 & 12 & 13 & 2 & 3 \end{matrix} \\ \begin{bmatrix} -1 & -1 & -1 & & \\ & 1 & & -1 & \\ & & 1 & & -1 \\ & & & \alpha_{24} & \alpha_{34} \\ & & & \alpha_{25} & \alpha_{35} \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

where $\alpha_{24} \neq \alpha_{34}$.

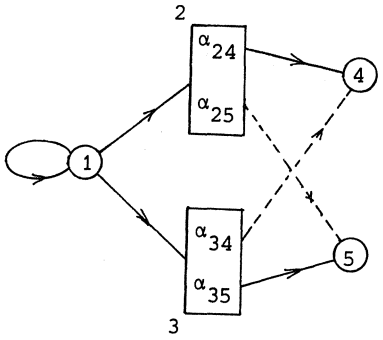


Figure 3.3.1. Example of a basis graph in a pure processing network.

This example demonstrates several important aspects:

1. The reverse of the statement in Theorem 3.3.7 is not necessarily true: a graph satisfying the properties mentioned in Theorem 3.3.7 is not necessarily a basis graph.
   If $\alpha_{24} = \alpha_{34}$, B would be singular (recall that $\alpha_{24} + \alpha_{25} = 1$ and $\alpha_{34} + \alpha_{35} = 1$) and, consequently, the graph drawn in Figure 3.3.1 would not be a basis graph.
   The same conclusion can be drawn as in generalized networks (Section 2.5):

Whether a subgraph of G(N,A) is a basis graph or not does not only depend on its topology but also on the values of the processing coefficients (the α_{ij}'s).

2. The proper choice of the representative arcs need not be unique: arcs (2,5) and (3,4) could also have been chosen to represent processes 2 and 3, respectively.

Considering Theorem 3.3.7 it is quite natural to give special attention to the representative spanning tree of the basis graph.
Let T be the matrix representation of the rooted representative spanning tree, such that each column $T_{\cdot j}$ of T corresponds to column $B_{\cdot j}$ of B ($j = 1,\ldots,m$). Then basis B can also be written as:

3.3.11. $B = TP$.

Matrix P in 3.3.11 can be specified as:

3.3.12. $$P = \begin{bmatrix} I & Q \\ & R \end{bmatrix},$$

in which
I is the identity matrix of order $(m-q)$,
Q is an $(m-q) \times q$ matrix, and
R is a square nonsingular matrix of order q.
For matrix B in 3.3.10 equation 3.3.11 becomes:

3.3.13.
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 11 & 12 & 13 & 2 & 3 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & & \\ & 1 & & -1 & \\ & & 1 & & -1 \\ & & & \alpha_{24} & \alpha_{34} \\ & & & \alpha_{25} & \alpha_{35} \end{bmatrix} \end{array} =$$

$$= \begin{bmatrix} -1 & -1 & -1 & & \\ & 1 & & -1 & \\ & & 1 & & -1 \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & & -\alpha_{25} & \alpha_{34} \\ & & 1 & \alpha_{25} & -\alpha_{34} \\ \hline & & & \alpha_{24} & \alpha_{34} \\ & & & \alpha_{25} & \alpha_{35} \end{bmatrix} .$$

$$\begin{array}{ccccc} 11 & 12 & 13 & 24 & 35 \end{array}$$

REMARK 3.3.8. Let $P_{\cdot j}$ be the j th column of matrix P ($m-q < j \leq m$) and $B_{\cdot j}$ the j th column of B, which is associated with refining or blending process $k = i_j$. According to 3.3.11:

3.3.14. $$P_{\cdot j} = T^{-1} B_{\cdot j} .$$

Suppose process k is a refining process, then $B_{\cdot j}$ can be written as in 3.2.21:

3.3.15. $$B_{\cdot j} = \sum_{\ell \in A(k)} \alpha_{k\ell} a^*_{\cdot k\ell} ,$$

where $a^*_{\cdot k\ell}$ denotes the vector representation of arc (k,ℓ) as in pure networks (see Section 2.4).
Using 3.3.14 and 3.3.15, $P_{\cdot j}$ can be written as:

3.3.16. $$P_{\cdot j} = \alpha_{kr} T^{-1} a^*_{\cdot kr} + \sum_{\ell \in A(k)\setminus\{r\}} \alpha_{k\ell} T^{-1} a^*_{\cdot k\ell} ,$$

where arc (k,r) denotes the representative arc of process $k = i_j$.
In the example presented, column $P_{\cdot 4}$ can be written as:

3.3.17. $$P_{\cdot 4} = \alpha_{24} T^{-1} \begin{bmatrix} \\ -1 \\ \\ 1 \\ \\ \end{bmatrix} + \alpha_{25} T^{-1} \begin{bmatrix} \\ -1 \\ \\ \\ 1 \end{bmatrix} = \alpha_{24} \begin{bmatrix} \\ 1 \\ \\ \\ \\ \end{bmatrix} + \alpha_{25} \begin{bmatrix} -1 \\ 1 \\ \\ 1 \\ \\ \end{bmatrix} .$$

If process $k = i_j$ is a blending process, in a similar way it can be derived that:

3.3.18. $$P_{\cdot j} = \alpha_{rk} T^{-1} a^*_{\cdot rk} + \sum_{\ell \in B(k)\setminus\{r\}} \alpha_{\ell k} T^{-1} a^*_{\cdot \ell k},$$

with arc (r,k) the representative arc of process $k = i_j$.
From 3.3.16 and 3.3.18 one observes that $P_{\cdot j}$ is in fact some positive linear combination of the j th unit vector (which results from the representative arc of process i_j) and the cycle vectors (defined in Section 2.4) of the nonrepresentative arcs of process i_j.
This observation plays an important role in the Simplex algorithm of the next section. □

REMARK 3.3.9. If the first column of B and T is $-e_1$ ($i_0 = 1$), then the first row of Q in 3.3.12 is a row of zeros. This is immediately clear from 3.3.16 and 3.3.18, considering the fact that the self-loop associated with column $-e_1$ never takes part in a cycle induced in the representative spanning tree by a nonrepresentative arc of a basic refining or blending process. □

The aggregated graph of the basis graph

Recall that Lemma 2.4.4, which is not only valid for pure networks, but also for pure processing networks, gives a relation between the basic processes (including the basic transportation processes) and the nodes in the network. The basis graph describes the interaction between the basic processes and the nodes of G(N,A).
Lemma 3.3.5 provides a similar relation between the basic refining and blending processes and the transportation trees. Here the aggregated graph of the basis graph, which will be introduced next, describes the interaction between the basic refining and blending processes and the transportation trees.
Consider a basis graph as described in Theorem 3.3.7 and let there be $(q+1)$ transportation trees $(0 \leq q \leq m-1)$.
The *aggregated graph* of the basis graph is defined as the directed graph with
node set $N^* = \{1,\ldots,q+1\}$, where each node i corresponds to transportation tree T_i $(i = 1,\ldots,q+1)$, and
arc set A^*, which consists of all arcs (u,v), $u,v \in N^*$, for which in the basis graph a processing arc exists with begin point in T_u and end point in T_v.
Figure 3.3.2 illustrates the aggregated graph of the basis graph in Figure 3.3.1. There are three transportation trees T_1, T_2 and T_3 of which the node sets are given by {1,2,3}, {4} and {5}, respectively. Beside each arc in Figure 3.3.2 the corresponding processing arc in the basis graph is denoted.

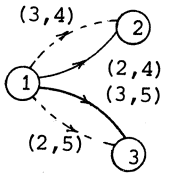


Figure 3.3.2. The aggregated graph of the basis graph in Figure 3.3.1.

Note that the arcs in the aggregated graph, which correspond to the representative arcs of the refining and blending processes in the basis graph, form the arc set of a spanning tree in the aggregated graph (in Figure 3.3.2 drawn with heavy lines).
Let the matrix representation of this spanning tree be given by $\bar{T}$. Clearly, $\bar{T}$ is a $(q+1) \times q$ matrix.
Consider the $m \times (q+1)$ matrix $V = [v_{ij}]$, where

3.3.19. $$\begin{cases} v_{ij} = 1 , & \text{if node } i \text{ of } N \text{ is contained in transportation tree } T_j, \\ v_{ij} = 0 , & \text{otherwise.} \end{cases}$$

Let B be written as in 3.3.2:

3.3.20. $$B = [-e_{i_0} \quad B_T \quad B_P] ,$$

and suppose that node $i_0 \in T_1$. Then the product V'B can be written as:

3.3.21. $$V'B = \underset{\leftarrow m-q\rightarrow}{[-e_1} \quad \underset{\leftarrow q\rightarrow}{0} \quad R^*] ,$$

where $R^* = [r^*_{ij}]$ is some $(q+1) \times q$ matrix. Note that, given the basis B, R^* is unique except for row permutations.
Considering the structure of V in 3.3.19, B_P in 3.3.20 and the fact that each basic refining or blending process is incident to at least two transportation trees (Lemma 3.3.5) it can be observed that $r^*_{ij} \neq 0$ iff the process associated with column j in R is incident to transportation tree T_i. Furthermore, is is easy to verify that R^* satisfies the properties of a matrix as described in Remark 3.2.4.

Properties of matrix R

Not only the representative spanning tree plays an important role in the subsequently discussed Simplex algorithm, but also matrix R in 3.3.12. R will be used as a working basis. We discuss some properties of R.

THEOREM 3.3.10. *The main diagonal of* R *in* 3.3.12 *is strictly positive.*

PROOF. It must be proved that each element p_{jj} $(m-q < j \leq m)$ of matrix P is strictly positive.
According to Remark 3.3.8 we can state the following facts:
A positive contribution to p_{jj} is given by the j th unit vector, which results from the representative arc of process i_j.
Each cycle vector of a nonrepresentative arc of process i_j gives
a zero contribution to p_{jj} if the representative arc of i_j is not contained in this cycle, and
a positive one if the representative arc is contained in this cycle.
The latter statement follows from the fact that, if the representative arc is present in such a cycle, it must be present as a backward arc, which has a "+1" in the cycle vector (see Section 2.4).
This completes the proof. □

Considering 3.3.11 and 3.3.12 we may expect that R depends on the specific choice of the representative arcs of the basic refining and blending processes. We will show that matrix R depends only on the specific form of matrix $\bar{T}$, which represents the spanning tree in the aggregated graph associated with the representative spanning tree in the basis graph.

THEOREM 3.3.11. *Given a basis* B, *matrix* R *in* 3.3.12 *depends only on the particular form of matrix* $\bar{T}$.

PROOF. First evaluate the product V'T:

3.3.22. $V'T = [-e_1 \quad 0 \quad \bar{T}]$.

Suppose we write (see Remark 3.3.9):

$$Q = \begin{bmatrix} 0 \\ Q^* \end{bmatrix} \leftarrow \text{row } 1 \quad ,$$

then P in 3.3.12 can be written as:

3.3.23. $$P = \begin{bmatrix} 1 & & \\ & I & Q^* \\ & & R \end{bmatrix} .$$

From 3.3.22 and 3.3.24 it can be seen that:

3.3.24. $V'TP = [-e_1 \quad 0 \quad \bar{T}R]$.

In view of the fact that $B = TP$ (formula 3.3.11), comparison of 3.3.21 and 3.3.24 results in

3.3.25. $R^* = \bar{T}R$.

Since R^* is unique (except for row permutations) the theorem has been proved. □

The basis structure will be exploited in a specification of the primal Simplex algorithm, presented in the next section.

3.4. *The Simplex algorithm for the minimal cost flow problem in a pure processing network*

In Section 2.3 an outline is given of the Simplex algorithm for general LP-problems with simple upper bounds. The present section discusses a specification of this algorithm for the minimal cost flow problem in a pure processing network, formulated by 3.2.14-3.2.16. The basis structure, discussed in the previous section, will be exploited in this specification.
It is assumed that the rooted representative spanning tree and the inverse R^{-1} of R in 3.3.12 are stored in some convenient way.
In several steps of the Simplex algorithm we will have to evaluate equations of the form $Tx = b^*$ or $\pi'T = c_B'$, where T describes the rooted representative spanning tree. This can be done in the way explained in Section 2.4. In the text we will simply state that the required quantities are determined by pure-network techniques.
At some places we will need (a submatrix of) matrix Q in 3.3.12 or a row of Q. According to formulae 3.3.11 and 3.3.12, Q can be determined directly from the original data by means of pure-network techniques (Section 2.4). Therefore, it is not necessary to store Q: the information required is determined when needed.

For expository reasons we first discuss the representation of the entering column in terms of the basic columns (step 5 in the algorithm of Section 2.3).

3.4.1. The representation of the entering column in terms of B

Let the column which enters the basis be given by a. Column a represents either a transportation process or a refining process or a blending process. In order to find the representation vector y of a in terms of the basis B we must solve the system:

3.4.1. $By = a$.

According to 3.3.11, y can be found from

3.4.2. $TPy = a$.

Hence the calculation of y can be split up in two portions:
First, determine the vector $\bar{y}$ from:

3.4.3. $T\bar{y} = a$,

using pure-network techniques (T denotes a rooted spanning tree).
Secondly, calculate y from

3.4.4. $Py = \bar{y}$.

In general, this two-step procedure involves less arithmetical operations than a direct evaluation of y from 3.4.1, as can be seen from the structure of P in 3.3.12.
These calculations can be accelerated even further by using the following labeling procedure, which determines
the basic processes which have in any case a zero coefficient in the representation vector y, and
the basic processes which possibly have a nonzero coefficient in y.
The labeling procedure attaches a two-index label to some of the arcs in the representative spanning tree. For detecting the structural zeros and nonzeros in vector y, it is only relevant whether an arc in the representative spanning tree is labeled or not. The labels themselves will be used later on in order to reestablish a representative spanning tree when the leaving process is known (that is after the minimal ratio test).
In the subsequent labeling procedure we consider a process labeled whenever its (representative) arc in the representative spanning tree has a label.

We consider a basic refining or blending process scanned if all the arcs contained in the cycles, induced by the nonrepresentative arcs of this process in the representative spanning tree, are labeled.

Labeling procedure

1. If the entering process is a transportation process, say (i^*,j^*), its corresponding arc (i^*,j^*) induces a single cycle in the representative spanning tree. Trace this cycle and attach the label $[i^*,j^*]$ to all the arcs in this cycle. Continue with step 3.
   Otherwise the entering process is a refining or blending process, say i^*. Put $W = PA(i^*)$.
2. Determine all the cycles, induced in the representative spanning tree by the arcs $(i,j) \in W$, one by one. Start tracing a cycle from node j if i is a refining node and from node i if j is a blending node and stop tracing a cycle as soon as a labeled arc is encountered. Label the arcs in these cycles in the following way.
   If i is a refining node, the arcs in the cycle induced by (i,j) in the representative spanning tree get the label [i,j].
   If j is a blending node, the arcs in the cycle induced by (i,j) in the representative spanning tree get the label [-j,i].
3. List all basic refining and blending processes which are labeled, but not scanned.
   If this list is empty, then stop: the labeled processes are the only ones that may have a nonzero coefficient in the representation vector y (see Theorem 3.4.3).
   Otherwise let W denote the set of all nonrepresentative arcs of the labeled, but not scanned, refining and blending processes. Continue with step 2.

REMARK 3.4.1. Note that, if a transportation process (i^*,j^*) enters the basis, which is incident to only one transportation tree (i.e., both i^* and j^* belong to the same transportation tree), the same situation occurs as in pure networks. □

EXAMPLE 3.4.2. Figure 3.4.1 shows the labeled part of a representative spanning tree, assuming that transportation process (4,5) enters the basis.

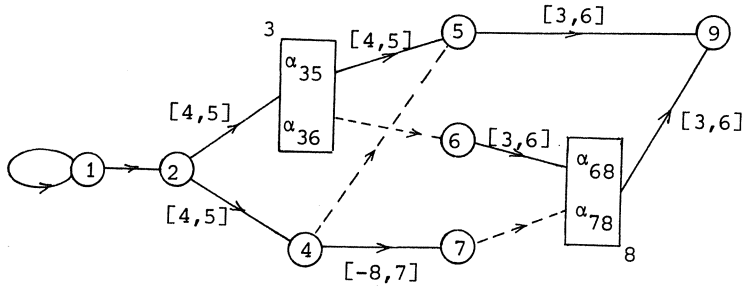


Figure 3.4.1. The labeled part of a representative spanning tree.

RYAN & CHEN [1981] discuss how cycles induced in a spanning tree can be traced.

Let us reexamine the relations 3.4.3 and 3.4.4 which have to be solved in order to find the representation vector y.
After completion of the first pass through step 3 of the labeling procedure vector $\bar{y}$ in 3.4.3 can easily be determined as in pure networks (see Section 2.4). One may verify that $\bar{y}$ has entries unequal to zero in all the rows (and only in those rows) which correspond to the then labeled arcs.
After completion of the labeling procedure the columns and rows of P can be partitioned symmetrically into four classes:

I columns (rows) associated with labeled transportation processes,
II columns (rows) associated with unlabeled transportation processes,
III columns (rows) associated with labeled refining and blending processes,
IV columns (rows) associated with unlabeled refining and blending processes.

Then equation 3.4.4 can also be written as:

3.4.5.
$$\begin{array}{c} \\ \text{I} \\ \text{II} \\ \text{III} \\ \text{IV} \end{array} \begin{array}{c} \begin{array}{cccc} \text{I} & \text{II} & \text{III} & \text{IV} \end{array} \\ \begin{bmatrix} I & & Q_1 & Q_2 \\ & I & Q_4 & Q_3 \\ & & R_1 & R_2 \\ & & R_4 & R_3 \end{bmatrix} \end{array} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \bar{y}_1 \\ 0 \\ \bar{y}_3 \\ 0 \end{bmatrix} .$$

THEOREM 3.4.3. *All nonlabeled processes have a zero coefficient in the representation vector* y *of the entering column* a.

PROOF. The basic observation which provides the proof is that Q_4 and R_4 in 3.4.5 must be zero matrices. For suppose $Q_4 \neq 0$. Then, according to Remark 3.3.8, there must be an unlabeled transportation arc which is contained in some cycle induced by a nonrepresentative arc of a labeled refining or blending process. However, this is impossible since all those cycles are traced and labeled in the labeling procedure. Consequently, $Q_4 = 0$.
Similarly it is proved that $R_4 = 0$.
Since R_3 must be a square nonsingular matrix, it follows from equation 3.4.5-IV that $y_4 = 0$, and, consequently, from 3.4.5-II that $y_2 = 0$.
This completes the proof. □

System 3.4.5 reduces to (with $y_2 = 0$ and $y_4 = 0$):

3.4.6. $$\begin{bmatrix} I & Q_1 \\ & R_1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_3 \end{bmatrix} .$$

Since R^{-1} is kept stored, y_3 immediately follows from:

3.4.7. $$y_3 = R_1^{-1} \bar{y}_3 .$$

Furthermore, y_1 can be found from

3.4.8. $$y_1 = \bar{y}_1 - Q_1 y_3 ,$$

where Q_1 can be determined by pure-network techniques (Section 2.4).
A more appropriate way to determine y_1 is the following: let the column partitioning of B, compatible with 3.4.5, be

3.4.9. $$B = [B_I, B_{II}, B_{III}, B_{IV}] .$$

Then y_1 can be determined from

3.4.10. $$B_I y_1 = a - B_{III} y_3 .$$

Since B_I and B_{III} are known, and B_I has a tree structure (B_I denotes the labeled transportation processes) this system can be solved by pure-network techniques.

From the above discussion it is clear that R can be written in block triangular form:

$$3.4.11. \quad R = \begin{bmatrix} R_1 & R_2 \\ & R_3 \end{bmatrix} .$$

The inverse of R is given by:

$$3.4.12. \quad R^{-1} = \begin{bmatrix} R_1^{-1} & R_0 \\ & R_3^{-1} \end{bmatrix} ,$$

where

$$3.4.13. \quad R_0 = - R_1^{-1} R_2 R_3^{-1} .$$

The following theorem plays an essential role in proving Theorem 6.2.3 in Chapter 6 and may also be important for implementations of the present Simplex algorithm (see Section 3.6).

THEOREM 3.4.4. *The number of basic refining and blending processes with a nonzero coefficient in the representation vector* y, *is zero iff the entering process is incident to only one transportation tree.*

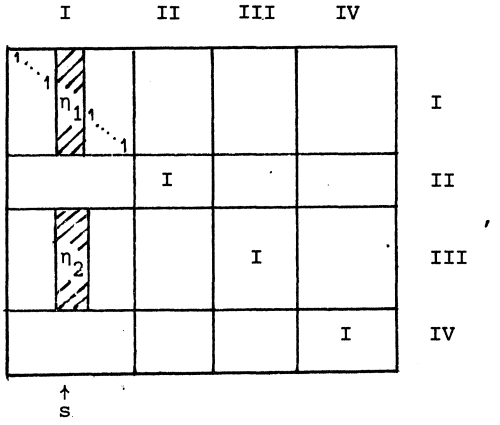
PROOF. if. If the entering process is incident to only one transportation tree, no refining or blending process is labeled in the above described labeling procedure. According to Theorem 3.4.3, the number of basic refining and blending processes with a nonzero coefficient in the representation vector is zero.

only if. Suppose the entering process is incident to at least two transportation trees. Then obviously there must be at least one basic refining or blending process labeled after the first pass through step 3 of the labeling procedure. As noted before all entries in vector $\bar{y}$ which correspond to these labeled processes are nonzero. In other words: $\bar{y}_3 \neq 0$. Then relation 3.4.7 implies that also $y_3 \neq 0$, or: there is at least one basic refining or blending process which has a nonzero coefficient in vector y. □

3.4.2. Determining the process which leaves the basis

The process which leaves the basis is determined by means of the standard minimal ratio test (see Section 2.3).
Let s be the pivot row. The elementary matrix E, required to update the basis inverse, can be calculated in the way described in Section 2.3.
Note that E must have one of the following structures:

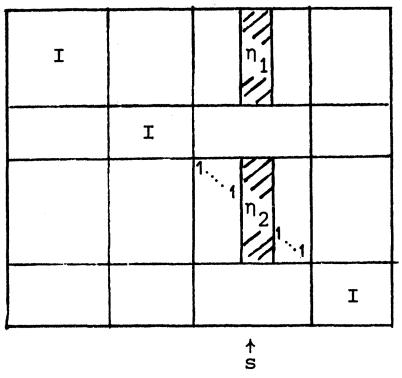
3.4.14. $E =$



,

in case a transportation process leaves the basis, and

3.4.15. $E =$



,

in case a refining or blending process leaves the basis.

3.4.3. Basis change

The leaving, say the s th, column of B is replaced by the entering column. The main questions of this subsection are: what is the inverse of the working basis in the new situation and how can we determine a representative spanning tree for the new situation.
Let B = TP denote the basis before the change, where P is given in 3.4.5 with $Q_4 = 0$ and $R_4 = 0$. Write P^{-1} as:

$$3.4.16. \quad P^{-1} = \begin{bmatrix} I & & S_1 & S_2 \\ & I & & S_3 \\ & & R_1^{-1} & R_0 \\ & & & R_3^{-1} \end{bmatrix},$$

where R_0 is given by 3.4.13 and S_1, S_2 and S_3 by:

$$3.4.17. \quad \begin{bmatrix} S_1 & S_2 \\ & S_3 \end{bmatrix} := - \begin{bmatrix} Q_1 & Q_2 \\ & Q_3 \end{bmatrix} \begin{bmatrix} R_1^{-1} & R_0 \\ & R_3^{-1} \end{bmatrix}.$$

Let $\hat{B}^{-1}$ denote the basis inverse after the change. Then (see Section 2.3):

$$3.4.18. \quad \hat{B}^{-1} = EB^{-1} = EP^{-1}T^{-1}.$$

We want to write $\hat{B}$ as:

$$3.4.19. \quad \hat{B} = \hat{T}\hat{P},$$

where $\hat{T}$ describes a rooted representative spanning tree, such that column $\hat{T}_{\cdot j}$ of $\hat{T}$ corresponds to column $\hat{B}_{\cdot j}$ of $\hat{B}$ $(j = 1,\ldots,m)$.
A representative spanning tree for the new situation can be obtained by updating the previous representative spanning tree, using the labels attached to the arcs of the previous representative spanning tree.

Reestablishing a representative spanning tree

The entering process is either a transportation process (i^*,j^*) or a refining or blending process i^*.
Let the label attached to the leaving process be given by $[i_1,j_1]$.
Put k = 1.

1. If $|i_k| = i^*$, then
   if i^* is a refining node make (i_k, j_k) the representative arc of process i^*,
   if i^* is a blending node make $(j_k, -i_k)$ the representative arc of process i^*.
   Stop: there is a representative spanning tree for the new situation.
   Otherwise, inspect the representative arc of process $|i_k|$. Let this arc have label $[i_{k+1}, j_{k+1}]$.
2. If $i_k > 0$ make (i_k, j_k) the representative arc of refining process i_k.
   Otherwise, make $(j_k, -i_k)$ the representative arc of blending process $|i_k|$.
   Put $k = k + 1$ and goto 1.

EXAMPLE 3.4.5. Suppose that in Figure 3.4.1 arc (4,7) leaves the basis graph. The representative arc of process 8 becomes (7,8), that of process 3: (3,6). See Figure 3.4.2.

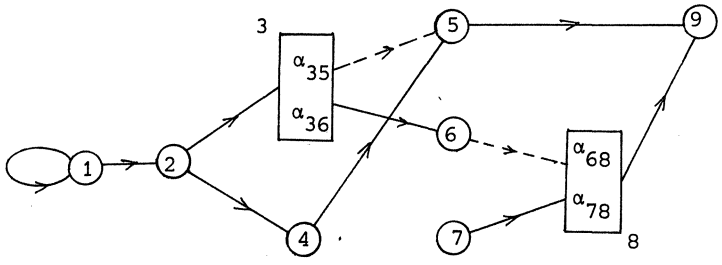


Figure 3.4.2. The reestablished part of the representative spanning tree.

Considering a single basis change, the labeling procedure of Subsection 3.3.1 and the reestablishing procedure of this subsection were developed in such a way, that the number of changes in the previous representative spanning tree, required to obtain a representative spanning tree for the new situation, would be as small as possible.

THEOREM 3.4.6. *The number of changes of representative arcs of the basic refining and blending processes, required to obtain a new representative spanning tree from the previous one, is minimal when the above described labeling and reestablishing procedures are used.*

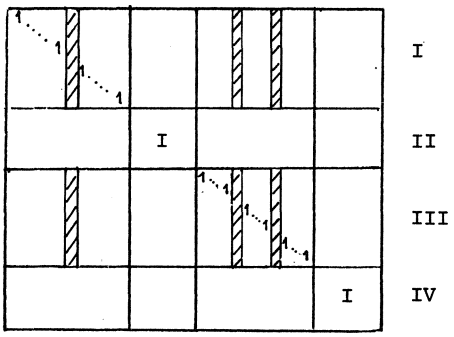
PROOF. We consider a single basis change. Regard the labeling procedure of Subsection 3.4.1. All processes which are labeled after the first pass through step 3 of the labeling procedure are said to have distance 0 (to the entering process). All processes which are labeled after the k th ($k \geq 2$) pass through step 3 of the labeling procedure, but not labeled after the $(k-1)$ th pass, are said to have distance $k-1$.
The reestablishing procedure is such that, if the leaving process has distance ℓ ($\ell = 0,1,\ldots$), exactly ℓ representative arcs of basic refining and blending processes are chosen different from the old situation.
In order to prove the theorem it is sufficient to show that at least ℓ replacements are required to accomplish a new representative spanning tree from the old one ($\ell \geq 1$).
Consider the old representative spanning tree. Add the (c.q. an arbitrary representative) arc associated with the entering process. Leave out the (representative) arc associated with the leaving process.
This graph clearly contains a cycle in which all labeled arcs have distance 0.
In order to achieve a representative spanning tree in the new situation obviously a representative arc, currently contained in this cycle, has to be chosen in a different way. In doing this, a new cycle arises in which all labeled arcs have distance 0 or 1. Again this cycle must be broken, i.e., one of the representative arcs in this cycle must be chosen in a different way, leading to a new cycle in which all labeled arcs have distance 0, 1 or 2.
By repeating this process it is seen that at least ℓ replacements are required, which completes the remaining part of this proof. □

Now that we have a new representative spanning tree, $\hat{T}$ is 3.4.19 is known and $\hat{P}^{-1}$ can be evaluated from (see 3.4.18 and 3.4.19):

3.4.20. $\quad \hat{P}^{-1} = (EP^{-1})(T^{-1}\hat{T})$.

It is assumed that the s th column of matrix $\hat{T}$ corresponds to the (representative) arc of the entering process.
Matrix $(T^{-1}\hat{T})$ has a simple structure. Using the same partitioning as for P in 3.4.5 the product $T^{-1}\hat{T}$ can be written as:

3.4.21. $T^{-1}\hat{T} =$



where the shaded columns denote cycle vectors (see Section 2.4).
In column set I there is one cycle vector in position s if a transportation process leaves the basis. There is no cycle vector in this set in case a refining or blending process leaves the basis.
Column set III has zero or more cycle vectors in case a transportation process leaves the basis and one or more if a refining or blending process leaves the basis.
$T^{-1}\hat{T}$ has at least one cycle vector (in position s); if there is more than one cycle vector this is caused by the changed representative arcs.

REMARK 3.4.7. The s th row of $T^{-1}\hat{T}$ is a unit vector (or a negative unit vector). This fact follows immediately from the way labeling and reestablishing of a representative spanning tree is performed. □

For ease of notation denote $T^{-1}\hat{T}$ in 3.4.21 by:

3.4.22. $$T^{-1}\hat{T} = \begin{bmatrix} T_1 & & T_2 & \\ & I & & \\ T_3 & & T_4 & \\ & & & I \end{bmatrix}.$$

Note that post-multiplying EP^{-1} by $T^{-1}\hat{T}$ modifies only a few columns of EP^{-1}. The modified columns can be obtained by addition and subtraction of columns of EP^{-1} since $T^{-1}\hat{T}$ is a matrix exclusively consisting of elements equal to 0 or ± 1.
Using 3.4.14-3.4.17 and 3.4.22 the expression for $\hat{P}^{-1}$ in 3.4.20 can be evaluated.

Four cases can be distinguished:

1. A refining or blending process enters the basis, a refining or blending process leaves the basis.
2. A transportation process enters the basis, a refining or blending process leaves the basis.
3. A transportation process enters the basis, a transportation process leaves the basis.
4. A refining or blending process enters the basis, a transportation process leaves the basis.

Cases 1 and 2. In these cases a refining or blending process leaves the basis. The matrix product in expression 3.4.20 becomes:

$$3.4.23.\quad \hat{P}^{-1} = \begin{bmatrix} I & & E_1 & \\ & I & & \\ & & E_2 & \\ & & & I \end{bmatrix} \begin{bmatrix} I & & S_1 & S_2 \\ & I & & S_3 \\ & & R_1^{-1} & R_0 \\ & & & R_3^{-1} \end{bmatrix} \begin{bmatrix} I & & T_2 & \\ & I & & \\ & & T_4 & \\ & & & I \end{bmatrix},$$

where the first matrix after the equality sign denotes matrix E given in 3.4.15.

Consequently:

$$3.4.24.\quad \hat{P}^{-1} = \begin{bmatrix} I & & T_2 + (S_1 + E_1 R_1^{-1}) T_4 & S_2 + E_1 R_0 \\ & I & & S_3 \\ & & E_2 R_1^{-1} T_4 & E_2 R_0 \\ & & & R_3^{-1} \end{bmatrix}.$$

In case 1 the dimension of the working basis does not change and the new working basis inverse $\hat{R}^{-1}$ can be written as:

$$3.4.25.\quad \hat{R}^{-1} = \begin{bmatrix} \hat{R}_1^{-1} & \hat{R}_0 \\ & \hat{R}_3^{-1} \end{bmatrix},$$

with

3.4.26. $\hat{R}_1^{-1} = E_2 R_1^{-1} T_4$

3.4.27. $\hat{R}_0 = E_2 R_0$

3.4.28. $\hat{R}_3^{-1} = R_3^{-1}$.

In case 2 the dimension of the working basis is reduced by one, because the s th column of $\hat{P}^{-1}$ is the s th unit vector (cf. 3.3.11 and 3.3.12). By dropping the s - (m-q) th column and row of $\hat{R}^{-1}$ in 3.4.25 the new working basis inverse has been obtained.

Cases 3 and 4. In these cases a transportation process leaves the basis. Expression 3.4.20 becomes:

3.4.29. $$\hat{P}^{-1} = \begin{bmatrix} E_1 & & & \\ & I & & \\ E_2 & & I & \\ & & & I \end{bmatrix} \begin{bmatrix} I & & S_1 & S_2 \\ & I & & S_3 \\ & & R_1^{-1} & R_0 \\ & & & R_3^{-1} \end{bmatrix} \begin{bmatrix} T_1 & & T_2 & \\ & I & & \\ T_3 & & T_4 & \\ & & & I \end{bmatrix},$$

where the first matrix after the equality sign denotes matrix E in 3.4.14. This leads to:

3.4.30. $$\hat{P}^{-1} = \begin{bmatrix} E_1T_1 + E_1S_1T_3 & & E_1T_2 + E_1S_1T_4 & E_1S_2 \\ & I & & S_3 \\ E_2T_1 + (E_2S_1 + R_1^{-1})T_3 & & E_2T_2 + (E_2S_1 + R_1^{-1})T_4 & E_2S_2 + R_0 \\ & & & R_3^{-1} \end{bmatrix}.$$

In case 3 the s th column of $\hat{P}^{-1}$ is again the s th unit vector. The new working basis inverse becomes:

3.4.31. $$\hat{R}^{-1} = \begin{bmatrix} \hat{R}_1^{-1} & \hat{R}_0 \\ & \hat{R}_3^{-1} \end{bmatrix},$$

with

3.4.32. $\hat{R}_1^{-1} = E_2T_2 + (E_2S_1 + R_1^{-1})T_4 = E_2T_2 + (I - E_2Q_1)R_1^{-1}T_4$

3.4.33. $\hat{R}_0 = E_2S_2 + R_0 = (I - E_2Q_1)R_0 - E_2Q_2R_3^{-1}$

3.4.34. $\hat{R}_3^{-1} = R_3^{-1}$.

In case 4 the dimension of the working basis increases by one. The new working basis inverse is given by:

3.4.35. $$\begin{bmatrix} p^*_{ss} & p^*_{s\cdot} \\ p^*_{\cdot s} & \hat{R}^{-1} \end{bmatrix},$$

where $\hat{R}^{-1}$ is given by 3.4.31,
p^*_{ss} is the element in the s th column and s-th row of $\hat{P}^{-1}$ in 3.4.30,
$p^*_{s\cdot}$ is the part in column regions III and IV of the s th row of $\hat{P}^{-1}$,
$p^*_{\cdot s}$ is the part in row regions III and IV of the s th column of $\hat{P}^{-1}$.

Observe from 3.4.30 that the part of vector $p^*_{\cdot s}$ in row region IV is a zero vector.
From the expressions in 3.4.30, p^*_{ss}, $p^*_{s\cdot}$ and $p^*_{\cdot s}$ can be determined, taking advantage of the facts that

- T_1, T_2, T_3 and T_4 contain only elements 0 and ± 1.
- T_2 is possibly a zero matrix and T_4 a unit matrix (this is the case if no representative arcs are replaced by others in the reestablishing procedure).
- The s th row of $[T_1 \quad T_2]$ is a (negative) unit vector.

REMARK 3.4.8. In cases 1 and 2 we do not need matrix Q in 3.3.12 in order to determine the new working basis inverse.
It can easily be verified that in cases 3 and 4 we only need the s th row of Q in 3.3.12. As noted before, this can be done by means of pure-network techniques.
A special way to find the s th row of Q (s = 2,...,m-q), or in other words the elements p_{sj}, j = m-q+1,...,m, of P (see 3.3.12) is the following.
Suppose the j th column of P describes the refining process $k = i_j$. Then $P_{\cdot j}$ can be written as (see 3.3.16):

$$P_{\cdot j} = \alpha_{kr} T^{-1} a^*_{\cdot kr} + \sum_{\ell \in A(k) \setminus \{r\}} \alpha_{k\ell} T^{-1} a^*_{\cdot k\ell}$$

where $a^*_{\cdot k\ell}$ is the vector representation of arc (k,ℓ) as in pure networks (see Section 2.4).
We see that the s th element of $P_{\cdot j}$ can be considered as a linear combination of the s th elements of the cycle vectors $T^{-1} a^*_{\cdot k\ell}$, $\ell \in A(k) \setminus \{r\}$.

Suppose that arc (i_1,j_1) corresponds to the s th column of T (the matrix representation of the rooted representative spanning tree).
An arc (k,ℓ), $\ell \in A(k) \setminus \{r\}$, induces the cycle $C_{k\ell}$ in the representative spanning tree. From the theory of pure networks we know that the s th element of $T^{-1} a^*_{\cdot k\ell}$ is

+1 if arc (i_1,j_1) is a backward arc in $C_{k\ell}$,
-1 if arc (i_1,j_1) is a forward arc in $C_{k\ell}$,
0 otherwise.

We can determine whether arc (i_1,j_1) is a forward or backward arc in $C_{k\ell}$ in the following way:
Leave out the arc (i_1,j_1) from the representative spanning tree.
Then two trees arise, say T_1 with $i_1 \in T_1$ and T_2 with $j_1 \in T_2$. Determine to which of these two trees each node $i \in N$ belongs.
The reader may verify that the following is true:
if $k \in T_1$ and $\ell \in T_2$ then (i_1,j_1) is a backward arc in $C_{k\ell}$,
if $k \in T_2$ and $\ell \in T_1$ then (i_1,j_1) is a forward arc in $C_{k\ell}$,
otherwise (i_1,j_1) is not contained in $C_{k\ell}$.
Based on these observations we state an algorithmic way to determine element p_{sj}, where column $P_{\cdot j}$ corresponds to refining process $k = i_j$. Put $p_{sj} = 0$.

If $k \in T_1$
then do for all nodes $v \in A(k) \setminus \{r\}$
    if $v \in T_2$ then $p_{sj} = p_{sj} + \alpha_{kv}$
If $k \in T_2$
then do for all nodes $v \in A(k) \setminus \{r\}$
    if $v \in T_1$ then $p_{sj} = p_{sj} - \alpha_{kv}$

In an completely analogous way p_{sj} can be determined if $k = i_j$ denotes a blending process. □

The above discussion reveals several important aspects:
- It is sufficient to maintain a working basis of the size equal to the number of basic refining and blending processes.
- After each basis change the working basis inverse has a block triangular form with two blocks on the main diagonal.
- Both S_3 and R_3^{-1} remain what they are in performing the basis change (see 3.4.24 and 3.4.30). This fact will be exploited in determining the Simplex multipliers.

3.4.4. Finding the Simplex multipliers

Assume we have a basis B, which can be written as in 3.3.11:

3.4.36. $B = TP$,

with P^{-1} as in 3.4.16:

3.4.37.
$$P^{-1} = \begin{array}{c} \begin{matrix} \text{I} & \text{II} & \text{III} & \text{IV} \end{matrix} \\ \begin{bmatrix} I & & S_1 & S_2 \\ & I & & S_3 \\ & & R_1^{-1} & R_0 \\ & & & R_3^{-1} \end{bmatrix} \end{array} \begin{matrix} \text{I} \\ \text{II} \\ \text{III} \\ \text{IV} \end{matrix}$$

The Simplex multipliers can be determined from (cf. 2.3.12):

3.4.38. $\pi' B = c_B'$.

Define

3.4.39. $\theta' := \pi' T$,

then, according to 3.4.36 and 3.4.38, θ can be found from:

3.4.40. $\theta' = c_B' P^{-1}$.

After a partitioning of θ and c_B, compatible with the one of P^{-1} in 3.4.37, 3.4.40 can be written as

3.4.41.
$$[\theta_1' \ \theta_2' \ \theta_3' \ \theta_4'] = [c_1' \ c_2' \ c_3' \ c_4'] \begin{bmatrix} I & & S_1 & S_2 \\ & I & & S_3 \\ & & R_1^{-1} & R_0 \\ & & & R_3^{-1} \end{bmatrix}$$

which reduces to:

3.4.42. $\theta_1' = c_1'$

3.4.43. $\theta_2' = c_2'$

3.4.44. $\theta_3' = c_1' S_1 + c_3' R_1^{-1}$

3.4.45. $\theta_4' = c_1' S_2 + c_2' S_3 + c_3' R_0 + c_4' R_3^{-1}$.

Using 3.4.17 θ_3 and θ_4 can also be written as:

3.4.46. $\theta_3' = (c_3' - c_1'Q_1)R_1^{-1}$

and

3.4.47. $\theta_4' = (c_3' - c_1'Q_1)R_0 + (c_4' - c_1'Q_2 - c_2'Q_3)R_3^{-1}$.

The matrices Q_1, Q_2 and Q_3 can be found using pure-network techniques (Section 2.4).
After determination of θ, using 3.4.42, 3.4.43, 3.4.46 and 3.4.47, π in 3.4.39 can also be evaluated by means of pure-network techniques.

We conclude this subsection by pointing out that, after each basis change, θ_4 in 3.3.45 can be found in an alternative way.
Consider the basis $\hat{B} = \hat{T}\hat{P}$ after the basis change and assume $\hat{P}^{-1}$ is partitioned in the way obtained in the previous section.
We must solve:

3.4.48. $\hat{\pi}'\hat{B} = \hat{c}_B'$

or in the same way as before, we first determine

3.4.49. $\hat{\theta}' = \hat{c}_B' \hat{P}^{-1}$

and secondly solve

3.4.50. $\hat{\pi}'\hat{T} = \hat{\theta}$.

Let $\hat{\theta}$ and $\hat{c}_B$ be partitioned compatible with $\hat{P}^{-1}$.
Consider the two possible cases:

(a) A refining or blending process has left the basis. Then $\hat{c}_1 = c_1$, $\hat{c}_2 = c_2$, $\hat{c}_3$ differs in one element from c_3, $\hat{c}_4 = c_4$.
According to 3.4.24, $\hat{\theta}_4$ can be written as:

3.4.51. $\hat{\theta}_4' = \hat{c}_1'(S_2 + E_1R_0) + \hat{c}_2'S_3 + \hat{c}_3'E_2R_0 + \hat{c}_4'R_3^{-1} =$

$= c_1'S_2 + c_1'E_1R_0 + c_2'S_3 + \hat{c}_3'E_2R_0 + c_4'R_3^{-1}$.

Subtraction of 3.4.51 and 3.4.45 gives:

3.4.52. $\hat{\theta}_4' - \theta_4' = (c_1'E_1 + \hat{c}_3'E_2 - c_3')R_0$.

Considering the structure of E_1, E_2, $\hat{c}_3$ and c_3 the right-hand side of 3.4.52 denotes the product of a scalar with the s - (m-q) th row of R_0.

(b) A transportation process has left the basis. Then $\hat{c}_1$ differs in one element from c_1, $\hat{c}_2 = c_2$, $\hat{c}_3 = c_3$, and $\hat{c}_4 = c_4$.
According to 3.4.30, $\hat{\theta}_4$ can be written as:

3.4.53. $$\hat{\theta}_4 = \hat{c}_1(E_1S_2) + \hat{c}_2S_3 + \hat{c}_3(E_2S_2 + R_0) + \hat{c}_4 R_3^{-1} =$$
$$= \hat{c}_1 E_1 S_2 + c_2S_3 + c_3 E_2 S_2 + c_3 R_0 + c_4 R_3^{-1} .$$

Subtraction of 3.4.53 and 3.4.45 gives:

3.4.54. $$\hat{\theta}_4 - \theta_4 = (\hat{c}_1E_1 - c_1 + c_3E_2)S_2 = \text{scalar} \times \text{the s th row of } S_2 .$$

In order to determine the s-th row of S_2 we see from the fact that (formula 3.4.17):

3.4.55. $$S_2 = - Q_1 R_0 - Q_2 R_3^{-1} ,$$

we only need the s th row of Q.

The above discussion makes clear that $\hat{\theta}_4$ can be determined in the following way:
Given π determine θ_4 from 3.4.39 - another possibility is of course to store $[\theta_3 \quad \theta_4]$ in every iteration - and $\hat{\theta}_4$ is found from 3.4.52 or 3.4.54. Note that the only parts of Q, required to determine the Simplex multipliers, are Q_1 in order to evaluate θ_3 in 3.4.46 and the s th row of Q in case a transportation process has left the basis in the basis change.

3.4.5. Calculating the reduced costs

Assuming that the current Simplex multipliers are denoted by π, the reduced costs can be found from:

3.4.56. $$\bar{c}_{ij} = - \pi_i + \pi_j - c_{ij} \qquad (i,j) \in TP$$

3.4.57. $$\bar{c}_i = - \pi_i + \sum_{j \in A(i)} \alpha_{ij} \pi_j - c_i \qquad i \in RP$$

3.4.58. $$\bar{c}_i = \pi_i - \sum_{j \in B(i)} \alpha_{ji} \pi_j - c_i \qquad i \in BP .$$

Using the standard rules of the Simplex algorithm (Section 2.3), it is determined whether the current solution is optimal. If not, a nonbasic process is selected to enter the basis.

3.4.6. Initialization

An easy way of finding a starting basis for pure network problems is described in Section 2.4. This starting procedure can also be applied to problem 3.2.14-3.2.16. The starting basis is then simply a rooted spanning tree with transportation arcs only. Matrix P in 3.3.12 is the identity matrix and the working basis has size zero.

3.5. *Another view on pure processing networks*

In the previous section a specification of the primal Simplex algorithm has been developed, in which the basis structure, in particular the representative spanning tree, is exploited. For each process $i \in BAP$ - the set of basic refining and blending processes - a representative arc was chosen from PA(i), in such a way that the arcs in the transportation trees plus these representative arcs form the arc set of a spanning tree in G(N,A).
In this section we discuss an alternative way to regard and solve pure processing network problems. Instead of choosing a representative arc for each process $i \in BAP$ we here discuss the possibility to choose a representative node for each process $i \in BAP$ in a special way, namely, we can select a node from each set N(i), $i \in BAP$, in such a way that these nodes belong to different transportation trees. This way of looking at pure processing networks gives rise to several modifications in the Simplex algorithm of Section 3.4. These modifications will be discussed.

Assume again that a basis B is given by:

3.5.1. $\quad B = [-e_{i_0} \quad B_T \quad B_P]$,

where B_T is an $m \times (m-q-1)$ matrix denoting the structural basic transportation processes, and B_P is an $m \times q$ matrix representing the basic refining and blending processes.
Suppose that the slack column $-e_{i_0}$ in 3.5.1 has its nonzero entry in a row which corresponds to a node in transportation tree T_1.

Lemma 3.3.5 says that every subset S_p of the basic refining and blending processes is incident to at least $|S_p|$ transportation trees from the set $\{T_2,\ldots,T_{q+1}\}$. Recall that this is essentially HALL's condition (see Section 2.5). Considering the definition of $T(S_p)$ in the previous section, HALL's theorem (Theorem 2.5.8) implies the following lemma.

LEMMA 3.5.1. *For each process* i ∈ BAP *a node can be chosen from the set* N(i) *in such a way that these nodes belong to different transportation trees from the set* $\{T_2,\ldots,T_{q+1}\}$.

Suppose that for each process i ∈ BAP a representative node is chosen from N(i) in the way of Lemma 3.5.1. Attach a self-loop to each of these nodes. These self-loops can be considered as the root-arcs of the transportation trees $T_2,\ldots,T_{q+1}$. Then the self-loop (i_0,i_0), the basic transportation arcs associated with B_T in 3.5.1, and the self-loops attached to the representative nodes of the basic refining and blending processes form the arc set of a "representative spanning forest of rooted transportation trees" (abbreviated to representative forest in the sequel).
In matrix terms the self-loops are represented by negative unit columns.
It is quite clear that we can use a representative forest instead of a representative spanning tree in the Simplex PRON procedure of the previous section.

EXAMPLE 3.5.2. In the example of Figure 3.3.1 nodes 4 and 5 can be thought to represent the processes 2 and 3, respectively. The corresponding representative forest is drawn in Figure 3.5.1.

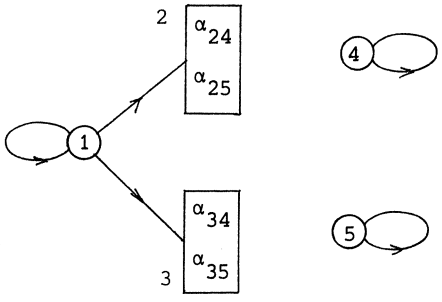


Figure 3.5.1. A representative forest for the basis graph in Figure 3.3.1.

Again the representative forest is not necessarily unique.
Let T be the matrix representation of the representative forest such that each column $T_{\cdot j}$ of T corresponds to column $B_{\cdot j}$ of B in 3.5.1, j = 1,...,m.
Obviously T is a square nonsingular matrix with the last q columns a set of negative unit vectors.
As in Section 3.3 we can write

3.5.2. $B = TP$,

with

3.5.3. $$P = \begin{bmatrix} I & Q \\ & R \end{bmatrix} ,$$

where
I is the identity matrix of order (m - q),
Q is an (m-q) × q matrix, and
R is a square nonsingular matrix of order q.
For matrix B in 3.3.10 formula 3.5.2 specifies to:

3.5.4.
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 11 & 12 & 13 & 2 & 4 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & & \\ & 1 & & -1 & \\ & & 1 & & -1 \\ & & & \alpha_{24} & \alpha_{34} \\ & & & \alpha_{25} & \alpha_{35} \end{bmatrix} \end{array} =$$

$$= \begin{array}{c} \begin{array}{ccccc} 11 & 12 & 13 & 44 & 55 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & & \\ & 1 & & & \\ & & 1 & & \\ & & & -1 & \\ & & & & -1 \end{bmatrix} \end{array} \begin{bmatrix} 1 & & & 1 & 1 \\ & 1 & & -1 & \\ & & 1 & & -1 \\ & & & -\alpha_{24} & -\alpha_{34} \\ & & & -\alpha_{25} & -\alpha_{35} \end{bmatrix} .$$

REMARK 3.5.3. Consider a column $P_{\cdot j}$ (m-q < j ≤ m) of P and column $B_{\cdot j}$ of B which corresponds to process $k = i_j$. According to 3.5.2:

3.5.5. $P_{\cdot j} = T^{-1} B_{\cdot j}$.

If $B_{\cdot j}$ denotes the refining process $k = i_j$ we can write:

3.5.6. $$B_{\cdot j} = - e_k + \sum_{\ell \in A(k)} \alpha_{k\ell} e_\ell \ .$$

Consequently, using 3.5.5 and 3.5.6,

3.5.7. $$P_{\cdot j} = - T^{-1} e_k + \sum_{\ell \in A(k)} \alpha_{k\ell} T^{-1} e_\ell \ .$$

Similarly, if $B_{\cdot j}$ denotes the blending process $k = i_j$, column $P_{\cdot j}$ can be written as:

3.5.8. $$P_{\cdot j} = T^{-1} e_k - \sum_{\ell \in B(k)} \alpha_{\ell k} T^{-1} e_\ell \ .$$

The vectors $T^{-1} e_k$ and $T^{-1} e_\ell$ in 3.5.7 and 3.5.8 describe either a negative unit vector or a more general root-path vector (see Section 2.4). Consequently, each column $P_{\cdot j}$ $(m-q < j \leq m)$ can be considered as a linear combination of the j th unit vector (which results from the representative node of process i_j) and the root-path vectors which describe the path from a nonrepresentative node $\ell \in N(i_j)$ to the root of the transportation tree to which node ℓ belongs.
In the example presented, column $P_{\cdot 4}$ can be written as:

$$P_{\cdot 4} = - T^{-1} e_2 + \alpha_{24} T^{-1} e_4 + \alpha_{25} T^{-1} e_5 =$$

$$= - \begin{bmatrix} -1 \\ 1 \\ \\ \end{bmatrix} + \alpha_{24} \begin{bmatrix} \\ -1 \\ \\ \end{bmatrix} + \alpha_{25} \begin{bmatrix} \\ \\ -1 \end{bmatrix} .$$

This observation will again be used in the Simplex algorithm. □

REMARK 3.5.4. If the first column of B and T is $-e_1$ $(i_0 = 1)$, then one easily proves that the first row of Q in 3.5.3 is <u>not</u> a row of zeros (cf. Remark 3.3.9). □

As in Section 3.3 we can introduce an aggregated graph and prove several important properties of matrix R in 3.5.3.

ASSUMPTION 3.5.5. *The transportation trees ($\neq T_1$) are numbered in such a way that the representative node of process i_j (associated with column $B_{\cdot j}$ in B, j = m-q+1,...,m) belongs to transportation tree $T_{j-(m-q-1)}$.*

The aggregated graph

The *aggregated graph*, associated with basis B in 3.5.1, is the directed graph $G(N^*,A^*)$, with
$N^* = \{2,...,q+1\}$; node i corresponds to transportation tree T_i, and
A^* as follows. The self-loops (i,i), i = 2,...,q+1, belong to A^*. Furthermore, if the nonrepresentative nodes of process i_j belong to transportation trees $T_{j_1},...,T_{j_s}$ ($\neq T_1$), then also the arcs $(k,j_1),...,(k,j_s)$ with $k = j - (m-q-1)$ belong to A^*.
This statement holds for all processes i_j, j = m-q+1,...,m.

Similarly as in Section 3.4 we introduce the m × q matrix $V = [v_{ij}]$ (note that in this section also the transportation tree T_1 has been taken into account), with

3.5.9. $v_{ij} = 1$ if node i belongs to transportation tree T_{j+1}, (i = 1,...,m; j = 1,...,q),
$= 0$ otherwise.

Suppose B is written as in 3.5.1, then the product V'B satisfies:

3.5.10. $V'B = [0 \quad 0 \quad R^*]$,
($\leftarrow 1 \rightarrow \leftarrow m-q-1 \rightarrow \leftarrow q \rightarrow$)

where $R^* = [r^*_{ij}]$ is a q × q matrix.
Note that R^* in 3.5.10 is identical to the last q rows of R^* in 3.3.21 if in Section 3.3 the transportation trees are numbered in the same way as indicated in Assumption 3.5.5. Consequently, we can immediately state the following three facts for matrix R^* in 3.5.10 (cf. the properties of matrix R^* in Section 3.3):

1. R^* is unique, given the basis B (considering Assumption 3.5.5 no row permutations are possible).
2. An element r^*_{ij} of R^* is unequal to zero iff the process associated with the j-th column of R^* is incident to transportation tree T_{i+1}. Hence, each column of R contains as many elements unequal to zero as the number

of transportation trees in the set $\{T_2,\ldots,T_{q+1}\}$ to which the associated refining or blending process is incident.

3. The main diagonal of R^* is zero-free. This follows immediately from point 2 and Assumption 3.5.5.

If we define the matrix $R^{**} = [r^{**}_{ij}]$ by:

3.5.11. $r^{**}_{ij} = 1$, if $r^{*}_{ij} \neq 0$,

$= 0$, if $r_{ij} = 0$,

then one can easily observe from the definition of the aggregated graph, that matrix R^{**} is the adjacency matrix of the aggregated graph.
For simplicity we will say that R^* describes the adjacency structure of the aggregated graph.

Properties of matrix R

THEOREM 3.5.6. *For matrices* R *in* 3.5.3 *and* R^* *in* 3.5.10 *the following relation holds:*

3.5.12. $R = -R^*$.

PROOF. We can write the product V'T as:

3.5.13. $V'T = [0 \;\; 0 \;\; -I]$

(see the definition of V in 3.5.9 and Assumption 3.5.5).
Since P is given as in 3.5.3 we also have:

3.5.14. $V'TP = [0 \;\; 0 \;\; -R]$.

Considering 3.5.2, 3.5.10 and 3.5.14:

$R = -R^*$. □

Hence the above discussion on the properties of matrix R^* makes clear that the following theorems are valid.

THEOREM 3.5.7. *Matrix* R *in* 3.5.3 *is unique.*

THEOREM 3.5.8. *An element* r_{ij} *of* R *is unequal to zero iff the proces associated with the* j*-th column of* R *is incident to transportation tree* T_{i+1}.

This theorem implies that matrix R is at least as sparse as matrix B_p in 3.5.1 in the following sense: each column of R does not contain more elements unequal to zero than its corresponding column in B_p.

In the next section we discuss the possibility to permute matrix R to a block triangular matrix with irreducible blocks on the main diagonal. In this respect the following theorems are important:

THEOREM 3.5.9. *The main diagonal of* R *is zero-free.*

THEOREM 3.5.10. *Matrix* R *describes the adjacency structure of the aggregated graph.*

Thus far we have discussed that we can describe the basis structure in a pure processing network in a somewhat different way than in Section 3.3. The present specification of the basis structure gives rise to a specification of the primal Simplex algorithm, which is different from the one in Section 3.4 in several aspects. What has been said in Section 3.4 remains valid except for the modifications discussed next.

Modifications of the Simplex algorithm in Section 3.4

A. Instead of a representative spanning tree a representative forest is kept stored in some convenient way.

B. In finding the representation y of the entering column a in terms of the basis B (Subsection 3.4.1), the labeling procedure becomes slightly different.
   Let the set of arcs on the path from a node $j \in N$ to the root of the transportation tree to which node j belongs, plus the self-loop attached to this root, be denoted by P_j.
   Now a basic refining or blending process i is considered labeled whenever the self-loop attached to the representative node of i has a label.
   A basic refining or blending process i is considered scanned if all arcs on the paths P_j, $j \in N(i)$, are labeled.

Labeling procedure

1. If the entering process is a transportation process, say (i^*,j^*), then label all arcs in P_{i^*} by $[i^*,i^*]$ and label all arcs in P_{j^*} by $[i^*,j^*]$, provided they do not have yet a label.

   If the entering process is a refining or blending process, say i^*, then determine the paths P_j for all $j \in N(i^*)$, one by one. Stop tracing a path as soon as a labeled arc is encountered. Label the arcs in P_j by $[i^*,j]$.

   If the entering process is incident to only one transportation tree, then stop.

2. List all basic refining and blending processes which are labeled, but not scanned.

   If this list is empty, then stop.

   Otherwise, let W denote the set of all nonrepresentative nodes of the labeled, but not scanned, refining and blending processes.

3. Determine the paths P_j for all nodes $j \in W$, one by one. Stop tracing a path as soon as a labeled arc is encountered. Whenever node j belongs to N(i), label the arcs in P_j by $[i,j]$.

   Continue with step 2.

EXAMPLE 3.5.11. Figure 3.5.2 shows the labeled part of a representative forest, assuming that transportation process (4,5) enters the basis. The situation in Figure 3.5.2 corresponds to the one in Figure 3.4.1. It is assumed that node 5 is the representative node of process 3, node 6 is the representative node of process 8.

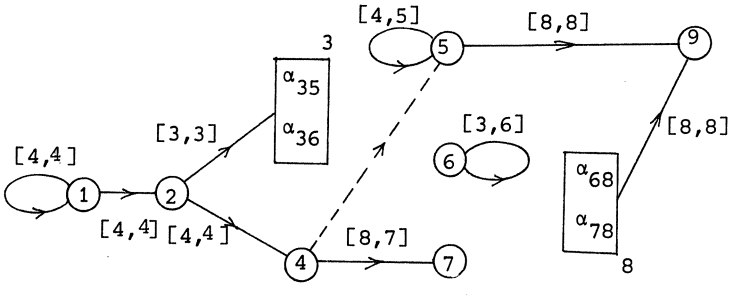


Figure 3.5.2. The labeled part of a representative forest in pure processing networks.

REMARK 3.5.12. Note that possibly some more arcs are labeled than in the labeling procedure of Subsection 3.4.1. To be precise: let the set of labeled processes which would arise in the labeling procedure of Subsection 3.4.1 be given by L.
If the labeled processes in BAP contain some process i for which some node $j \in N(i)$ belongs to T_1, then the labeled part of the representative forest consists of L plus all arcs on the path from the root of T_1 to L (including the root-arc of T_1). Compare in this respect Figure 3.4.1 and Figure 3.5.2.
If the entering process is incident to only one transportation tree, say T_ℓ, then the labeled part of the representative forest consists of L plus all arcs on the path from the root of T_ℓ to L (including the root-arc of T_ℓ).
In both cases we may just as well consider the arcs on such paths as not being labeled (cf. Subsection 3.4.1 and Section 2.4, Figure 2.4.2). □

Theorems 3.4.3 and 3.4.4 also hold in the present view on pure processing networks.

C. In the basis change (cf. Subsection 3.4.3) a representative forest is reestablished as follows.

Reestablishing a representative forest

Let the label attached to the leaving process be given by $[i_1,j_1]$. The entering process is either a transportation process, say (i^*,j^*), or a refining or blending process, say i^*. Put k = 1.

1. If $i_k = i^*$ then
   if i^* is a refining or blending process, take j_k as the representative node of process i^*.
   Stop. There is a representative forest for the new situation.
   Otherwise ($i_k \neq i^*$), let the self-loop attached to the representative node of process i_k have label $[i_{k+1},j_{k+1}]$.
   Make j_k the representative node of process i_k.
   Put k = k + 1 and goto 1.

EXAMPLE 3.5.13. Suppose that in Figure 3.5.2 arc (4,7) leaves the basis graph. The representative node of process 8 becomes node 7, that of process 3: node 6. See Figure 3.5.3.

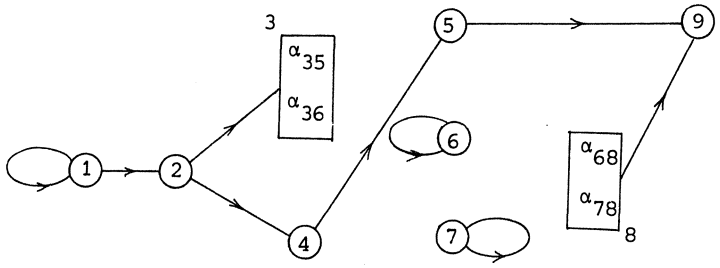


Figure 3.5.3. The reestablished part of the representative forest.

Theorem 3.4.6 holds in the present view on pure processing networks too. We note that the shaded columns in the matrix product $T^{-1}\hat{T}$ in 3.4.21 now denote root-path vectors. The reader may verify that matrices T_3 and T_4 in 3.4.22 in general have a somewhat easier shape than in the solution procedure of Section 3.4. For instance, if a transportation process leaves the basis, T_4 is a permutation matrix.
Hence, the expressions in which T_3 and T_4 appear (formulae 3.4.26, 3.4.30 and 3.4.32) can usually be evaluated in a somewhat easier way than in the solution procedure of Section 3.4.
The statement in Remark 3.4.8 on determining the s th row of Q in 3.3.12 can easily be adapted.
A comparison of the Simplex PRON procedures of Sections 3.4 and 3.5 will be given in the next section.

3.6. Remarks

In this section some remarks are made with respect to pure processing network problems.

Implementation considerations

The development of efficient implementations of the Simplex PRON procedures, described in the previous two sections, is a field of future study. Here we only want to point out several aspects that may be important in this respect.

In the solution procedure of Section 3.4 (PRON 1) both the rooted representative spanning tree and matrix R^{-1} play an important role.
The rooted representative spanning tree, which has the matrix representation T in 3.3.11, is used in:

1. Solving equations of the form

   $$Tx = b^*,$$

   as in 3.4.3, 3.4.10, or of the form

   $$\pi'T = \theta' ,$$

   as in 3.4.39.
2. Determining the processes which take part in the representation of the entering process in terms of the basis (Subsection 3.4.1). The labeling procedure described in that section can be seen as tracing a number of cycles induced in this tree.
3. Finding a representative spanning tree for the situation after a basis change. This process can be regarded as making and breaking cycles a number of times consecutively.
4. Updating the working basis inverse (see Subsection 3.4.3).

Such operations also arise in solving pure network flow problems (Section 2.4) or LP-problems with an embedded pure network structure, e.g. GLOVER & KLINGMAN [1981]. Therefore the techniques developed for those problems can be applied here. Relevant references are given in Subsection 1.1.1.
Matrix R^{-1} can be stored explicitly, but, if its size is large, a product form or elimination form would be more appropriate. A product form can be developed by the same kind of reasoning as in HELGASON & KENNINGTON [1977]. GLOVER & KLINGMAN [1981] use a product form of the working basis inverse in their Simplex SON procedure.
About the solution procedure of Section 3.5 (PRON 2) similar things could be said.
Both in PRON 1 and in PRON 2 an important question is how to choose the process which enters the basis (i.e., selecting the pivot column).
It is worthwhile to test whether (and if so, how) priorities should be given to the following four possible cases (cf. Theorem 3.4.4).

1. A transportation process, incident to exactly one transportation tree enters the basis.
   Characteristics:
   - the representation vector y in 3.4.1 is as in the pure network situation;
   - R^{-1} does not change, as can be verified by inspection of 3.4.30, considering the fact that $E_2 = 0$, $T_2 = 0$ and $T_4 = I$.
2. A transportation process, incident to two transportation trees enters the basis.
   Characteristics:
   - at least one basic refining or blending process has a nonzero coefficient in the representation vector y;
   - R^{-1} changes. The size of R^{-1} reduces by one or remains the same.
3. A refining or blending process, incident to exactly one transportation tree enters the basis.
   Characteristics:
   - the representation vector y can be found by pure-network techniques (Section 2.4);
   - the size of R^{-1} increases by one. R^{-1} becomes as in 3.4.35 with $\hat{R}^{-1}$ unchanged and $p_{\cdot s} = 0$. This follows from 3.4.30, considering that $E_2 = 0$, $T_2 = 0$, $T_3 = 0$ and $T_4 = I$.
4. A refining or blending process, incident to at least two transportation trees enters the basis.
   Characteristics:
   - at least one basic refining or blending process has a nonzero coefficient in the representation vector y;
   - R^{-1} changes. The size of R^{-1} remains the same or increases by one.

   By recording in each iteration to which transportation tree each node in the network belongs, the number of transportation trees, to which a certain process is incident, can easily be determined.

A comparison of PRON 1 and PRON 2

Although PRON 2 (Section 3.5) is perhaps less intuitive than PRON 1 (Section 3.4), PRON 2 may be preferred because of several reasons:

1. Matrix R, the working basis, is unique in PRON 2 (Theorem 3.5.7), independent of the specific choice of the representative nodes.

In PRON 1, matrix R depends on the specific choice of the representative arcs of the basic refining and blending processes (see 3.3.25).

2. In PRON 2 matrix R is at least as sparse as matrix B_p in 3.5.1 in the sense that the number of nonzero elements in each column of R is not greater than the number of nonzeros in the corresponding column in B_p. In PRON 1 this need not be (and in many instances is not) the case.
3. The ideas of PRON 2 are easier generalized to generalized processing network problems than those of PRON 1 (see Section 4.3).
4. An other advantage of PRON 2 over PRON 1 is explained in the subsequent discussion on block triangularization of matrix R.

In our opinion the pure processing network structure is exploited as far as possible in the Simplex PRON procedures of Sections 3.4 and 3.5.
However, in many applications the working basis R will be a sparse matrix and the question arises whether it is possible to reorder R in some desirable form using sparsity considerations. Some of these forms are discussed in DUFF [1977a].
In the sequel the possibility to block triangularize R (and consequently R^{-1}) further than the block triangular form with two blocks on the main diagonal, obtained from the labeling procedure in Subsection 3.4.1, is pointed out.

Block triangularization of the working basis R

Consider an arbitrary square nonsingular matrix A. The essential question in block triangularizing A is to find permutation matrices P_1 and P_2 such that:

3.6.1. $$P_1 A P_2 = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ & A_{22} & \cdots & A_{2N} \\ & & \ddots & \vdots \\ & & & A_{NN} \end{bmatrix}$$

where A_{ii} $(i = 1,\ldots,N)$ are square irreducible matrices.
Usually P_1 and P_2 are determined in two stages (see DUFF [1977a]):

(1) Determine a row permutation matrix P_3 such that $A_1 := P_3 A$ is a matrix with a zero-free diagonal.

(2) Find a permutation matrix P_4 such that $P_4 A_1 P_4'$ has the desired form of 3.6.1, Such a permutation is called a symmetric permutation.

After performing these two steps, we have for P_1 and P_2 in 3.6.1: $P_1 = P_4 P_3$ and $P_2 = P_4'$.

The problem under (1) is known under several names, a.o. "finding a maximal transversal" or "finding a set of distinct representatives" (HALL [1935]). GUSTAVSON [1976] and DUFF [1981] present algorithms, based on HALL's ideas, which require $O(n\tau)$ computations in the worst case, where n is the order of matrix A and τ the number of nonzeros in A.

Well-known algorithms to solve a problem as under (2) are those of SARGENT & WESTENBERG [1964] and TARJAN [1972] (see also GUSTAVSON [1976], DUFF & REID [1978a, 1978b]). TARJAN's algorithm appears to be efficient in practice and also has the lowest computational complexity of all algorithms known to solve problems as under (2), namely $O(n + \tau)$.

TARJAN's algorithm is based on the following ideas: Associate with matrix $A = [a_{ij}]$ a directed graph. Each row i of A corresponds to a node i in this graph. Arc (i,j) is present in this graph iff $a_{ij} \neq 0$. So matrix A is essentially the adjacency matrix of this graph.

Using depth-first search, the so-called strong components (see DUFF & REID [1978a]) of this graph are detected, which correspond to the irreducible blocks A_{ii} (i = 1,...,N) in 3.6.1.

This two-stage approach is justified by the fact that the obtained block triangular form $P_4 P_3 A P_4'$ is unique in the sense that the number of blocks and the rows and columns lying in each block is fixed, independent of the particular choice for P_3. This is proved in HOWELL [1976] and DUFF [1977b].

If a matrix A has one or more zero elements on the main diagonal, it may very well be that there exists no symmetric permutation which leads to a block triangular form of A with irreducible blocks on the main diagonal (see HOWELL's example [1976]).

These statements give new insight in the Simplex PRON procedures discussed in Sections 3.4 and 3.5. By choosing the representative arcs (Section 3.4) or the representative nodes (Section 3.5) in a special way, it has been accomplished that the main diagonal of the working basis R is always zero-free (see Theorems 3.3.10 and 3.5.9). Conclusion: in block triangularizing R only the second stage is required. Applying TARJAN's algorithm to matrix R implies that the graph, of which R describes the adjacency structure, is available. In PRON 2 this is implicitly the case if for each basic refining

and blending process i it is known to which transportation trees the nodes in N(i) belong. Recall that R is in fact the adjacency matrix of the aggregated graph (Theorem 3.5.10). In PRON 1 the structure of R is not directly available (the elements in each column of R are found by tracing the nonrepresentative cycles of the corresponding process). So here is another argument in favor of PRON 2.

It is suggested to block triangularize R in every iteration of the Simplex algorithm. Since the nonlabeled part (R_3^{-1} in 3.4.24 or 3.4.30) remains unchanged in the basis change, only the labeled part (R_1^{-1}) has to be updated.

The advantages of a block triangularized version of R^{-1} are obvious: reduction of storage requirements and computation time (see for further discussion DUFF [1977a]).

Is it necessary to use a working basis (inverse)?

In the algorithms of Sections 3.4 and 3.5 it is assumed that a working basis inverse R^{-1} is used. Is it necessary to do this or is it also possible to work without R^{-1}? This question is inspired by the situation in pure and generalized networks where all the information required is obtained by manipulating on the basis graph (in generalized networks some algebraic work has to be done but this only comes up to solving a number of single equations with one unknown). The main advantage of such an approach is that it is possible to work with the original data, thus reducing (cumulative) round-off errors and storage requirements.
The answer to the question posed primarily depends on whether a basic system as $Bx = b$ or $\pi'B = c_B'$ can be solved, using the structure of the basis graph in such a way that it does not require too much work. The hard part in solving $Bx = b$ is, considering the analysis in Sections 3.3 and 3,5, ultimately: solve a system $Rx = b^*$ (where R is given in 3.3.12 or 3.5.3). Although we tried to work out several intuitive ideas no satisfactory results were obtained. Considering the facts known about matrix R and after reading Chapter 5 this should not cause too much astonishment.

Degeneracy

Degenerate steps in the Simplex algorithm are likely to occur frequently and theoretically the possibility of cycling exists. Of course techniques known for general LP problems (perturbation, lexicographic ordering, BLAND's [1977] rule) can be applied to prevent cycling. An interesting subject for further study is to investigate whether finite modifications can be developed using similar ideas as in CUNNINGHAM [1976, 1979], ELAM, GLOVER & KLINGMAN [1979] and ADOLPHSON [1980].

4. GENERALIZED PROCESSING NETWORKS

4.1. Introduction

In the previous chapter networks have been considered in which flow is conserved. However, in practice, there are many situations in which flow is not conserved due to leakage, damage, conversion losses, growth, etc. Sometimes is is natural to say that the decrease or increase of flow takes place on an arc, sometimes it is more appropriate to state that the decrease or increase takes place in a node. For the purpose of describing a mathematical framework for such networks it is sufficient to regard only one of these possibilities. Here we have chosen for a description in which flow is conserved in nodes, but possibly not on arcs. In the literature the same approach is usually followed, simply because in general it gives rise to more compact formulations than in the case where flow is not necessarily conserved in nodes.

The concept of a "generalized processing network" has already been introduced in Section 3.2.
Again it is assumed that the special topological properties, mentioned in Remark 3.2.2, hold.
The main intent of this chapter is to show that the ideas of Chapter 3 can easily be generalized to generalized processing network problems.
Proofs of lemmas and theorems are omitted since they are either completely analogous to the ones of corresponding statements in Chapter 3 or simple to provide.

4.2. *Mathematical formulation*

As in Section 3.2, we present two distinct LP-formulations of the minimal cost flow problem in a generalized processing network.
Consider a directed and connected graph G(N,A) with node set N, containing m nodes, and arc set A, consisting of n arcs. Self-loops are allowed to be present.
Suppose that with each arc $(i,j) \in A$ a multiplier g_{ij} is associated. The meaning of this multiplier is the same as in generalized networks, described in Section 2.5.
Using the notation, definitions and assumptions of Sections 2.5 and 3.2, the LP-formulation of the minimal cost flow problem in a generalized processing network is

Formulation I

4.2.1. $$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij} ,$$

4.2.2. $$-\sum_{j} x_{ij} + \sum_{j} g_{ji} x_{ji} = b_i \qquad i \in N ,$$

4.2.3. $$\frac{\alpha_{ij}}{\alpha_{ir}} x_{ir} - x_{ij} = 0 \qquad i \in RN,\ r \in A(i),\ j \in A(i) \setminus \{r\} ,$$

4.2.4. $$\frac{\alpha_{ji}}{\alpha_{ri}} x_{ri} - x_{ji} = 0 \qquad i \in BN,\ r \in B(i),\ j \in B(i) \setminus \{r\} ,$$

4.2.5. $$0 \le x_{ij} \le u_{ij} \qquad (i,j) \in A .$$

Observe that this formulation is only slightly different from 3.2.9-3.2.13. It reflects the fact that the minimal cost flow problem can be considered as a generalized network flow problem (4.2.1, 4.2.2 and 4.2.5) with side constraints 4.2.3 and 4.2.4. If equations 4.2.2 are linearly dependent, the problem can be reduced to a pure processing network problem by means of scaling. This is immediately clear from the way TRUEMPER's [1977] scaling procedure performs. For this reason we take the following assumption.

ASSUMPTION 4.2.1. *The equations in* 4.2.2 *are linearly independent.*

Problem 4.2.1-4.2.5 can be solved by an algorithm of HULTZ & KLINGMAN [1976]. Then, under the given assumptions, a working basis of fixed size: $\Sigma_{i \in PN} (n_i - 1)$, i.e., the number of constraints in 4.2.3 and 4.2.4, would be required.

However, the solution procedure developed in Section 4.4 (based on the subsequently discussed formulation II) uses a working basis of variable size q, with $0 \leq q \leq \Sigma_{i \in PN} 1 \ (= |PN|)$, which is usually much smaller than $\Sigma_{i \in PN} (n_i - 1)$.

Formulation II

After substitution of the expressions for x_{ij} in 4.2.3 and x_{ji} in 4.2.4 into 4.2.2 and a suitable scaling of the columns a compact formulation results (cf. formulation II in Section 3.2):

4.2.6. minimize $c'x$

4.2.7. $Ax = b$

4.2.8. $0 \leq x \leq u$,

with A an $m \times n$ matrix, $b \in \mathbb{R}^m$ and $c,x,u \in \mathbb{R}^n$.
Each row i of A is associated with node $i \in N$.
Each column of A describes one of the three types of processes:

(a) refining process i. The elements in column $a_{\cdot i}$ are:

-1	in row i,
$\alpha_{ij} g_{ij}$	in row j, $j \in A(i)$,
0	otherwise.

(b) blending process i. The elements in column $a_{\cdot i}$ are:

+ 1	in row i,
$-\alpha_{ji} g_{ji}$	in row j, $j \in B(i)$,
0	otherwise.

(c) transportation process (i,j). The elements in column $a_{\cdot ij}$ are

-1	in row i,
g_{ij}	in row j,
0	otherwise.

The variable associated with a column $a_{\cdot i}$, corresponding to a refining or blending process i, describes the total throughput of process i.
Formulation 4.2.6-4.2.8 is one of a generalized network flow problem with side activities. For such problems no special (network oriented) algorithms are known.
The dual problem of 4.2.6-4.2.8 is given by:

4.2.9. $\quad$ maximize $b'\pi - u'\nu$

4.2.10. $\quad - \pi_i + g_{ij}\pi_j - \nu_{ij} \leq c_{ij} \qquad (i,j) \in TP$,

4.2.11. $\quad - \pi_i + \sum_{j \in A(i)} \alpha_{ij} g_{ij} \pi_j - \nu_i \leq c_i \qquad i \in RP$,

4.2.12. $\quad \pi_i - \sum_{j \in B(i)} \alpha_{ji} g_{ji} \pi_j - \nu_i \leq c_i \qquad i \in BP$,

4.2.13. $\quad \nu \geq 0$,

where TP denotes the set of transportation processes, RP represents the set of refining processes and BP the set of blending processes.
If column $a_{\cdot i}$ describes a refining process i, $a_{\cdot i}$ can also be written as (cf. 3.2.21):

4.2.14. $\quad a_{\cdot i} = \sum_{j \in A(i)} \alpha_{ij} a^*_{ij}$,

where a^*_{ij} denotes the vector representation of arc (i,j) with multiplier g_{ij} (see Section 2.5). Formula 4.2.14 makes clear that the set of processing arcs PA(i) can be associated with refining process i. A similar statement as in 4.2.14 can be made for a blending process i.
Note that if all multipliers g_{ij} are positive, $(i,j) \in A$, matrix $\hat{A}$ in 4.2.7 satisfies the following property (cf. Remark 3.2.4):
if there is more than one negative (positive) element in a column of $\hat{A}$, then there is only one positive (negative) element.

REMARK 4.2.2. If we would have taken the constraint

4.2.15. $\quad - \sum_{j \in A(i)} h_{ij} x_{ij} - \sum_{j \in B(i)} g_{ji} x_{ji} = b_i \qquad i \in N$

instead of 4.2.2, then problem 4.2.1-4.2.5 could still be regarded as a generalized processing network problem (cf. Remark 2.5.2) and the contents of this chapter also holds for such problems after a few obvious adaptations.

Then the columns of $\bar{A}$ in 4.2.7 would have the following shape:

(a) column $a_{\cdot i}$, representing refining process i:

$-\sum_{j\in A(i)} \alpha_{ij} h_{ij}$ in row i,

$\alpha_{ij} g_{ij}$ in row j, $j \in A(i)$,

0 otherwise.

(b) column $a_{\cdot i}$, representing blending process i:

$\sum_{j\in B(i)} \alpha_{ji} h_{ji}$ in row i,

$-\alpha_{ji} g_{ji}$ in row j, $j \in B(i)$

0 otherwise.

(c) column $a_{\cdot ij}$, representing transportation process (i,j):

$-h_{ij}$ in row i,

g_{ij} in row j,

0 otherwise.

Note that if h_{ij} and g_{ij}, $(i,j) \in A$, are allowed to be arbitrary real numbers, formulation 4.2.6-4.2.8 is in fact one of a general LP-problem. This aspect is discussed further in Chapter 5. □

Formulation II will be used for the Simplex PRON procedure of Section 4.4. In the next section the basis structure in a generalized processing network problem is explained.

4.3. Basis structure

It is not restrictive to take the following assumption.

ASSUMPTION 4.3.1. *The rank of* $\bar{A}$ *in* 4.2.7 *equals* m.

Let B denote a basis of $\bar{A}$, partitioned as:

4.3.1. $B = [B_T \quad B_P]$,

where

B_T is an m × (m-q) matrix denoting the (generalized) transportation processes, and
B_P is an m × q matrix representing the basic refining and blending processes ($0 \leq q \leq m$).

Let the set of basic refining and blending processes again be denoted by BAP. BAP contains q elements. The basis graph associated with B is defined as the directed graph with node set N and as arc set: all transportation arcs associated with the columns in B_T in 4.3.1, and all processing arcs associated with the columns in B_P in 4.3.1, i.e., all arcs in PA(i), $i \in$ BAP (cf. 4.2.14).

Consider the graph which arises if in the basis graph all processing arcs are left out. Let this graph be denoted by G(N,BT).

LEMMA 4.3.2. *Each connected component of* G(N,BT) *contains at most one cycle.*

A connected component of G(N,BT) which contains no cycle is again called a *transportation tree.*
If a connected component of G(N,BT) contains a cycle (possibly a self-loop) it is called a *transportation quasi-tree.*
So Lemma 4.3.2 states that G(N,BT) consists of a number of transportation trees and a number of transportation quasi-trees.

LEMMA 4.3.3. *A basis graph contains* q *transportation trees iff the number of basic refining and blending processes equals* q.

Observe that Lemma 2.5.6 is valid because B denotes a square nonsingular matrix (see Remark 2.5.7). It is possible to state a lemma, closely related to Lemma 2.5.6.
Suppose BAP $\neq \emptyset$.
Let S_P be a nonempty subset of BAP.
Furthermore, let $T(S_P)$ denote the set of transportation trees which are incident to the processes $i \in S_P$.

LEMMA 4.3.4. *Any nonempty subset* S_P *of basic refining and blending processes is incident to at least* $|S_P|$ *transportation trees:*

4.3.2. $\quad |T(S_P)| \geq |S_P|$.

Using this lemma the following lemma can be proved:

LEMMA 4.3.5. *The representative arcs of the basic refining and blending processes can be chosen in such a way that the basic transportation arcs plus these representative arcs form the arc set of a spanning forest of quasi-trees in* G(N,A).

Such a forest is called a *representative forest.*

The four stated lemmas prove:

THEOREM 4.3.6. *A basis graph in a generalized processing network* G(N,A) *consists of*
a forest of quasi-trees formed by the basic transportation arcs and the properly chosen representative arcs of the basic refining and blending processes, and
all nonrepresentative arcs of the basic refining and blending processes.

The structure of a basis graph is illustrated in the following example.

EXAMPLE 4.3.7. Consider the basis B:

4.3.3.

$$B = \begin{array}{c} \begin{matrix} 12 & 13 & 15 & 2 & 3 \end{matrix} \\ \begin{bmatrix} -1 & -1 & -1 & & \\ 2 & & & -1 & \\ & 1 & & & -1 \\ & & & \alpha_{24} & \alpha_{34} \\ & & 1 & \alpha_{25} & \alpha_{35} \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

The associated basis graph is drawn in Figure 4.3.1, where the representative forest is indicated by heavy lines.

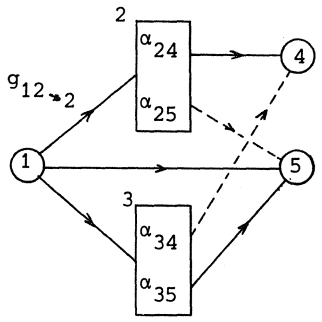


Figure 4.3.1. An example of a basis graph in a generalized processing network.

Let T be the matrix representation of the representative forest with the convention that each (representative) arc (i,j) has multipliers g_{ij} as previously defined. This convention is plausible if it is tried to set up a solution procedure in the same spirit as in Section 3.4 (PRON 1). However, now there is no guarantee that T is nonsingular. For the example presented T would be:

$$
\begin{array}{cc}
 & \begin{array}{ccccc} 12 & 13 & 15 & 24 & 35 \end{array} \\
\begin{array}{c} \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix} -1 & -1 & -1 & & \\ & & & -1 & \\ & 1 & & & -1 \\ & & & 1 & \\ & & 1 & & 1 \end{bmatrix}
\end{array}
\qquad 4.3.4.
$$

which is seen to be singular since the cycle factor of the cycle formed by the arcs (1,3), (3,5) and (1,5) is equal to one (see Section 2.5). It is still an open question whether there exists, for every basis B, a particular choice of the representative arcs such that T would be nonsingular. In any case it is clear that a labeling procedure and a reestablishing procedure as in Section 3.4 alone might not be sufficient.
Considering this observation an approach as proposed in Section 3.5 would be more appealing.

Suppose that for each process $i \in BAP$ a node j is chosen from N(i), which belongs to some transportation tree. This is possible because of Lemma 4.3.4. Such a node j is called the *representative node* of process i.

Furthermore, suppose that BAP $\neq \emptyset$. The condition mentioned in Lemma 4.3.4 together with HALL's theorem (Theorem 2.5.8) implies the following lemma:

LEMMA 4.3.8. *For each process* i $\in$ BAP *a node can be chosen from the set* N(i) *in such a way that these nodes belong to different transportation trees.*

Attach, as in the procedure of Section 3.5, a self-loop to each of these representative nodes. In matrix terms such a self-loop is again represented by a negative unit column (it has multiplier 1). Now a basis is represented by a number of transportation quasi-trees and a number of rooted transportation trees. The collection of these quasi-trees and rooted trees is again called the representative forest.

EXAMPLE 4.3.9. In the example of Figure 4.3.1 nodes 4 and 5 can be thought to represent the sets N(3) and N(2), respectively. The corresponding representative forest is drawn in Figure 4.3.2.

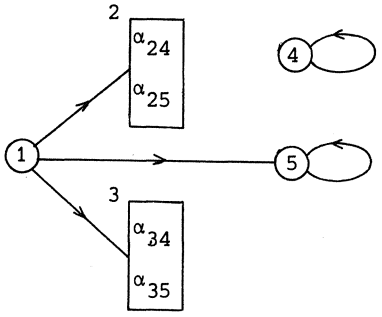


Figure 4.3.2. A representative forest for the basis graph in Figure 4.3.1.

Let T be the matrix representation of the representative forest. The sequence of the columns in T corresponds to the sequence of columns in B. Then B can be written as

4.3.5. $\quad B = TP$

with

4.3.6. $\quad P = \begin{bmatrix} I & Q \\ & R \end{bmatrix}$,

where

I is the identity matrix of order (m - q),
Q is some (m-q) × q matrix, and
R is a square nonsingular matrix of order q.
For matrix B in 4.3.3 formula 4.3.5 specializes to:

4.3.7.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 12 & 13 & 15 & 2 & 3 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & & \\ 2 & & & -1 & \\ & 1 & & & -1 \\ & & & \alpha_{24} & \alpha_{34} \\ & & 1 & \alpha_{25} & \alpha_{35} \end{bmatrix} \end{array} =$$

$$= \begin{array}{c} \begin{array}{ccccc} 12 & 13 & 15 & 55 & 44 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & & \\ 2 & & & & \\ & 1 & & & \\ & & & & -1 \\ & & 1 & -1 & \end{bmatrix} \end{array} \begin{bmatrix} 1 & & & -\frac{1}{2} & \\ & 1 & & & -1 \\ & & 1 & \frac{1}{2} & 1 \\ & & & -\alpha_{25}+\frac{1}{2} & -\alpha_{35}+1 \\ & & & -\alpha_{24} & -\alpha_{34} \end{bmatrix} .$$

REMARK 4.3.10. Consider a column $P_{\cdot j}$ $(m-q < j \leq m)$ of P and a column $B_{\cdot j}$ of B which corresponds to process $k = i_j$. Suppose process k is a refining process. Since $B_{\cdot j}$ can be written as:

4.3.8. $$B_{\cdot j} = - e_k + \sum_{\ell \in A(k)} \alpha_{k\ell}\, g_{k\ell}\, e_\ell$$

relation 4.3.5 makes clear that $P_{\cdot j}$ can be written as:

4.3.9. $$P_{\cdot j} = - T^{-1} e_k + \sum_{\ell \in A(k)} \alpha_{k\ell}\, g_{k\ell}\, T^{-1} e_\ell \ .$$

Similarly, if process $k = i_j$ is a blending process, $P_{\cdot j}$ can be written as:

4.3.10. $$P_{\cdot j} = T^{-1} e_k - \sum_{\ell \in B(k)} \alpha_{\ell k}\, g_{\ell k}\, T^{-1} e_\ell \ .$$

The vectors $T^{-1} e_\ell$ $(\ell \in N(i_j))$ in 4.3.9 or 4.3.10 are the cycle-path vectors of the nodes $\ell \in N(i_j)$ in the representative forest (see Section 2.5). Formulae 4.3.9 and 4.3.10 express the fact that each column $P_{\cdot j}$ can be considered as a linear combination of the j th negative unit vector, which

results from the representative node of process i_j, and the cycle-path vectors, originated by the nonrepresentative nodes of $N(i_j)$. This observation is used in the Simplex algorithm of Section 4.4.

We can define an aggregated graph associated with basis B in almost the same way as in Section 3.5. The following assumption is similar to Assumption 3.5.5.

ASSUMPTION 4.3.11. *The transportation trees are numbered in such a way that the representative node of process* i_j *(associated with column* $B_{\cdot j}$ *in* B, $j = m-q+1,\ldots,m$*) belongs to transportation tree* $T_{j-(m-q)}$.

The aggregated graph

The aggregated graph, associated with basis B in 4.3.1, is the directed graph $G(N^*,A^*)$ with $N^* = \{1,\ldots,q\}$ in which node i corresponds to transportation tree T_i, and A^* as follows. The self-loops (i,i), $i = 1,\ldots,q$, belong to A^*. Furthermore, if the nonrepresentative nodes of process i_j, which are not contained in some transportation quasi-tree, belong to transportation trees $T_{j_1},\ldots,T_{j_s}$, then also the arcs $(k,j_1),\ldots,(k,j_s)$ with $k = j - (m-q)$ belong to A^*. This statement holds for all processes i_j, $j = m-q+1,\ldots,m$.

EXAMPLE 4.3.12. In Figure 4.3.1 node 5 is the representative node of process 2, node 4 the representative node of process 3. Transportation tree T_1 has node set {1,2,3,5} and T_2 has node set {4}. The aggregated graph, associated with B in 4.3.3, is drawn in Figure 4.3.3.

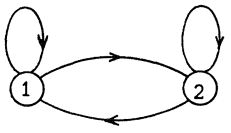


Figure 4.3.3. The aggregated graph associated with B in 4.3.3.

Properties of matrix R

With respect to matrix R in 4.3.6 the following can be said (cf. Theorems 3.5.7-3.5.10 in the pure processing network situation).

THEOREM 4.3.13. *If* B *in* 4.3.5 *denotes a basis in a generalized processing network and* R *is given in* 4.3.6, *matrix* R *is unique.*

THEOREM 4.3.14. *Each column of matrix* R *contains at most as many elements unequal to zero as the number of transportation trees to which its corresponding process is incident.*

The main diagonal of R is not necessarily zero-free, as can be seen from the example presented, whenever $\alpha_{24} = \alpha_{25} = \frac{1}{2}$ (see formula 4.3.7).

REMARK 4.3.15. In Section 3.5 matrix R describes the adjacency structure of the there defined aggregated graph (Theorem 3.5.10). We note that this statement no longer holds for generalized processing networks.
Observe from 4.3.9 and 4.3.10 that not only the cycle-path vectors and the processing coefficients α_{ij} influence the structure of R (this is the case in Section 3.5), but also the multipliers g_{ij}. These multipliers may cause some element of matrix R to be zero although there may be an arc in the aggregated graph which corresponds to this element. We will call this phenomenon "multiplier degeneracy".
In the example presented $r_{11} = 0$ if $\alpha_{24} = \alpha_{25} = \frac{1}{2}$, although there is an arc in the aggregated graph which corresponds to element r_{11}, namely the self-loop (1,1), see Figure 4.3.3.
We will say that matrix R describes the adjacency structure of the aggregated graph, except for "multiplier degeneracy".

4.4. *The Simplex algorithm for the minimal cost flow problem in a generalized processing network*

The minimal cost flow problem in a generalized processing network can be solved in almost the same way as described in Section 3.4, with the adaptations of Section 3.5. Hence, in this section we only point out some important aspects and discuss the differences with the procedures developed in Sections 3.4 and 3.5.
We assume that the representative forest and the inverse of R in 4.3.6 are kept stored in some convenient way.

In every instance where in Section 3.4 or 3.5 pure-network techniques are used these should be replaced by generalized-network techniques (matrix T now describes a forest of quasi-trees and rooted transportation trees as in generalized networks).

In finding the representation y of the entering column a in terms of the basis B (see Subsection 3.4.1), the labeling procedure becomes different from the one in Section 3.5 at two points:

1. The statement after the third if in step 1 is left away, i.e., we do not stop the labeling procedure at that point, even if the entering process is incident to only one transportation tree.
2. If a node $j \in N$ belongs to a transportation quasi-tree, P_j now denotes the set of arcs contained in the cycle of this quasi-tree, plus all arcs on the path from node j to this cycle.

EXAMPLE 4.4.1. Figure 4.4.1 shows the labeled part of a representative forest assuming that transportation process (4,5) enters the basis. It is assumed that node 5 is the representative node of process 3, node 6 is the representative node of process 8. Compare this situation with Figure 3.5.2.

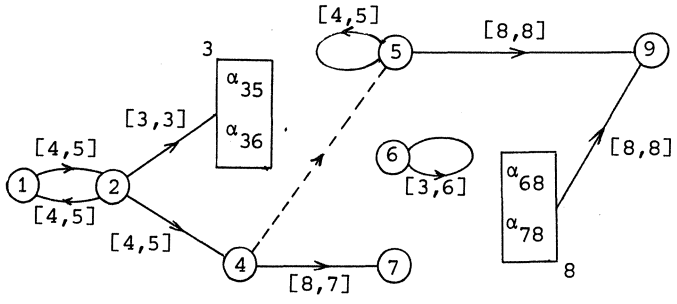


Figure 4.4.1. The labeled part of a representative forest in a generalized processing network.

Theorem 3.4.3 holds for generalized processing networks too.
The following two theorems may be important for implementations of the present Simplex PRON algorithm. Moreover, Theorem 4.4.3 plays a role in the discussion on generalized processing networks with additional linear constraints in Section 6.3.

THEOREM 4.4.2. *The number of basic refining and blending processes with a nonzero coefficient in the representation vector* y *of* a *in terms of the basis* B, *is zero* <u>*if*</u> *the entering process is incident to only* <u>*transportation quasi-trees*</u>.

THEOREM 4.4.3. *If the vector* a *represents a transportation process, say* (i^*,j^*), *then the number of basic refining and blending processes with a nonzero coefficient in the representation vector* y *of* a *in terms of the basis* B, *is zero* <u>*iff*</u> *one of the following two statements hold:*

1. (i^*,j^*) *is incident to only transportation quasi-trees,*
2. (i^*,j^*) *is not a self-loop (i.e.,* $i^* \neq j^*$*),*
   both i^* *and* j^* *belong to one transportation tree, say* T_ℓ, *and the cycle factor of the cycle, induced by* (i^*,j^*) *in* T_ℓ, *equals* 1.

In the <u>basis change</u> (cf. Subsection 3.4.3 and Section 3.5) we can reestablish a representative forest in the same way as in Section 3.5.
Theorem 3.4.6 holds for generalized processing networks too.
We note that the shaded columns in the matrix product $T^{-1}\hat{T}$ in 3.4.21 now denote cycle-path vectors (see Section 2.5).
The s th row of $(T^{-1}\hat{T})$ has only one element unequal to zero (cf. Remark 3.4.7).
Furthermore, the statement in Remark 3.4.8, on determining the s th row of Q in 3.3.12, can easily be generalized.

The <u>reduced costs</u> (cf. Subsection 3.4.5) can be found from:

4.4.1. $$\bar{c}_{ij} = -\pi_i + g_{ij}\pi_j - c_{ij} \qquad (i,j) \in TP ,$$

4.4.2. $$\bar{c}_i = -\pi_i + \sum_{j\in A(i)} \alpha_{ij} g_{ij} \pi_j - c_i \qquad i \in RP ,$$

4.4.3. $$\bar{c}_i = \pi_i - \sum_{j\in B(i)} \alpha_{ji} g_{ji} \pi_j - c_i \qquad i \in BP .$$

A <u>starting basis</u> can be taken in the same way as in Subsection 3.4.6.

4.5. Remarks

Similar remarks as in Section 3.6 can be made on implementation questions. One important difference with pure processing networks is the fact that the main diagonal of the matrix R in 4.3.6 is not necessarily zero-free (see Section 4.3). So perhaps there exists no symmetric permutation of R such that a block triangular form arises with irreducible blocks on the main diagonal. However, it is important to note that the intention of the solution procedure is to exploit the network structure. Particular values of coefficients α_{ij} or g_{ij} have not been considered in the labeling procedure or in the reestablishing procedure. It is noted that this is also commonplace in the primal Simplex solution procedures for generalized networks, described in the literature. Therefore it is not strange to do just the same in block triangularizing the working basis: disregard "multiplier degeneracy" (see Remark 4.3.15) and only use the structure of the basis graph. More precise: use the structure of the aggregated graph, which is implicitly available if it is known to which transportation trees each refining or blending process is incident.

5. PROCESSING NETWORKS AND GENERAL LINEAR PROGRAMMING

5.1. Introduction

On one hand processing network problems are more general than pure or generalized network problems, on the other hand they seem more special than general Linear Programming problems.
In the previous chapters attention has been paid to the relation between processing networks and pure or generalized networks. Here we investigate the relation between processing networks and general LP-problems of the form:

5.1.1. minimize $c'x$

5.1.2. $Ax = b$

5.1.3. $0 \leq x \leq u$,

where A is an $m \times n$ matrix, $b \in \mathbb{R}^m$ and $c,x,u \in \mathbb{R}^n$.
It will appear that a processing network structure is not as special as it seems at first sight.

5.2. Generalized processing networks and general linear programming

In this section we show that an arbitrary LP-problem of the form 5.1.1-5.1.3 can readily be interpreted as a generalized processing network problem in which both positive and negative multipliers may appear. A direct consequence is that, in principle, the solution procedure of Chapter 4 can be applied to general LP-problems, leading to a specification of the primal Simplex algorithm in which the (working) basis is kept stored in block triangular form. The relation between this approach and other sparse matrix primal Simplex procedures proposed in the literature will be discussed.

THEOREM 5.2.1. *The formulation* 5.1.1-5.1.3 *of an arbitrary LP-problem can be considered as the compact formulation (formulation* II *in Section* 4.2*) of a generalized processing network problem in which both positive and negative multipliers may appear.*

PROOF. Consider the LP-problem 5.1.1-5.1.3 and let $a_{\cdot j}$ be the j th column of matrix A in 5.1.2. If column $a_{\cdot j}$ contains at most two nonzero elements it represents a transportation process (see Remarks 2.5.2 and 4.2.2).
If column $a_{\cdot j}$ contains ℓ ($\ell \geq 3$) nonzero elements we can easily associate a refining or blending process with column $a_{\cdot j}$.
Suppose $a_{\cdot j}$ contains a negative element then, after an appropriate positive scaling of $a_{\cdot j}$ and suitable row permutations, $a_{\cdot j}$ can always be written as:

5.2.1. $$a_{\cdot j} = [-1, a_{2j}, \ldots, a_{\ell j}, 0, \ldots, 0]' ,$$

where all $a_{ij} \neq 0$, $i = 2, \ldots, \ell$, and $3 \leq \ell \leq m$.
Alternatively $a_{\cdot j}$ can (for instance) be written as:

5.2.2. $$a_{\cdot j} = \frac{1}{\ell - 1} \begin{bmatrix} -1 \\ (\ell-1)a_{2j} \\ \\ \\ \end{bmatrix} + \frac{1}{\ell - 1} \begin{bmatrix} -1 \\ \\ (\ell-1)a_{3j} \\ \\ \end{bmatrix} + \ldots + \frac{1}{\ell - 1} \begin{bmatrix} -1 \\ \\ \\ \\ (\ell-1)a_{\ell j} \end{bmatrix} .$$

This formulation indicates that we can associate with column $a_{\cdot j}$ a bundle of $(\ell - 1)$ generalized arcs - all incident and directed from one particular node and having either positive or negative multipliers - on which proportionality of flow is required (cf. 4.2.14). Hence, such a column $a_{\cdot j}$ corresponds to a refining process in a generalized processing network.
If column $a_{\cdot j}$ contains no negative element we can associate a blending process with $a_{\cdot j}$ in a similar way.
The conclusion is that we can associate a transportation, refining or blending process with each column $a_{\cdot j}$ of A and the theorem has been proved. □

The relevance of Theorem 5.2.1 is clear. Apparently, with the generalized processing network interpretation in mind, we can in principle apply the solution procedure of Section 4.4 to general LP-problems. We review several important aspects of the Simplex PRON procedure of Section 4.4 in the light

of well-known sparse matrix primal Simplex solution techniques, proposed in the literature:

- The transportation part of a basis B (i.e., those columns of B which have at most two nonzero elements) is included in the representative forest. In each iteration of the Simplex PRON algorithm of Section 4.4 there is an interaction between the representative forest and the working basis inverse R^{-1}.

This idea - extract the transportation part of a basis and do the rest of the work by means of a working basis - also comes up in a large number of algorithms dealing with LP-problems with an embedded pure or generalized network structure, see e.g. HARTMAN & LASDON [1972], HULTZ & KLINGMAN [1976], CHEN & SAIGAL [1977], and GLOVER & KLINGMAN [1981].

- The size of the working basis varies dynamically.

This is, for instance, also the case in the methods proposed in HARTMAN & LASDON [1972] and GLOVER & KLINGMAN [1981].

- A labeling procedure determines which basic processes can take part at a nonzero level in the representation of the entering process in terms of the basis.

Similar labeling procedures are used in the well-known specifications of the primal Simplex algorithm for pure, generalized and multicommodity network flow problems.

- The representative forest plays an essential role in the distinct steps of the Simplex PRON procedure of Section 4.4.

A concept, similar to that of a representative forest, is the so-called master basis tree, introduced by GLOVER & KLINGMAN [1981] in their Simplex SON approach (see also the discussion in Chapter 6).

- The working basis inverse R^{-1} is kept stored in block triangular form.

In the literature many times the suggestion is made to exploit the sparsity of LP-models (and perhaps a natural block structure, BASTIAN [1980]), by using a block triangular form of the basis inverse. Some references are DANTZIG [1955], PHILLIPS [1970], and SAUNDERS [1972].
A difference with the procedures known in the literature is, that in the present Simplex PRON approach the working basis inverse R^{-1}, rather than the whole basis inverse B^{-1}, is kept stored in a block triangular form. Only in case a basis does not contain any transportation column the entire basis inverse B^{-1} is kept stored in block triangular form.

REMARK 5.2.2. A suggestion for future research is to consider factorization of the blocks on the main diagonal of the block triangular working basis. Such a suggestion is made earlier by KEVORKIAN [1979]. All in all it would result in an approach in the same spirit as GRAVES & MC BRIDE [1976] and MC BRIDE [1978]. □

5.3. *"Almost" pure processing networks and general linear programming*

In this section we give an "almost" pure processing network interpretation to general LP-problems of the form 5.1.1-5.1.3, to which the redundant constraint:

5.3.1. $\quad -e'Ax = -e'b$

is added.
Moreover, we will show that the Simplex PRON procedures of Sections 3.4 and 3.5 can easily be adapted to solve LP-problems of the form 5.1.1-5.1.3, 5.3.1, although some of the properties which hold for pure processing networks are no longer valid.

By an "almost" pure processing network problem we mean a network flow problem with the following characteristics:

- conservation of flow, both in nodes and on arcs.
- proportionality of flow in particular subsets of the arc set. In each such a subset the arcs are incident to one common node, but they may be directed both towards and from this common node.
- capacity bounds on arcs.

THEOREM 5.3.1. *The formulation* 5.1.1-5.1.3,5.3.1 *of an arbitrary LP-problem can be considered as a compact formulation (similar to formulation* II *in Section* 3.2*) of an "almost" pure processing network problem.*

PROOF. Consider the LP-problem 5.1.1-5.1.3,5.3.1 and let $a^*_{\cdot j}$ denote the j th column of the coefficient matrix $A^* = \begin{bmatrix} A \\ -e'A \end{bmatrix}$ of this LP-problem.
Clearly column $a^*_{\cdot j}$ has a column sum zero.
If $a^*_{\cdot j}$ contains only one positive element or only one negative element, $a^*_{\cdot j}$ corresponds to one of the three types of processes in a pure processing network (see Remark 3.2.4).

Suppose $a^*_{\cdot j}$ contains two or more positive elements and two or more negative ones, and let us assume that column $a^*_{\cdot j}$ is scaled such that the sum of the positive elements in $a^*_{\cdot j}$ is equal to one. After a suitable interchange of rows, $a_{\cdot j}$ can be written as:

$$a^*_{\cdot j} = [-a^*_{1j},\ldots,-a^*_{kj},a^*_{k+1,j},\ldots,a^*_{\ell j},0,\ldots,0]'$$

where $a^*_{ij} > 0$, $i = 1,\ldots,\ell$, $2 \leq k \leq m-1$; $k+1 \leq \ell \leq m+1$.
Now $a^*_{\cdot j}$ can for instance be written as:

5.3.2.
$$a^*_{\cdot j} = \begin{bmatrix} -a^*_{1j} \\ -a^*_{2j} \\ \vdots \\ -a^*_{kj} \\ a^*_{k+1,j} \\ \vdots \\ a^*_{\ell j} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = a^*_{2j}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \ldots + a^*_{kj}\begin{bmatrix} 1 \\ & -1 \end{bmatrix} + a^*_{k+1,j}\begin{bmatrix} -1 \\ & 1 \end{bmatrix} + \ldots + a^*_{\ell j}\begin{bmatrix} -1 \\ & \\ & \\ & 1 \end{bmatrix}.$$

Formula 5.3.2 shows that we can associate with column $a^*_{\cdot j}$ a bundle of $(\ell - 1)$ arcs - all incident to one particular node, some directed from this node and some directed to this node - on which proportionality of flow is required.
Since the above described interpretation can be given to all columns $a^*_{\cdot j}$ of A^*, the theorem has been proved. □

In Figure 5.3.1 the bundle of arcs associated with the vectors in 5.3.2 is depicted.

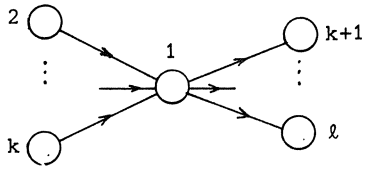


Figure 5.3.1. The bundle of arcs associated with column $a^*_{\cdot j}$ in 5.3.2.

If we reexamine the proofs of lemmas and theorems in Sections 3.3-3.5 we see that, except for Theorems 3.3.10, 3.4.4, 3.5.8 and 3.5.9, the only arguments used are:

1. B denotes a basis.
2. All basic nonslack columns have the zero-sum property.

In the Simplex PRON procedures of Sections 3.4 and 3.5 Theorems 3.3.10, 3.4.4, 3.5.8 and 3.5.9 were not used and hence these procedures can be adapted to solve "almost" pure processing network problems (we say "adapted" because in the discussions in Sections 3.4 and 3.5 we always assumed that all arcs associated with a nontransportation process were either directed to or from a processing node).

For "almost" pure processing network problems Theorems 3.3.10, 3.4.4, 3.5.8 and 3.5.9 are no longer valid.

We note that Theorems 3.3.10, 3.5.8 and 3.5.9 play an essential role in the discussion on block triangularizing the working basis (Section 3.6) and conclude that, if TARJAN's algorithm [1972] is applied to the working basis, we do not necessarily obtain a block triangular form with irreducible blocks on the main diagonal (cf. the situation in generalized processing networks, especially Remark 4.3.15). Theorem 3.4.4 has been used in the discussion on implementation considerations (Section 3.6) and will be used in proving Theorem 6.2.3 in Chapter 6.

5.4. *Transforming general LP-problems to pure processing network problems*

We discuss the possibility to transform a general LP-problem of the form 5.1.1-5.1.3 to a pure processing network problem, at the possible expense of blowing up the size of the problem.

THEOREM 5.4.1. *Any LP-problem of the form* 5.1.1-5.1.3 *can be transformed to an LP-problem associated with a pure processing network.*

PROOF. Consider the general LP-problem 5.1.1-5.1.3 and add the redundant constraint 5.3.1. We then have the LP-problem

5.4.1. minimize $c'x$

5.4.2. $A^*x = b^*$

5.4.3. $0 \leq x \leq u$,

where

$$A^* = \begin{bmatrix} A \\ -e'A \end{bmatrix} \quad \text{and} \quad b^* = \begin{bmatrix} b \\ -e'b \end{bmatrix} .$$

Clearly the column sum of each column in A^* is zero ($e'A^* = 0$).
Scale the columns of A^* in such a way that the sum of the positive elements in each column of A^* is equal to 1. Next partition A^* as:

5.4.4 $\quad A^* = [A^{1*} \quad A^{2*}]$

where A^{1*} consists of all columns in A^* which have exactly one positive element or exactly one negative element. Hence A^{2*} consists of the columns in A^* which have at least two positive elements and at least two negative ones.
Let $[c_1' \quad c_2']$, $[x_1' \quad x_2']$, and $[u_1' \quad u_2']$ be the partitioning of c', x' and u', compatible with the partitioning of A^* in 5.4.4.
Suppose we write:

5.4.5. $\quad A^{2*} = A^{3*} + A^{4*}$,

where $A^{2*} = [a_{ij}^{2*}]$, $A^{3*} = [a_{ij}^{3*}]$ and $A^{4*} = [a_{ij}^{4*}]$, with

$$a_{ij}^{3*} = a_{ij}^{2*} \text{ , if } a_{ij}^{2*} > 0$$
$$= 0 \quad \text{, } \quad a_{ij}^{2*} \leq 0 \text{ ,}$$

$$a_{ij}^{4*} = a_{ij}^{2*} \text{ , if } a_{ij}^{2*} < 0$$
$$= 0 \quad \text{, } \quad a_{ij}^{2*} \geq 0 \text{ .}$$

Now it can easily be observed that the LP-problem:

5.4.6. $\quad$ minimize $c_1' x_1 + c_2' x_2$

5.4.7. $\quad A^{1*}x_1 + A^{3*}x_2 + A^{4*}x_3 = b^*$

5.4.8. $\quad - I x_2 + I x_3 = 0$

5.4.9. $\quad 0 \leq x_1 \leq u_1$

5.4.10. $\quad 0 \leq x_2 \leq u_2$

5.4.11. $\quad 0 \leq x_3 \leq u_2$

is equivalent to 5.1.1-5.1.3. Observe that each column of

5.4.12. $$\begin{bmatrix} A^{1*} & A^{2*} & A^{3*} \\ & -I & I \end{bmatrix}$$

satisfies the two properties, mentioned in Remark 3.2.4, which characterize a pure processing network problem. Hence the theorem has been proved. □

The proof of Theorem 5.4.1 is in fact nothing more than an algebraic way to say that the picture of Figure 5.3.1 changes into that of Figure 5.4.1.

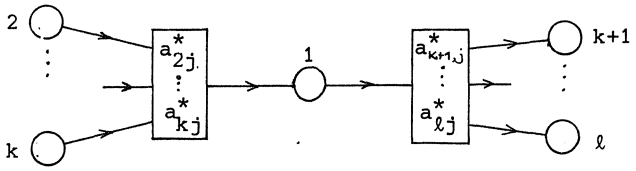


Figure 5.4.1. The transformation of Figure 5.3.1 to a pure processing network form.

Moreover the proof shows that the transformed problem 5.4.6-5.4.11 has a coefficient matrix with $(m+1+n)$ rows and $2n$ columns in the worst case (recall that the coefficient matrix A in 5.1.2 has m rows and n columns). So we see that in general we have to blow up the size of an LP-problem in order to cast it into the LP-formulation of a pure processing network problem.

We have pointed out in Section 5.2 that, having the generalized processing network interpretation in mind, the Simplex PRON procedure of Section 4.4 can in principle be applied to general LP-problems of the form 5.1.1-5.1.3. Moreover, in Section 5.3, we discussed that, having the "almost" pure processing network interpretation in mind, the Simplex PRON procedures of Sections 3.4 and 3.5 can easily be adapted to solve general LP-problems of the form 5.1.1-5.1.3,5.3.1.

Hence we do not believe that a transformation of a general LP-problem 5.1.1-5.1.3 to a pure processing network problem, as described in the proof of Theorem 5.4.1, yields a problem which can be solved easier.

Nevertheless Theorem 5.4.1 is important for several reasons:

1. It shows that a pure processing network structure is not as special as it seems at first sight.

2. It gives a certain reassurance that we have exploited the processing network structure in an adequate way in the Simplex PRON procedures of Chapters 3 and 4, considering the fact that these procedures are closely related to well-known sparse matrix LP-approaches (see Section 5.2).
3. Processing networks have the nice feature that their structure can be visualized by drawing network diagrams. The relevance of visualizing a model has already be pointed out by GLOVER & KLINGMAN [1977] and by GLOVER, HULTZ & KLINGMAN [1978]. Both for model builders and management, a diagram can give more insight into the model structure than an algebraic statement alone. Hence a good visualization of a model may tend to increase management's confidence in such a model.
   Since a general LP-problem can be cast into a processing network fitting a tool is available to visualize its structure. Especially in case such an LP-problem has a natural interpretation as a network flow problem, it may be useful to draw a processing-network diagram. In Chapter 7 we briefly describe a bank balance problem for which we have done this.

5.5. *Some examples*

In this section we discuss three important classes of LP-problems which can be interpreted as a pure processing network problem or as a generalized processing network problem with positive multipliers, without having to blow up the size of the problem.

1. Consider the LP-problem:

5.5.1. minimize $c'x$

5.5.2. $A_1x = b_1$

5.5.3. $A_2x \leq b_2$

5.5.4. $0 \leq x \leq u$

where A_1 is a matrix with at least two elements unequal to zero and exactly one negative element in each column,
A_2 is a nonnegative matrix.
LP-problems of the form 5.5.1-5.5.4 appear in many practical situations.
Observe that A_1 may describe:

- a pure network,
- a pure processing network with only refining nodes,
- a generalized network with positive multipliers,
- a generalized processing network with positive multipliers and only refining nodes.

Considering the fact that each column of $\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ has exactly one negative element, problem 5.5.1-5.5.4 can immediately be regarded as a generalized processing network with positive multipliers and only refining nodes.
A well-known member of this class of LP-problems is the so-called multicommodity network flow problem, which can be stated as follows:
Consider a network G(N,A), with node set N and arc set A. Suppose we have k types of goods (commodities) which flow through this network.
Let the demand or supply of commodity ℓ ($\ell = 1,\ldots,k$) in node i be given by b_i^ℓ. Furthermore, we assume that

x_{ij}^ℓ denotes the amount of flow of the ℓ th commodity through arc $(i,j) \in A$,

c_{ij}^ℓ denotes the cost for transporting a unit of flow of the ℓ th commodity through arc $(i,j) \in A$, and

u_{ij}^ℓ denotes the upper bound for the amount of flow of the ℓ th commodity through arc $(i,j) \in A$.

Finally, let u_{ij} denote the upper bound for the total amount of flow (i.e., the sum of the flows of the k commodities) through arc $(i,j) \in A$. Then the LP-formulation of the multicommodity network flow problem is:

5.5.5. $$\text{minimize} \sum_{\ell=1}^{k} c_{ij}^\ell x_{ij}^\ell$$

5.5.6. $$-\sum_{j \in A(i)} x_{ij}^\ell + \sum_{j \in B(i)} x_{ji}^\ell = b_i^\ell \quad , \quad i \in N,\ \ell = 1,\ldots,k$$

5.5.7. $$\sum_{\ell=1}^{k} x_{ij}^\ell \leq u_{ij} \quad , \quad (i,j) \in A$$

5.5.8. $$0 \leq x_{ij}^\ell \leq u_{ij}^\ell \quad , \quad (i,j) \in A,\ \ell = 1,\ldots,k \ .$$

Equations 5.5.6 are the conservation of flow equations, relations 5.5.7 describe the multicommodity aspect of the problem: the sum of the flows of the k commodities through arc (i,j) has upper bound u_{ij}. The relations

5.5.8 simply denote the capacity bounds for each commodity in each arc of the network. Obviously 5.5.6 is related to 5.5.2 and 5.5.7 to 5.5.3. Consider the network G(N,A) for each of the k commodities. Denote the nodes of the network of the ℓ th commodity by i_ℓ and the arcs by $(i,j)_\ell$, $\ell = 1,\ldots,k$.

Figure 5.5.1 illustrates how the multicommodity network flow problem can be interpreted as a generalized processing network problem with positive multipliers and only refining nodes of order 2.

Suppose there is a flow of magnitude x_{ij}^{ℓ} in an outgoing arc of node i_ℓ in Figure 5.5.1. Multiply this flow with a factor 2, and split it up in two equal portions using a refining node. On each of the outgoing arcs of such a refining node the flow equals again x_{ij}^{ℓ}. One of the outgoing arcs is attached to node j_ℓ of the network of the ℓ th commodity. The other one leads to an additional node, say v_{ij}. We do this for all $\ell = 1,\ldots,k$. Finally we consider an outgoing arc of node v_{ij} and assume it has a lower bound 0 and upper bound u_{ij}.

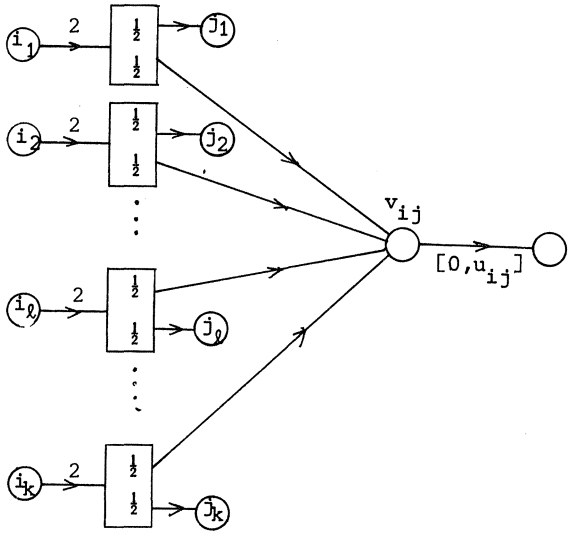


Figure 5.5.1. A processing network interpretation to the multicommodity network flow problem.

2. Consider the LP-problem:

5.5.9. minimize $c'x$

5.5.10. $Ax \leq b$

5.5.11. $0 \leq x \leq u$,

where $A = [a_{ij}]$ is a nonnegative matrix. Vector b is a vector with all elements $b_i > 0$. Let the columns of A be scaled such that $e'A = e'$. Introduce, similarly as in 5.3.1, the redundant constraint:

5.5.12. $-e'Ax \geq -e'b$,

then problem 5.5.9-5.5.12 can immediately be interpreted as a pure processing network problem. See Figure 5.5.2, where for convenience it is assumed that all $a_{ij} > 0$, $i = 1,\ldots,m$, $j = 1,\ldots,n$.

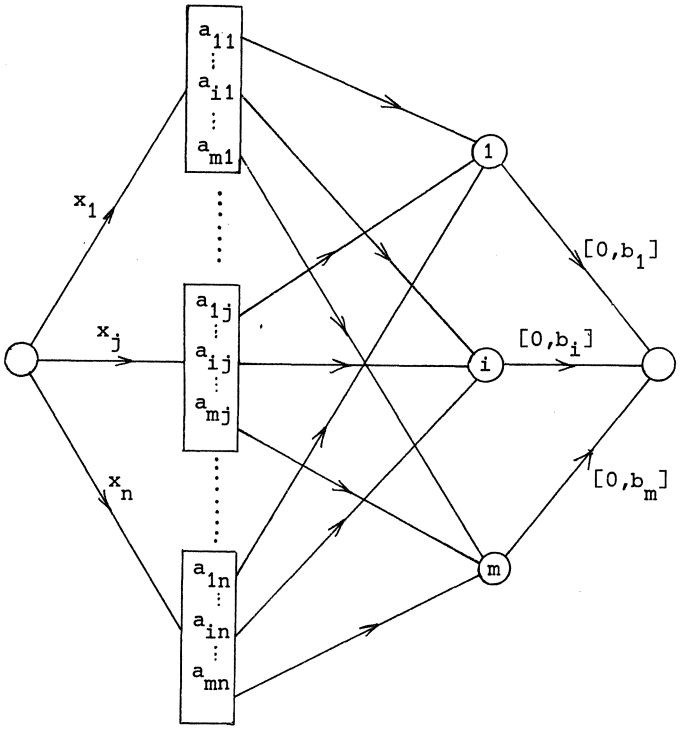
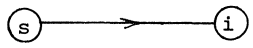


Figure 5.5.2. Pure processing network diagram for the LP-problem 5.5.9-5.5.12.

3. Consider the LP-problem 5.1.1-5.1.3. Assume that each column of A in 5.1.2 has at most two elements unequal to zero. Clearly 5.1.1-5.1.3 can be considered as the LP-formulation of a generalized network flow problem, say with corresponding network G(N,A) (cf. Remark 2.5.2).
Add again constraint 5.3.1 then 5.1.1-5.1.3,5.3.1 is the LP-formulation of a pure processing network problem. The redundant constraint 5.3.1 corresponds to a node s, which is added to the original network G(N,A). This node s can be considered as a source from or a sink to "outside" the original network. The columns of the coefficient matrix $A^* = \begin{bmatrix} A \\ -e'A \end{bmatrix}$ of LP-problem 5.1.1-5.1.3, 5.3.1 can be classified into seven basic cases.
If a column of A has only one element unequal to zero, say in row i (which corresponds to node i in G(N,A)), there are two basic cases:

1. the element in row i is + 1. Then in row s of A^* a "-1" appears.
   Schematically:

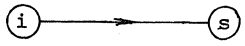
   i : + 1
   s : - 1

   Such a column of A^* describes a transportation process from node s to node i.

2. the element in row i is - 1. Then in row s of A^* a "+1" appears.
   Schematically:

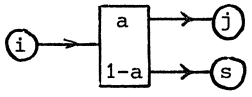
   i : - 1
   s : + 1

   Such a column of A^* describes a transportation process from node i to node s.

If a column of A has two elements unequal to zero, say in rows i and j, the following five basic cases appear for a column in A^*:

3. i : - 1
   j : a (0 < a < 1)
   s : 1 - a (> 0)

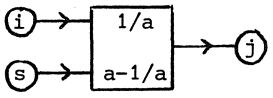
   A fraction a is transported from node i to node j. The rest is lost.

4. i : - 1
   j : 1
   s : 0

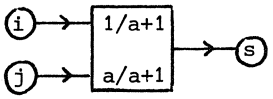
   Transport from node i to node j.

5. i : -1
   j : a (a > 1)
   s : 1-a (< 0)



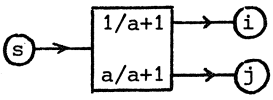
Here a true gain of flow occurs.

6. i : -1
   j : -a (a > 0)
   s : 1+a (> 0)



This process describes the possibility to extract flow from both node i and node j in given proportions.

7. i : 1
   j : a (a > 0)
   s : -1-a (< 0)



Here flow is injected in nodes i and j in given proportions.

Obviously a generalized network flow problem can be transformed to a pure processing network problem in which:

1. all nontransportation processes are of order 2;
2. there exists a particular node s, such that each refining or blending process is incident to this node s.

We know that any LP-problem can be transformed to a pure processing network problem (Section 5.4). Furthermore, it can easily be observed that any pure processing network can be transformed to a pure processing network in which all nontransportation processes are of order 2.
Hence, the essential difference between general LP and generalized networks is the nonvalidity/validity of requirement 2.

6. PROCESSING NETWORKS WITH ADDITIONAL LINEAR CONSTRAINTS

6.1. Introduction

As remarked in Subsection 1.1.2 the processing network structure appears in a large number of real-life situations such as production planning, energy models, assembly and input/output models. In many of such practical situations it occurs that additional requirements must be satisfied, for instance quality requirements, multicommodity aspects, limitations on shared resources. For this reason we consider in this chapter LP-problems of the following type:

6.1.1. minimize $c'x$

6.1.2. $A_1 x = b_1$

6.1.3. $A_2 x = b_2$

6.1.4. $0 \leq x \leq u$,

where c, x and $u \in \mathbb{R}^n$, $b_1 \in \mathbb{R}^m$, $b_2 \in \mathbb{R}^k$.
A_1 is an $m \times n$ matrix, which corresponds to some pure or generalized processing network $G(N,A)$, with node set N consisting of m nodes, and arc set A containing n arcs.
A_2 is a general $k \times n$ matrix.
This type of problem is referred to as a pure or generalized processing network problem (6.1.1, 6.1.2 and 6.1.4) with additional linear constraints (6.1.3).
Sometimes we will simply call these additional linear constraints side constraints.
The dual problem of 6.1.1-6.1.4 is given by

6.1.5. maximize $b_1' \pi_1 + b_2' \pi_2 - u'\nu$

6.1.6. $A_1' \pi_1 + A_2' \pi_2 - \nu \leq c$

6.1.7. $\nu \geq 0$.

We have noted in Sections 5.2 and 5.3 that we can adapt the Simplex PRON procedures of Chapters 3 and 4 in such a way, that they can be applied to general LP-problems. Hence this can be done for LP-problems of the form 6.1.1-6.1.4. However, then the transportation part of A_1 in 6.1.2 would not be exploited fully. For problems of the type 6.1.1-6.1.4 it is natural to partition the coefficient matrix in a processing network part and a non-processing network part. We note that in many practical situations the non-processing network part is small in comparison with the processing network part (i.e., m >> k, cf. GLOVER & KLINGMAN [1981]).
In Section 6.2 we discuss pure processing networks with side constraints. First, we explain the way in which we will partition a basis. Secondly, the basis structure will be exploited in a specification of the primal Simplex algorithm. Similarities and differences with the Simplex SON algorithm of GLOVER & KLINGMAN [1981] will be discussed.
Finally, in Section 6.3, we briefly denote how the contents of Section 6.2 can be generalized to generalized processing networks with side constraints.

6.2. *Pure processing networks with additional linear constraints*

Assume that matrix A_1 in 6.1.2 is associated with a pure processing network as described in Chapter 3.
First we will explain the structure of a basis.

6.2.1. Basis structure

Without loss of generality we can take the following assumptions.

ASSUMPTION 6.2.1. *The rank of matrix* A_1 *in* 6.1.2 *equals* $(m-1)$

ASSUMPTION 6.2.2. *The rank of matrix*

$$6.2.1. \qquad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

equals $(m-1+k)$.

The columns of matrix A in 6.2.1 are associated with the transportation, refining and blending processes of the processing network G(N,A). As in pure networks (Section 2.4) and in pure processing networks (Section 3.3) we introduce a single artificial variable with associated vector $-e_{i_0}$ (i_0 arbitrarily chosen from the set $\{1,\ldots,m\}$). Then it can easily be proved that matrix

6.2.2. $$A^* = [-e_{i_0} \quad A]$$

has rank $(m+k)$.

Let B denote a basis of A^*. Then B can be partitioned as:

6.2.3. $$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix},$$

where B_{11} is a square nonsingular submatrix of $[-e_{i_0} \quad A_1]$ of order m. Note that the column $-e_{i_0}$ is always present in B_{11}.

We observe that B_{11} describes a basis for the pure processing network problem 6.1.1, 6.1.2 and 6.1.4.

Suppose B_{11} contains $(m-q)$ transportation columns, including the slack column, and q refining and blending columns $(0 \leq q \leq m-1)$.

Let matrix T denote the matrix representation of a representative forest, as defined in Section 3.5. The columns of T are sequenced in the same way as the corresponding columns of B_{11}.

According to 3.5.2 we can write:

6.2.4. $$B_{11} = TP ,$$

with

6.2.5. $$P = \begin{bmatrix} I & Q \\ & R \end{bmatrix}$$

where

I is the identity matrix of order $(m-q)$,

Q is a $(m-q) \times q$ matrix, and

R is a square nonsingular matrix of order q.

With respect to matrix B_{12} in 6.2.3 we assume that it contains $(k-g)$ transportation columns and, consequently, g refining and blending columns $(0 \leq g \leq k)$.

Matrix B in 6.2.3 can also be written as:

$$6.2.6. \qquad B = \begin{bmatrix} B_{11} & \\ B_{21} & W \end{bmatrix} \begin{bmatrix} I & B_{11}^{-1} B_{12} \\ & I \end{bmatrix} ,$$

where

$$6.2.7. \qquad W := B_{22} - B_{21} B_{11}^{-1} B_{12} .$$

In the subsequently discussed Simplex algorithm we will use two working bases, namely R in 6.2.5 and W in 6.2.7. R is called the processing working basis and W the general working basis.

6.2.2. The Simplex algorithm for the minimal cost flow problem in a pure processing network with additional linear constraints

In this subsection the essential steps of the Simplex algorithm will be explained in the same sequence as in Section 2.3.
It is assumed that the representative forest, associated with matrix T in 6.2.4 and the inverses of R and W in 6.2.5 and 6.2.7 are kept stored in some convenient way.
Other quantities, which are required in the steps of the Simplex algorithm, are determined when needed by means of pure-network techniques or pure processing-network techniques. This will become apparent in discussing the distinct steps of this Simplex algorithm.

Initialization

Assuming that b_2 in 6.1.3 satisfies $b_2 \geq 0$, the starting basis can be chosen as:

$$6.2.8. \qquad B = \begin{bmatrix} B_{11} & \\ & I \end{bmatrix} ,$$

where B_{11} represents a rooted spanning tree, containing only transportation arcs, as obtained by the procedure in Subsection 3.4.6. Needless to say that all columns of this matrix B may be artificial ones. Comparing 6.2.8 and 6.2.7 we see that the general working basis is the identity matrix. Moreover, P in 6.2.5 is also a unit matrix: initially the processing working basis has size zero.

1. Determining the Simplex multipliers

Let $[c_1' \quad c_2']$ be the partitioning of the basic part of the cost vector c in 6.1.1, compatible with the partition of B in 6.2.3. In order to find the Simplex multipliers we must solve the system:

$$6.2.9. \qquad [\pi_1' \quad \pi_2'] \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = [c_1' \quad c_2'] \ .$$

Considering 6.2.6, the computation of $[\pi_1' \quad \pi_2']$ can be split up in two portions.
First, determine $[\theta_1' \quad \theta_2']$ from:

$$6.2.10. \qquad [\theta_1' \quad \theta_2'] \begin{bmatrix} I & B_{11}^{-1} B_{12} \\ & I \end{bmatrix} = [c_1' \quad c_2'] \ .$$

Formula 6.2.10 reduces to:

$$6.2.11. \qquad \theta_1' = c_1'$$

$$\theta_2' = c_2' - c_1' B_{11}^{-1} B_{12} = c_2' - (c_1' P^{-1})(T^{-1} B_{12}) \ .$$

Note that the expression $T^{-1} B_{12}$ can be evaluated by pure-network techniques (Section 2.4).
Moreover, $c_1' P^{-1}$ can be determined as in pure processing networks, in the way explained in Subsection 3.4.4. We note that P^{-1} can be written as in 3.4.37, as the subsequent discussions in steps 5 and 7 will show.
Secondly, $[\pi_1' \quad \pi_2']$ can be found from

$$6.2.13. \qquad [\pi_1' \quad \pi_2'] \begin{bmatrix} B_{11} & \\ B_{21} & W \end{bmatrix} = [\theta_1' \quad \theta_2'] \ ,$$

which reduces to:

$$6.2.14. \qquad \pi_2' = \theta_2' W^{-1}$$

$$6.2.15. \qquad \pi_1' = (\theta_1' - \pi_2' B_{21}) B_{11}^{-1} = (\theta_1' - \pi_2' B_{21}) P^{-1} T^{-1} \ .$$

The expression in 6.2.15 can be evaluated as explained in Subsection 3.4.4.

2. Calculate the reduced costs

The reduced cost vector $\bar{c}$ can be determined from

$$\bar{c} = \pi_1' A_1 + \pi_2' A_2 - c \ .$$

3. Perform the optimality test

If $\bar{c}_j \leq 0$ for all nonbasic variables at their lower bound, and
if $\bar{c}_j \geq 0$ for all nonbasic variables at their upper bound,
the current solution is optimal and the algorithm stops.

4. Choose the nonbasic variable to enter the basis

Let I denote the index set of all nonbasic variables which violate the optimality test in step 3.
Choose a variable x_k, $k \in I$, to enter the basis. The column of A associated with x_k is given by $\begin{bmatrix} a_{1k} \\ a_{2k} \end{bmatrix}$.

5. Find the representation of the entering column in terms of the basis

Let $\begin{bmatrix} y_{1k} \\ y_{2k} \end{bmatrix}$ denote the representation vector of $\begin{bmatrix} a_{1k} \\ a_{2k} \end{bmatrix}$ in terms of B. This means that $\begin{bmatrix} y_{1k} \\ y_{2k} \end{bmatrix}$ can be evaluated from

$$6.2.16. \quad \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} y_{1k} \\ y_{2k} \end{bmatrix} = \begin{bmatrix} a_{1k} \\ a_{2k} \end{bmatrix} .$$

Again we do this in two stages.
First, solve:

$$6.2.17. \quad \begin{bmatrix} B_{11} & \\ B_{21} & W \end{bmatrix} \begin{bmatrix} \bar{y}_{1k} \\ \bar{y}_{2k} \end{bmatrix} = \begin{bmatrix} a_{1k} \\ a_{2k} \end{bmatrix} ,$$

which leads to:

$$6.2.18. \quad \bar{y}_{1k} = B_{11}^{-1} a_{1k} = P^{-1} T^{-1} a_{1k}$$

$$6.2.19. \quad \bar{y}_{2k} = W^{-1}(a_{2k} - B_{21} \bar{y}_{1k}) .$$

The vector $\bar{y}_{1k}$ can be determined as in pure processing networks (see Subsection 3.4.1). Note that in doing this, matrix P^{-1} is partitioned as in 3.4.37.

Secondly, solve:

6.2.20 $$\begin{bmatrix} I & B_{11}^{-1} B_{12} \\ & I \end{bmatrix} \begin{bmatrix} y_{1k} \\ y_{2k} \end{bmatrix} = \begin{bmatrix} \bar{y}_{1k} \\ \bar{y}_{2k} \end{bmatrix},$$

which reduces to:

6.2.21. $$y_{2k} = \bar{y}_{2k}$$

6.2.22. $$y_{1k} = \bar{y}_{1k} - B_{11}^{-1} B_{12} y_{2k} = \bar{y}_{1k} - P^{-1} T^{-1} B_{12} y_{2k}.$$

We can evaluate the expression $P^{-1} T^{-1} B_{12} y_{2k}$ in 6.2.22 from right to left. Note that for P^{-1} the partitioned form, obtained in determining $\bar{y}_{1k}$ in 6.2.18, can be used.

6. Perform the minimal ratio test

We can perform the minimal ratio test in the standard way, described in Section 2.3. Suppose that x_s with corresponding column $\begin{bmatrix} a_{1s} \\ a_{2s} \end{bmatrix}$ leaves the basis.

7. Update

The value of the objective function and the activity levels of the basic variables can be updated in the standard fashion (see Section 2.3). Updating the working bases is more complex.
Let the vector x_B of basic variables be partitioned as $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, compatible with the partitioning of B.
Using the same terminology as in HARTMAN & LASDON [1972] and CHEN & SAIGAL [1977], we call the variables in x_1 key variables, those in x_2 non-key variables.
In performing the basis change two cases are distinguished.

A. the leaving variable x_s is a non-key variable

In this case the new basis $\hat{B}$ can be written as:

6.2.23. $$\hat{B} = \begin{bmatrix} B_{11} & \hat{B}_{12} \\ B_{21} & \hat{B}_{22} \end{bmatrix},$$

where both $\hat{B}_{12}$ and $\hat{B}_{22}$ differ from B_{12} and B_{22} by exactly one column: column $\begin{bmatrix} a_{1s} \\ a_{2s} \end{bmatrix}$ is replaced by $\begin{bmatrix} a_{1k} \\ a_{2k} \end{bmatrix}$.

The new general working basis $\hat{W}$ satisfies (see 6.2.7):

6.2.24. $\hat{W} = \hat{B}_{22} - B_{21} B_{11}^{-1} \hat{B}_{12}$,

which differs from W in only one column. The column of $\hat{W}$ which is not contained in W is given by $a_{2k} - B_{21} B_{11}^{-1} a_{1k}$. The representation of this column in terms of W is given by the vector $\bar{y}_{2k}$, determined in 6.2.19. This can immediately be observed from 6.2.18 and 6.2.19. This means that $\hat{W}^{-1}$ can be found by performing the standard pivot operation directly to W^{-1}.
Since B_{11} and B_{21} are not changed, the representative forest and the inverse of the processing working basis do not change either. Note that this inverse can be written in block triangular form obtained in determining $\bar{y}_{1k}$ in 6.2.18.

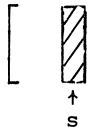
B. the leaving variable x_s is a key-variable

The new basis $\hat{B}$ can be written as

6.2.25. $$\hat{B} = \begin{bmatrix} \hat{B}_{11} & B_{12} \\ \hat{B}_{21} & B_{22} \end{bmatrix} ,$$

where $\hat{B}_{11}$ and $\hat{B}_{21}$ differ from B_{11} and B_{21} by exactly one column.
In this case it can happen that $\hat{B}_{11}$ is singular. Note that $\hat{B}_{11}$ is non-singular iff the coefficient of $\bar{y}_{1k}$ in 6.2.18, which corresponds to the leaving basic variable, is nonzero. If $\hat{B}_{11}$ is nonsingular, we observe that $\hat{W}$ satisfies the following expression:

6.2.26. $\hat{W} = B_{22} - \hat{B}_{21} \hat{B}_{11}^{-1} B_{12} = W - \left[\quad \right] B_{11}^{-1} B_{12}$.



Hence, $\hat{W}^{-1}$ cannot be found by just performing a pivot operation.

In order to update the basis inverse $\hat{B}^{-1}$ we use a two stage approach, proposed earlier by a.o. HARTMAN & LASDON [1972] and CHEN & SAIGAL [1977]:

(a) If possible, interchange column $\begin{bmatrix} a_{1s} \\ a_{2s} \end{bmatrix}$ with some non-key column $\begin{bmatrix} a_{1t} \\ a_{2t} \end{bmatrix}$, such that the resulting matrix B_{11} is nonsingular.

(b) If an interchange has taken place the situation is now that a non-key column leaves the basis.
Hence the rest of the work can be done according to case A described above.

We will call the step under (a) the interchange phase and explain it next in detail.

Interchange phase

The theoretical background for an interchange as meant under (a) is described in CHEN & SAIGAL [1977]. Here only the results are stated.
Let λ' denote the s th row of $B_{11}^{-1} B_{12}$.
Distinguish the two possible cases:

(α) λ is a zero vector.

Then no interchange as meant under (a) is possible. However, observe from 6.2.26 that $\hat{W} = W$, which means that the inverse of the general working basis does not change.
Moreover $\hat{B}_{11}$ in 6.2.25 must be nonsingular.
The inverse of the processing working basis and the new representative forest can be determined in the way of Section 3.5.

(β) λ is not a zero vector.

Then $\begin{bmatrix} a_{1s} \\ a_{2s} \end{bmatrix}$ can be interchanged with any $\begin{bmatrix} a_{1t} \\ a_{2t} \end{bmatrix}$ for which the corresponding coefficient in λ is unequal to zero.
By interchanging $\begin{bmatrix} a_{1s} \\ a_{2s} \end{bmatrix}$ and $\begin{bmatrix} a_{1t} \\ a_{2t} \end{bmatrix}$, the analysis in CHEN & SAIGAL [1977] shows that $\hat{W}^{-1}$ can be found from

6.2.27. $$\hat{W}^{-1} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ \hline & & & -\lambda' & \\ \hline & & & & 1 \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix} W^{-1} .$$

After determining $B_{11}^{-1} a_{1t}$ as in pure processing networks, the inverse of the processing working basis and the new representative forest can be updated using the procedures of Section 3.5.

After the basis change the Simplex algorithm proceeds with step 2.
This completes the description of the Simplex algorithm.

6.2.3. Maximizing the number of transportation processes contained in B_{11}, given B

Concerning matrix B_{11} in 6.2.3 the only requirement in the previous subsection is that B_{11} denotes a square nonsingular matrix. Here we discuss how we can maintain the number of transportation processes in B_{11} as large as possible, given the basis B.

The desirability of keeping the transportation part of B_{11}, given B, as large as possible is quite obvious:

- manipulating with transportation processes is easier than with refining or blending processes.
- given a basis, the size of the processing working basis is as small as possible.
  Except for storage requirements this may save time in performing the steps of the Simplex algorithm of the previous section.

We will first derive a necessary and sufficient condition under which the number of transportation processes in B_{11} is maximal, given a basis B. Furthermore we will present some modification rules for the update step of the Simplex algorithm in Subsection 6.2.2, which guarantee that this condition is satisfied in every iteration of this Simplex algorithm.

Consider the matrix B_1:

$$6.2.28. \quad B_1 := [B_{11} \quad B_{12}] .$$

B_{11} and B_{12} are already introduced in formula 6.2.3. B_{11} is square nonsingular and denotes the key part of the basis. B_{12} describes the non-key part of the basis.

We can partition B_1 in 6.2.28 somewhat further:

$$6.2.29. \quad B_1 = [B_{11}^T \quad B_{11}^P \quad B_{12}^T \quad B_{12}^P]$$

with:

B_{11}^T an m × (m-q) matrix denoting the key transportation processes,

B_{11}^P an m × q matrix denoting the key refining and blending processes,

B_{12}^T an m × (k-g) matrix denoting the non-key transportation processes,

B_{12}^P an m × g matrix denoting the non-key refining and blending processes.

For q and g the following expressions hold: $0 \leq q \leq m-1$ and $0 \leq g \leq k$.

As made clear in Section 3.3, matrix B_{11}^T is associated with the $(q+1)$ transportation trees of the representative forest. Using Theorem 3.4.4 of Subsection 3.4.1 we can prove the following theorem:

THEOREM 6.2.3. *Given the basis* B, *and the fact that* B_{11} *is nonsingular, the number of transportation processes in* B_{11}^T *is maximal iff every transportation process contained in* B_{12}^T *is incident to only one transportation tree of the representative forest.*

PROOF. If $q = 0$ or $q = k$ the statement is trivially true.
Consider the case that $q > 0$ and $q < k$. We first prove the "only if" part and then the "if" part.

only if. Consider a transportation process (i_0, j_0) in B_{12}^T which is incident to two transportation trees. Let its corresponding column in the A_1 part of A in 6.2.1 be given by a. Consider the representation vector y of a in terms of B_{11}, i.e., $y = B_{11}^{-1} a$. Theorem 3.4.4 says that there is at least one refining or blending process in B_{11} which has a nonzero coefficient in the representation vector y. Consequently, column a can be interchanged with a column of B_{11}^P such that the new B_{11} is again nonsingular. Hence we see that in performing such an interchange the number of transportation processes in B_{11} increases by one and the current number of transportation processes in B_{11} cannot be maximal.
if. Suppose that every transportation process in B_{12}^T is incident to only one transportation tree. Consider such a process (i_0, j_0), with corresponding column a in the A_1 part of A in 6.2.1. Theorem 3.4.4 makes clear that a can be written as a linear combination of columns in B_{11}^T only. Hence we can only interchange column a with a column currently contained in B_{11}^T (that is, if we want to keep the new B_{11} nonsingular). In performing such an interchange the number of transportation processes in B_{11} remains the same. Moreover, note that if we would perform such an interchange the nodes i_0 and j_0 both still belong to one transportation tree. Hence there is no use in considering a number of subsequent interchanges in order to achieve a new B_{11} with more transportation processes then the current one. □

We can use Theorem 6.2.3 to accomplish that, in every iteration of the Simplex algorithm of the previous subsection, the number of transportation processes in B_{11} is maximal. For this purpose we only have to adapt the update step of the Simplex algorithm. Note that, in using the initialization

discussed in Subsection 6.2.2, B_{11} initially consists of transportation processes only.
Consider the described possibilities of the update step in Subsection 6.2.2.

A. The leaving variable x_s is a non-key variable

After performing the basis update consider the two possible cases

(a) a transportation process has entered the basis
Determine whether this process is incident to two different transportation trees.
If so, determine its representation in terms of B_{11} and perform an interchange with a column of the B_{11}^P part. Update the working bases and the representative forest in the way described earlier. Obviously the size of the processing working basis is reduced by one.
Otherwise, keep the partition in the way it is now. There is no use in inspecting the other transportation processes currently contained in the B_{12}^T part, since they all still have entries in only one transportation tree.

(b) a refining or blending process has entered the basis
Here no favourable interchange is possible.

B. The leaving variable x_s is a key variable

Consider the interchange phase

(a) vector λ' (the s th row of $B_{11}^{-1} B_{12}$) is a row of zeros.
No interchange is possible.

(b) vector λ' is not a row of zeros

(α) a transportation process leaves the basis
This leaving transportation process corresponds to a transportation arc which is contained in some transportation tree, say T_ℓ. In leaving out this arc the transportation tree T_ℓ splits up in two new trees, say T_ℓ^1 and T_ℓ^2. Test whether there is any transportation process in the B_{12}^T part, which is incident to both T_ℓ^1 and T_ℓ^2.
If so, interchange the corresponding non-key variable with the leaving x_s.
Otherwise x_s must be interchanged with a non-key refining or blending process.

(β) a refining or blending process leaves the basis

In this case x_s can only be interchanged with a non-key refining or blending process.

Note that in all possible cases we have to perform at most one additional interchange of columns in order to accomplish that the number of transportation processes in B_{11}^T is maximal, given the basis B.
As might be expected, the size of the processing working basis increases by one, remains the same or decreases by one in every iteration of the Simplex algorithm.

6.2.4. An extension and a comparison with Simplex SON

We have noted in Section 5.3 that the Simplex PRON appraoch of Section 3.5 can be adapted to solve problems of the form 5.1.1-5.1.3,5.3.1. Hence we immediately see that the approach of the Subsections 6.2.2 and 6.2.3 can in fact be applied to LP/embedded - pure - network problems. These are LP-problems of the form

6.2.30. minimize $c_1' x_1 + c_2' x_2$

6.2.31. $A_{11} x_1 + A_{12} x_2 = b_1$

6.2.32. $A_{21} x_1 + A_{22} x_2 = b_2$

6.2.33. $0 \leq x_1 \leq u_1$

6.2.34. $0 \leq x_2 \leq u_2$

where matrix A_{11} is an m × n matrix, which reflects a pure network structure, and A_{12}, A_{21} and A_{22} are general matrices. The number of constraints in 6.2.32 is k.
GLOVER & KLINGMAN [1981] developed the Simplex SON algorithm to solve LP-problems of the form 6.2.30-6.2.34. It appears that the Simplex PRON approach of this section and Simplex SON use similar ideas at several points. At other points they are different. We briefly discuss the differences and similarities between the current Simplex PRON approach and Simplex SON.
In the Simplex PRON procedure we gave a processing network interpretation to the columns in A_{12} in 6.2.31. We developed a partitioning for a basis in such a way that we could work with two working bases and the representative

forest. The general working basis has a fixed size k (i.e., the number of constraints in 6.2.32). The processing working basis has a variable size q (i.e., the number of nontransportation processes in B_{11}), and is maintained in block triangular form.
In Simplex SON however, the columns in A_{12} are considered as arbitrary columns. The partitioning of a basis is such that all the steps of this algorithm can be performed by a single working basis, and the so-called master basis tree. In the same situation as described above for the Simplex PRON algorithm, the working basis in the Simplex SON algorithm has a variable size (k + q). This working basis is not maintained in block triangular form.

The master basis tree is in fact the same concept as the representative forest, which we used: if we introduce an additional node (m + 1) (GLOVER & KLINGMAN call it the master root) and replace the root-arcs of the transportation trees in the representative forest by arcs from node (m + 1) to the roots of these transportation trees, we have in fact a master basis tree.
Finally we note that GLOVER & KLINGMAN [1981] developed similar ideas as we did for maintaining the number of transportation processes in B_{11} as large as possible, given B (Subsection 6.2.3).

6.3. Generalized processing networks with additional linear constraints

We can easily generalize the ideas of Section 6.2 in order to solve generalized processing network problems with additional linear constraints. Whenever pure-network techniques or pure processing-network techniques were used in that section, they should be replaced by generalized-network techniques or generalized processing-network techniques as explained in Section 2.5 and Chapter 4.
Such an approach can be used to solve LP/embedded-generalized network problems of the form 6.2.30-6.2.34 where matrix A_{11} in 6.2.31 denotes a generalized network structure.
Finally we note that we can keep the transportation part in B_{11} as large as possible in a similar way as done in Subsection 6.2.3.
Using Theorem 4.4.3 we can prove a similar theorem as Theorem 6.2.3.

7. APPLICABILITY AND EXPECTED COMPUTATIONAL RESULTS

This final chapter briefly discusses where the procedures developed can be applied in solving real-world industrial or managerial problems. Also the expected computational performance is considered.

7.1. Applicability

A first class of problems to which the processing network procedures can be applied are of course those which have a natural meaning as a processing network problem (with or without additional linear constraints).
In Chapter 1 the following fields of application are already mentioned:

1. production scheduling in process industry,
2. assembly models,
3. energy models,
4. economic models.

With respect to the latter class we note that they often can be regarded as so-called (pre-) Leontief substitution models considered a.o. by VEINOTT [1968] and KOEHLER, WHINSTON & WRIGHT [1975].
A Leontief matrix has the property that it contains exactly one positive element in each column. Consequently, a Leontief substitution problem can be seen as a generalized processing network problem with positive multipliers and only refining nodes.
Another class of problems which can be interpreted as processing network problems is the class of

5. Markov-control problems,

where quite obviously transition probabilities correspond to processing coefficients in a processing network. It is remarked that Markov-control problems can often be considered as (pre-) Leontief substitution models, see KOEHLER, WHINSTON & WRIGHT [1975] and also KALLENBERG [1980].

Secondly, the procedures developed can in principle be applied to LP/embedded pure or generalized network problems (see Subsection 6.2.4 and Section 6.3), since they put an emphasis on exploiting single commodity network structure. There are many practical LP-problems which have a relatively large embedded network component. As GLOVER & KLINGMAN [1981] put it:
"In general it is our experience that most large-scale LP-problems involving production scheduling, physical distribution, facility location, personnel assignment or personnel promotion contain a large embedded network component, sometimes consisting of several smaller embedded networks."

Thirdly, as already pointed out in Chapter 5, the procedures of Chapters 3 and 4 can in principle be used as a sparse matrix approach for general LP-problems. They fit very well in the "compact-inverse" vision of BASTIAN [1980].

Finally, we refer to a case study performed by the working group "Financial Planning and OR" of the Dutch OR-Society (SOR), KOENE et al. [1981]. This study considers an LP-formulation of the bank balance problem of a general Dutch bank corporation. It is a multiperiod model with certain network flow characteristics: assets can be put out and liabilities can be attracted for a number of periods, such that their totals are in balance in each period. However, a bank is not totally free to do this as it pleases but has to satisfy certain requirements with respect to liquidity, solvability etc. The study shows that it is very well possible to picture out the structure of this model by means of a generalized processing-network diagram (cf. the discussion on visualization in Section 5.4).

7.2. *Expected computational results*

Unfortunately at this time no computational results can be reported, simply because no implementation has been carried out yet.
Nevertheless, in view of the fact that the approaches here are much in the same spirit as the Simplex SON approach of GLOVER & KLINGMAN [1981] and the fact that in addition the typical processing network structure is used (block triangularization) we expect that the approaches developed here should perform better than Simplex SON. GLOVER & KLINGMAN report encouraging preliminary results on some special classes of LP/embedded-pure-network problems, but stress that an exhaustive computational study is required before any serious conclusions can be drawn.

REFERENCES

ADOLPHSON, D.L. [1980], A nondegenerate network Simplex method, Working paper, Graduate School of Business, University of Washington, Seatlle, Washington.

ALI, A.I., R.V. HELGASON, J.L. KENNINGTON & R.S. LALL [1978], Primal Simplex network codes: State-of-the-art implementation technology, Networks, 8, 315-339.

ARONOFSKY, J.S., J.M. DUTTON & M.T. TAYYABKHAN [1978], Managerial planning with linear programming in process industry operations, John Wiley & Sons, New York.

BALACHANDRAN, V., V. SRINIVASAN & G.L. THOMPSON [1981], Applications of the operator theory of parametric programming for the transportation and generalized transportation problems, Mathematical Programming Study 15, 58-85.

BALAS, E. [1966]. The dual method for the generalized transportation problem, Management Science 12, 555-568.

BALAS, E. & P.L. HAMMER [1962]. On the transportation problem - Parts I and II, Cahiers du Centre d'Etudes de Recherche Opérationelle 4, 98-116, 131-160.

BARNES, J.W. & R.M. CRISP [1975], Linear programming: a survey of general purpose algorithms, AIIE Transactions 7, 212-221.

BARR, R., J. ELAM, F. GLOVER & D. KLINGMAN [1980], A network alternating path basis algorithm for transshipment problems, in Fiacco, A. and K. Kortanek (Eds.), Lecture notes in economics and mathematical systems no. 174, Springer Verlag, Berlin.

BARR, R., F. GLOVER & D. KLINGMAN [1974], An improved version of the out-of-kilter method and a comparative study of computer codes, Mathematical Programming 7, 60-87.

BARR, R., F. GLOVER & D. KLINGMAN [1979], Enhancements of spanning tree labeling procedures for network optimization, INFOR 17, 16-34.

BARR, R.S. & J.S. TURNER [1981], Microdata file merging through large-scale network technology, Mathematical Programming Study 15, 1-22.

BASTIAN, M. [1980], Lineare Optimierung grosser Systeme, Compact-Inverse Verfahren und Basisfaktorisierungen, Mathematical systems in economics, Verlagsgruppe Athenäum/Hain, Königstein.

BAZARAA, M.S. & J.J. JARVIS [1977], Linear programming and network flows, John Wiley & Sons, New York.

BENDERS, J.F. [1962], Partitioning procedures for solving mixed-variables programming problems, Numerische Mathematik 4, 238-252.

BENNINGTON, G.E. [1972], An efficient minimal cost flow algorithm, O.R. Report 75, North Carolina State University, Raleigh, North Carolina.

BERGE, C. & A. GHOUILA-HOURI [1965], Programming, games and transportation networks, John Wiley & Sons, New York.

BIXBY, R.E. [1981], Hidden structure in linear programs, in GREENBERG, H.J. & J.S. MAYBEE [1981], 327-360.

BLAND, R.G. [1977], New finite pivoting rules for the Simplex method, Mathematics of Operations Research 2, 103-107.

BOONEKAMP, P.G.M., N.J. KOENDERS, F. VAN OOSTVOORN [1979], Berekening van een centrale variant voor de periode 1976-2000 voor de Nederlandse energiesector met het model SELPE (Dutch), ECN-Report-79-070, ECN, Petten.

BRADLEY, G. [1975], Survey of deterministic networks, AIIE Transactions 7, 222-234.

BRADLEY, G.H., G.G. BROWN & G.W. GRAVES [1977], Design and implementation of large scale primal transshipment algorithms, Management Science 24, 1-34.

BROWN, G.G. & W.G. WRIGHT, Automatic identification of embedded structure in large-scale optimization models, in GREENBERG, H.J. & J.S. MAYBEE [1981], 369-388.

BUNCH, J.R. & P.J. ROSE (Eds.) [1976], Sparse matrix computations, Proceedings of the symposium on sparse matrix computations at Argonne National Laboratory in 1975, Academic Press Inc., New York.

CHARNES, A. & W. COOPER [1961], Management models and industrial applications of linear programming, Volumes I and II, John Wiley & Sons, New York.

CHARNES, A., F. GLOVER, D. KARNEY, D. KLINGMAN & J. STUTZ [1975], Past, present and future of large scale transshipment computer codes and applications, Computers and Operations Research 2, 71-81.

CHEN, S. & R. SAIGAL [1977], A primal algorithm for solving a capacitated network flow problem with additional linear constraints, Networks 7, 59-79.

CUNNINGHAM, W.H. [1976], A network Simplex method, Mathematical Programming 11, 105-116.

CUNNINGHAM, W.H. [1979], Theoretical properties of the network Simplex method, Mathematics of Operations Research 4, 196-208.

CZAYKA, L. [1972], Qualitative Input-Output Analyse, Schriften zur Wirtschaftwissenschaftlichen Forschung, Bd. 42, Verlag Anton Hain, Meisenheim am Glan.

DANTZIG, G.B. [1951], Application of the Simplex method to a transportation problem, in: Koopmans, T.C., Ed., Activity analysis of production and allocation, John Wiley & Sons, New York.

DANTZIG, G.B. [1955], Upper bounds, secondary constraints and block-triangularity, Econometrica 23, 174-183.

DANTZIG, G.B. [1963], Linear programming and extensions, Princeton University Press, Princeton.

DUFF, I.S. [1977a], A survey of sparse matrix research, Proceedings of the IEEE 65, 500-535.

DUFF, I.S. [1977b], On permutations to block triangular form, Journal of the Institute of Mathematics and its Applications 19, 339-342.

DUFF, I.S. [1981], On algorithms for obtaining a maximum transversal, ACM Transactions on Mathematical Software 7, 315-330.

DUFF, I.S. & J.K. REID [1978a], An implementation of Tarjan's algorithm for the block triangularization of a matrix, ACM Transactions on Mathematical Software 4, 137-147.

DUFF, I.S. & J.K. REID [1978b], Algorithm 529. Permutations to block triangular form, ACM Transactions on Mathematical Software 4, 189-192.

EDMONDS, J. & R. KARP [1972], Theoretical improvements in algorithmic efficiency for network flow problems, Journal of the ACM 19, 248-264.

EISEMANN, D. [1964], The generalized stepping stone method for the machine loading model, Management Science 11, 154-177.

ELAM, J., F. GLOVER & D. KLINGMAN [1979], A strongly convergent primal Simplex algorithm for generalized networks, Mathematics of Operations Research 4, 39-59.

ELMAGHRABY, S.E. [1970], The theory of networks and management science, Management Science 17, 1-34, B54-B71.

FERGUSON, A.R. & G.B. DANTZIG [1954], Notes on linear programming: part XVI - The problem of routing aircraft - a mathematical solution, Research Memorandum RM-1369, The RAND Corporation.

FONG, C.O. & V. SRINIVASAN [1977], Determining all nondegenerate shadow prices for the transportation problem, Transportation Science 11, 199-222.

FORD, L.R. & D.R. FULKERSON [1957], A primal-dual algorithm for the capacitated Hitchcock problem, Naval Research Logistics Quarterly 4, 47-54.

FORD, L.R. & D.R. FULKERSON [1958], Suggested computation of maximal multi-commodity network flows, Management Science 5, 57-101.

FORD, L.R. & D.R. FULKERSON [1962], Flows in networks, Princeton University Press, Princeton.

FULKERSON, D,R. [1961], An out-of-kilter method for minimal cost flow problems, SIAM Journal of Applied Mathematics 9, 18-27.

GEOFFRION, A.M. & G.W. GRAVES [1974], Multicommodity distribution system design by Benders decomposition, Management Science 20, 822.

GEURTS, J.G.M. [1980], Een linear programmeringsmodel van de productie-planning op korte termijn, M.Sc. Thesis, Department of Industrial Engineering and Management Science, Eindhoven University of Technology, Eindhoven.

GHOUILA-HOURI, A. [1960], Recherche du flot maximum dans certains réseaux lorsqu'on impose une condition de bouclage, Proceedings of the second International Conference on Operations Research, London, 156.

GLOVER, F. [1981], Creating network structure in LPs, in GREENBERG, H.J. & J.S. MAYBEE [1981], 361-367.

GLOVER, F., J. HULTZ & D. KLINGMAN [1978], Improved computer based planning techniques, Part I, Interfaces 8, 16-25.

GLOVER, F., J. HULTZ, D. KLINGMAN & J. STUTZ [1978], Generalized networks: a fundamental computer-based planning tool, Management Science 24, 1209-1220.

GLOVER, F., D. KARNEY & D. KLINGMAN [1972], The augmented predecessor index method for locating stepping stone paths and assigning dual prices in distribution problems, Transportation Science 6, 171-180.

GLOVER, F., D. KARNEY & D. KLINGMAN [1974], Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problems, Networks 4, 191-212.

GLOVER, F., D. KARNEY, D. KLINGMAN & A. NAPIER [1974], A computational study on start procedures, basis change criteria and solution algorithms for transportation problems, Management Science 20, 793-813.

GLOVER, F. & D. KLINGMAN [1973], On the equivalence of some generalized network problems to pure network problems, Mathematical Programming 4, 369-378.

GLOVER, F. & D. KLINGMAN [1975], Real world applications of network related problems and breakthroughs in solving them efficiently, ACM Transactions on Mathematical Software 1, 47-55.

GLOVER, F. & D. KLINGMAN [1977], Network applications in industry and government, AIIE Transactions 9, 363-376.

GLOVER, F. & D. KLINGMAN [1978a], Modelling and solving network problems, in: GREENBERG, H.J., Ed., Design and implementation of optimization software, Sythoff & Noordhoff, 185-224.

GLOVER, F. & D. KLINGMAN [1978b], Comments on a note by Hatch on network algorithms, Operations Research 26, 370-374.

GLOVER, F. & D. KLINGMAN [1980], Recent developments in computer implementation technology for network flow algorithms, International Workshop on advances in linear optimization algorithms and software, Pisa, Italy.

GLOVER, F. & D. KLINGMAN [1981], The Simplex SON algorithm for LP/embedded network problems, Mathematical Programming Study 15, 148-176.

GLOVER, F., D. KLINGMAN & A. NAPIER [1972], An efficient dual approach to network problems, OPSEARCH 9, 1-18.

GLOVER, F., D. KLINGMAN & J. STUTZ [1973], Extensions of the augmented predecessor index method to generalized network problems, Transportation Science 7, 377-384.

GLOVER, F., D. KLINGMAN & J. STUTZ [1974], The augmented treaded index method for network optimization, INFOR 12, 293-298.

GLOVER, F. & J. MULVEY [1980], Equivalence of the 0-1 integer programming problem to discrete generalized and pure networks, Operations Research 28, 829-836.

GOLDEN, B., M. BALL & L. BODIN [1981], Current and future research directions in network optimization, Computers & Operations Research 8, 71-81.

GRAVES, G.W. & R.D. MC BRIDE [1976], The factorization approach to large-scale linear programming, Mathematical Programming 10, 91-110.

GREENBERG, H.J. & J.S. MAYBEE [1981], Computer assited analysis and model simplification, Proceedings of the first symposium on computer-assisted analysis and model simplification, Boulder, Colorado, March 1980, Academic Press, New York.

GUNAWARDANE, G., S. HOFF & L. SCHRAGE [1981], Identification of special structure constraints in linear programs, Mathematical Programming 21, 90-97.

GUPTA, R. [1978], Solving the generalized transportation problem with constraints, Zeitschrift für Angewandte Mechanik und Mathematik 58, 451-458.

GUSTAVSON, F. [1976], Finding the block lower triangular form of a sparse matrix, in: BUNCH, J.R. and D.J. ROSE (Eds.) [1976], 275-289.

HALL, P. [1935], On representatives of subsets, Journal of the London Mathematical Society 10, 26-30.

HARTMAN, J.K. & L.S. LASDON [1972], A generalized upperbounding algorithm for multicommodity network flow problems, Networks 1, 333-354.

HATCH, R. [1975], Bench marks comparing transportation codes based on primal Simplex and primal-dual algorithms, Operations Research 23, 1167-1172.

HELGASON, R.V. & J.L. KENNINGTON [1977], A product form representation of the inverse of a multicommodity cycle matrix, Networks 7, 297-322.

HITCHCOCK, F.L. [1941], The distribution of a product from several sources to numerous locations, Journal of Mathematics and Physics 20, 224-236.

HOWELL, T.D. [1976], Partitioning using PAQ, in: BUNCH, J.R. & D.J. ROSE [1976], 23-37.

HULTZ, J. & D. KLINGMAN [1976], Solving constrained generalized network problems, Research Report CCS 257, Center for Cybernetic Studies, University of Texas, Austin.

ITAI, A. [1978], Two commodity flows, Journal of the ACM 25, 596-611.

JENSEN, P.A. & J.W. BARNES [1980], Network flow programming, John Wiley & Sons, New York.

JEWELL, W.S. [1962], Optimal flow through networks with gains, Operations Research 10, 476-499.

JEWELL, W.S. [1966], Multicommodity network solutions, Research Report ORC 66-24, University of California, Berkeley.

JOHNSON, E.L. [1966], Networks and basic solutions, Operations Research 14, 619-623.

KALLENBERG, L.C.M. [1980], Linear programming and finite Markovian control problems, Ph.D. Thesis, University of Leiden, Leiden.

KANTOROVICH, L.V. [1939], Mathematical methods in the organization and planning of production, Publication House of the Leningrad State University. Translated in Management Science 6 [1960], 366-422.

KANTOROVICH, L.V. & M.K. GAVURIN [1949], The application of mathematical methods to problems of freight flow analysis, Akademii Nauk SSSR.

KENNINGTON, J.L. [1977], Solving multicommodity transportation problems using a primal partitioning simplex technique, Naval Research Logistics Quarterly 24, 309-325.

KENNINGTON, J.L. [1978], A survey of linear cost multicommodity network flows, Operations Research 26, 209-236.

KEVORKIAN, A.K. [1979], The principal maxmin matrix transversal strategy, Mathematics of Operations Research 4, 274-290.

KLEIN, M. [1967], A primal method for minimal cost flow with applications to the assignment and transportation problems, Management Science 14, 205-220.

KLINGMAN, D., A. NAPIER & J. STUTZ [1974], NETGEN - A program for generating large scale (un)capacitated assignment, transportation and minimum cost flow network problems, Management Science 20, 814-821.

KLINGMAN, D. & R. RUSSELL [1975], Solving constrained transportation problems, Operations Research 23, 91-108.

KOEHLER, G.J., A.B. WHINSTON & G.P. WRIGHT [1975], Optimization over Leontief substitution systems, North Holland Publishing Co., Amsterdam.

KOENE, J. [1979a], A primal-dual algorithm for solving a maximal flow problem in a class of networks with gains, Memorandum COSOR 79-13, Eindhoven University of Technology, Eindhoven.

KOENE, J. [1979b], Scaling a network with positive gains to a lossy or gainy network, Memorandum COSOR 79-18, Eindhoven University of Technology, Eindhoven.

KOENE, J. [1980], Maximal flow through a processing network with the source as the only processing node, Memorandum COSOR 80-02, Revised version, Eindhoven University of Technology, Eindhoven.

KOENE, J. [1981a], Processing networks: introduction and basis structure, Memorandum COSOR 81-06, Eindhoven University of Technology, Eindhoven.

KOENE, J. [1981b], A primal Simplex algorithm for the minimal cost flow problem in a processing network, Memorandum COSOR 81-09, Eindhoven University of Technology, Eindhoven.

KOENE, J., J.W. BOOGERD, T.A. VAN BREUKELEN, H.J.J. BRONSEMA, Th.M. COFFENG, L.L.M. VAN GAAL, H.N. GLORIE, M.C. HUISMAN, M.J.M. SMOLDERS & R.W.M.M. VAN DE VEN [1981], Balance optimization for bank corporations, formulated as a processing network problem (in Dutch), Research memorandum no. 93, Institute for Economic Research, University of Groningen, Groningen.

KOOPMANS, T.C. [1947]. Optimum utilization of the transportation system, Proceedings of the International Statistical Conference, 1947. Also in: Scientific papers of Tjalling C. Koopmans, Springer Verlag, 1970, 184-190.

LASDON, L.S. [1970], Optimization theory for large systems, Macmillan, New York.

LAWLER, E.L. [1976], Combinatorial optimization; networks and matroids, Holt, Rinehart & Whinston, New York.

MAIER, S.F. [1971], A compact inverse scheme applied to a multicommodity network with resource constraints. Technical Report no. 71-8, Operations Research House, Stanford University, Stanford.

MANNE, A.S., R.G. RICHELS & J.P. WEYNANT [1979], Energy policy modelling: a survey, Operations Research 27, 1-36.

MARKOWITZ, H.M. [1954], Concepts and computing procedures for certain x_{ij} programming problems, Paper P-602, The RAND Corporation.

MAURRAS, J. [1972], Optimization of the flow through networks with gains, Mathematical Programming 3, 135-144.

MAYEDA, W. [1968], Maximal flow under controlled edge flows, Proceedings 1968 International Conference on Communications, Philadelphia, Pennsylvania, june 1968.

MC BRIDE, R.D. [1978], A spike collective dynamic factorization algorithm for the Simplex method, Management Science 24, 1031-1042.

MULVEY, J. [1978], Pivot strategies for primal Simplex network codes, Journal of the ACM 25, 266-270.

VAN NUNEN, J.A.E.E. & J.F. BENDERS [1981], A decision report system for location and allocation problems within a brewery, Working Paper, Graduate School of Management, Delft. To appear in DGOR-Operations Research Proceedings, 1981, Springer Verlag, Berlin.

ORDEN, A. [1956], The transshipment problem, Management Science 3, 276-285.

PETER, H. [1954], Mathematische Strukturlehre des Wirtschaftskreislaufes Göttingen.

PHILLIPS, W. [1970], The storage and inversion of large sparse matrices, ICI Wilmslow Report, Mathematics and Statistics Group.

RAMAKRISHNAN, K.G. [1980], Solving two-commodity transportation problems with coupling constraints, Journal of the ACM 27, 736-757.

ROBACKER, J.T. [1956], Concerning multicommodity networks, Research Memorandum RM-1799, The Rand Corporation, Santa Monica.

RYAN, D.R. & S. CHEN [1981], A comparison of three algorithms for finding fundamental cycles in a directed graph, Networks 11, 1-12.

SAIGAL, R. [1967], Multicommodity flows in directed networks, ORC Report 67-38, University of California, Berkeley.

SARGENT, R.W.H. & A.W. WESTENBERG [1964], "Speed up" in chemical engineering design, Transactions of the Institute of Chemical Engineers 42,190-197.

SAUNDERS, M.A. [1972], Large scale linear programming using the Choleski factorization, Report CS-72-252, Computer Science Department, Stanford University, Stanford.

SCHAEFER, A. [1978], Netze mit Verteilungsfaktoren, Verlag Anton Hain, Meisenheim am Glan.

SCHRAGE, L. [1975], Implicit representation of variable upper bounds in linear programming, Mathematical Programming Study 4, 118-132.

SCHRAGE, L. [1981], Some comments on hidden structure in linear programs, in GREENBERG, H.J. & J.S. MAYBEE [1981], 389-395.

STEINBERG, E. & H.A. NAPIER [1980], Optimal multi-level lot sizing for requirements planning systems, Management Science 26, 1258-1271.

TARJAN, R.E. [1972], Depth-first search and linear graph algorithms, SIAM Journal on Computing 1, 146-160.

TOMLIN, J.A. [1966], Minimum-cost multicommodity network flows, Operations Research 14, 45-51.

TRUEMPER, K. [1976], An efficient scaling procedure for gain networks, Networks 6, 151-160.

VEINOTT, A.F. [1968], Extreme points of Leontief substitution systems, Linear Algebra and its Applications 1, 181-194.

SUBJECT INDEX

MATHEMATICAL CENTRE TRACTS

1 T. van der Walt. *Fixed and almost fixed points.* 1963.

2 A.R. Bloemena. *Sampling from a graph.* 1964.

3 G. de Leve. *Generalized Markovian decision processes, part I: model and method.* 1964.

4 G. de Leve. *Generalized Markovian decision processes, part II: probabilistic background.* 1964.

5 G. de Leve, H.C. Tijms, P.J. Weeda. *Generalized Markovian decision processes, applications.* 1970.

6 M.A. Maurice. *Compact ordered spaces.* 1964.

7 W.R. van Zwet. *Convex transformations of random variables.* 1964.

8 J.A. Zonneveld. *Automatic numerical integration.* 1964.

9 P.C. Baayen. *Universal morphisms.* 1964.

10 E.M. de Jager. *Applications of distributions in mathematical physics.* 1964.

11 A.B. Paalman-de Miranda. *Topological semigroups.* 1964.

12 J.A.Th.M. van Berckel, H. Brandt Corstius, R.J. Mokken, A. van Wijngaarden. *Formal properties of newspaper Dutch.* 1965.

13 H.A. Lauwerier. *Asymptotic expansions.* 1966, out of print; replaced by MCT 54.

14 H.A. Lauwerier. *Calculus of variations in mathematical physics.* 1966.

15 R. Doornbos. *Slippage tests.* 1966.

16 J.W. de Bakker. *Formal definition of programming languages with an application to the definition of ALGOL 60.* 1967.

17 R.P. van de Riet. *Formula manipulation in ALGOL 60, part 1.* 1968.

18 R.P. van de Riet. *Formula manipulation in ALGOL 60, part 2.* 1968.

19 J. van der Slot. *Some properties related to compactness.* 1968.

20 P.J. van der Houwen. *Finite difference methods for solving partial differential equations.* 1968.

21 E. Wattel. *The compactness operator in set theory and topology.* 1968.

22 T.J. Dekker. *ALGOL 60 procedures in numerical algebra, part 1.* 1968.

23 T.J. Dekker, W. Hoffmann. *ALGOL 60 procedures in numerical algebra, part 2.* 1968.

24 J.W. de Bakker. *Recursive procedures.* 1971.

25 E.R. Paërl. *Representations of the Lorentz group and projective geometry.* 1969.

26 European Meeting 1968. *Selected statistical papers, part I.* 1968.

27 European Meeting 1968. *Selected statistical papers, part II.* 1968.

28 J. Oosterhoff. *Combination of one-sided statistical tests.* 1969.

29 J. Verhoeff. *Error detecting decimal codes.* 1969.

30 H. Brandt Corstius. *Exercises in computational linguistics.* 1970.

31 W. Molenaar. *Approximations to the Poisson, binomial and hypergeometric distribution functions.* 1970.

32 L. de Haan. *On regular variation and its application to the weak convergence of sample extremes.* 1970.

33 F.W. Steutel. *Preservation of infinite divisibility under mixing and related topics.* 1970.

34 I. Juhász, A. Verbeek, N.S. Kroonenberg. *Cardinal functions in topology.* 1971.

35 M.H. van Emden. *An analysis of complexity.* 1971.

36 J. Grasman. *On the birth of boundary layers.* 1971.

37 J.W. de Bakker, G.A. Blaauw, A.J.W. Duijvestijn, E.W. Dijkstra, P.J. van der Houwen, G.A.M. Kamsteeg-Kemper, F.E.J. Kruseman Aretz, W.L. van der Poel, J.P. Schaap-Kruseman, M.V. Wilkes, G. Zoutendijk. *MC-25 Informatica Symposium.* 1971.

38 W.A. Verloren van Themaat. *Automatic analysis of Dutch compound words.* 1972.

39 H. Bavinck. *Jacobi series and approximation.* 1972.

40 H.C. Tijms. *Analysis of (s,S) inventory models.* 1972.

41 A. Verbeek. *Superextensions of topological spaces.* 1972.

42 W. Vervaat. *Success epochs in Bernoulli trials (with applications in number theory).* 1972.

43 F.H. Ruymgaart. *Asymptotic theory of rank tests for independence.* 1973.

44 H. Bart. *Meromorphic operator valued functions.* 1973.

45 A.A. Balkema. *Monotone transformations and limit laws.* 1973.

46 R.P. van de Riet. *ABC ALGOL, a portable language for formula manipulation systems, part 1: the language.* 1973.

47 R.P. van de Riet. *ABC ALGOL, a portable language for formula manipulation systems, part 2: the compiler.* 1973.

48 F.E.J. Kruseman Aretz, P.J.W. ten Hagen, H.L. Oudshoorn. *An ALGOL 60 compiler in ALGOL 60, text of the MC-compiler for the EL-X8.* 1973.

49 H. Kok. *Connected orderable spaces.* 1974.

50 A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck, C.H.A. Koster, M. Sintzoff, C.H. Lindsey, L.G.L.T. Meertens, R.G. Fisker (eds.). *Revised report on the algorithmic language ALGOL 68.* 1976.

51 A. Hordijk. *Dynamic programming and Markov potential theory.* 1974.

52 P.C. Baayen (ed.). *Topological structures.* 1974.

53 M.J. Faber. *Metrizability in generalized ordered spaces.* 1974.

54 H.A. Lauwerier. *Asymptotic analysis, part 1.* 1974.

55 M. Hall, Jr., J.H. van Lint (eds.). *Combinatorics, part 1: theory of designs, finite geometry and coding theory.* 1974.

56 M. Hall, Jr., J.H. van Lint (eds.). *Combinatorics, part 2: graph theory, foundations, partitions and combinatorial geometry.* 1974.

57 M. Hall, Jr., J.H. van Lint (eds.). *Combinatorics, part 3: combinatorial group theory.* 1974.

58 W. Albers. *Asymptotic expansions and the deficiency concept in statistics.* 1975.

59 J.L. Mijnheer. *Sample path properties of stable processes.* 1975.

60 F. Göbel. *Queueing models involving buffers.* 1975.

63 J.W. de Bakker (ed.). *Foundations of computer science.* 1975.

64 W.J. de Schipper. *Symmetric closed categories.* 1975.

65 J. de Vries. *Topological transformation groups, 1: a categorical approach.* 1975.

66 H.G.J. Pijls. *Logically convex algebras in spectral theory and eigenfunction expansions.* 1976.

68 P.P.N. de Groen. *Singularly perturbed differential operators of second order.* 1976.

69 J.K. Lenstra. *Sequencing by enumerative methods.* 1977.

70 W.P. de Roever, Jr. *Recursive program schemes: semantics and proof theory.* 1976.

71 J.A.E.E. van Nunen. *Contracting Markov decision processes.* 1976.

72 J.K.M. Jansen. *Simple periodic and non-periodic Lamé functions and their applications in the theory of conical waveguides.* 1977.

73 D.M.R. Leivant. *Absoluteness of intuitionistic logic.* 1979.

74 H.J.J. te Riele. *A theoretical and computational study of generalized aliquot sequences.* 1976.

75 A.E. Brouwer. *Treelike spaces and related connected topological spaces.* 1977.

76 M. Rem. *Associons and the closure statement.* 1976.

77 W.C.M. Kallenberg. *Asymptotic optimality of likelihood ratio tests in exponential families.* 1978.

78 E. de Jonge, A.C.M. van Rooij. *Introduction to Riesz spaces.* 1977.

79 M.C.A. van Zuijlen. *Emperical distributions and rank statistics.* 1977.

80 P.W. Hemker. *A numerical study of stiff two-point boundary problems.* 1977.

81 K.R. Apt, J.W. de Bakker (eds.). *Foundations of computer science II, part 1.* 1976.

82 K.R. Apt, J.W. de Bakker (eds.). *Foundations of computer science II, part 2.* 1976.

83 L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the AUTOMATH system.* 1979.

84 H.L.L. Busard. *The translation of the elements of Euclid from the Arabic into Latin by Hermann of Carinthia (?), books vii-xii.* 1977.

85 J. van Mill. *Supercompactness and Wallman spaces.* 1977.

86 S.G. van der Meulen, M. Veldhorst. *Torrix I, a programming system for operations on vectors and matrices over arbitrary fields and of variable size.* 1978.

88 A. Schrijver. *Matroids and linking systems.* 1977.

89 J.W. de Roever. *Complex Fourier transformation and analytic functionals with unbounded carriers.* 1978.

90 L.P.J. Groenewegen. *Characterization of optimal strategies in dynamic games.* 1981.

91 J.M. Geysel. *Transcendence in fields of positive characteristic.* 1979.

92 P.J. Weeda. *Finite generalized Markov programming.* 1979.

93 H.C. Tijms, J. Wessels (eds.). *Markov decision theory.* 1977.

94 A. Bijlsma. *Simultaneous approximations in transcendental number theory.* 1978.

95 K.M. van Hee. *Bayesian control of Markov chains.* 1978.

96 P.M.B. Vitányi. *Lindenmayer systems: structure, languages, and growth functions.* 1980.

97 A. Federgruen. *Markovian control problems; functional equations and algorithms.* 1984.

98 R. Geel. *Singular perturbations of hyperbolic type.* 1978.

99 J.K. Lenstra, A.H.G. Rinnooy Kan, P. van Emde Boas (eds.). *Interfaces between computer science and operations research.* 1978.

100 P.C. Baayen, D. van Dulst, J. Oosterhoff (eds.). *Proceedings bicentennial congress of the Wiskundig Genootschap, part 1.* 1979.

101 P.C. Baayen, D. van Dulst, J. Oosterhoff (eds.). *Proceedings bicentennial congress of the Wiskundig Genootschap, part 2.* 1979.

102 D. van Dulst. *Reflexive and superreflexive Banach spaces.* 1978.

103 K. van Harn. *Classifying infinitely divisible distributions by functional equations.* 1978.

104 J.M. van Wouwe. *Go-spaces and generalizations of metrizability.* 1979.

105 R. Helmers. *Edgeworth expansions for linear combinations of order statistics.* 1982.

106 A. Schrijver (ed.). *Packing and covering in combinatorics.* 1979.

107 C. den Heijer. *The numerical solution of nonlinear operator equations by imbedding methods.* 1979.

108 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science III, part 1.* 1979.

109 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science III, part 2.* 1979.

110 J.C. van Vliet. *ALGOL 68 transput, part I: historical review and discussion of the implementation model.* 1979.

111 J.C. van Vliet. *ALGOL 68 transput, part II: an implementation model.* 1979.

112 H.C.P. Berbee. *Random walks with stationary increments and renewal theory.* 1979.

113 T.A.B. Snijders. *Asymptotic optimality theory for testing problems with restricted alternatives.* 1979.

114 A.J.E.M. Janssen. *Application of the Wigner distribution to harmonic analysis of generalized stochastic processes.* 1979.

115 P.C. Baayen, J. van Mill (eds.). *Topological structures II, part 1.* 1979.

116 P.C. Baayen, J. van Mill (eds.). *Topological structures II, part 2.* 1979.

117 P.J.M. Kallenberg. *Branching processes with continuous state space.* 1979.

118 P. Groeneboom. *Large deviations and asymptotic efficiencies.* 1980.

119 F.J. Peters. *Sparse matrices and substructures, with a novel implementation of finite element algorithms.* 1980.

120 W.P.M. de Ruyter. *On the asymptotic analysis of large-scale ocean circulation.* 1980.

121 W.H. Haemers. *Eigenvalue techniques in design and graph theory.* 1980.

122 J.C.P. Bus. *Numerical solution of systems of nonlinear equations.* 1980.

123 I. Yuhász. *Cardinal functions in topology - ten years later.* 1980.

124 R.D. Gill. *Censoring and stochastic integrals.* 1980.

125 R. Eising. *2-D systems, an algebraic approach.* 1980.

126 G. van der Hoek. *Reduction methods in nonlinear programming.* 1980.

127 J.W. Klop. *Combinatory reduction systems.* 1980.

128 A.J.J. Talman. *Variable dimension fixed point algorithms and triangulations.* 1980.

129 G. van der Laan. *Simplicial fixed point algorithms.* 1980.

130 P.J.W. ten Hagen, T. Hagen, P. Klint, H. Noot, H.J. Sint, A.H. Veen. *ILP: intermediate language for pictures.* 1980.

131 R.J.R. Back. *Correctness preserving program refinements: proof theory and applications.* 1980.

132 H.M. Mulder. *The interval function of a graph.* 1980.

133 C.A.J. Klaassen. *Statistical performance of location estimators.* 1981.

134 J.C. van Vliet, H. Wupper (eds.). *Proceedings international conference on ALGOL 68.* 1981.

135 J.A.G. Groenendijk, T.M.V. Janssen, M.J.B. Stokhof (eds.). *Formal methods in the study of language, part I.* 1981.

136 J.A.G. Groenendijk, T.M.V. Janssen, M.J.B. Stokhof (eds.). *Formal methods in the study of language, part II.* 1981.

137 J. Telgen. *Redundancy and linear programs.* 1981.

138 H.A. Lauwerier. *Mathematical models of epidemics.* 1981.

139 J. van der Wal. *Stochastic dynamic programming, successive approximations and nearly optimal strategies for Markov decision processes and Markov games.* 1981.

140 J.H. van Geldrop. *A mathematical theory of pure exchange economies without the no-critical-point hypothesis.* 1981.

141 G.E. Welters. *Abel-Jacobi isogenies for certain types of Fano threefolds.* 1981.

142 H.R. Bennett, D.J. Lutzer (eds.). *Topology and order structures, part 1.* 1981.

143 J.M. Schumacher. *Dynamic feedback in finite- and infinite-dimensional linear systems.* 1981.

144 P. Eijgenraam. *The solution of initial value problems using interval arithmetic; formulation and analysis of an algorithm.* 1981.

145 A.J. Brentjes. *Multi-dimensional continued fraction algorithms.* 1981.

146 C.V.M. van der Mee. *Semigroup and factorization methods in transport theory.* 1981.

147 H.H. Tigelaar. *Identification and informative sample size.* 1982.

148 L.C.M. Kallenberg. *Linear programming and finite Markovian control problems.* 1983.

149 C.B. Huijsmans, M.A. Kaashoek, W.A.J. Luxemburg, W.K. Vietsch (eds.). *From A to Z, proceedings of a symposium in honour of A.C. Zaanen.* 1982.

150 M. Veldhorst. *An analysis of sparse matrix storage schemes.* 1982.

151 R.J.M.M. Does. *Higher order asymptotics for simple linear rank statistics.* 1982.

152 G.F. van der Hoeven. *Projections of lawless sequences.* 1982.

153 J.P.C. Blanc. *Application of the theory of boundary value problems in the analysis of a queueing model with paired services.* 1982.

154 H.W. Lenstra, Jr., R. Tijdeman (eds.). *Computational methods in number theory, part I.* 1982.

155 H.W. Lenstra, Jr., R. Tijdeman (eds.). *Computational methods in number theory, part II.* 1982.

156 P.M.G. Apers. *Query processing and data allocation in distributed database systems.* 1983.

157 H.A.W.M. Kneppers. *The covariant classification of two-dimensional smooth commutative formal groups over an algebraically closed field of positive characteristic.* 1983.

158 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science IV, distributed systems, part 1.* 1983.

159 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science IV, distributed systems, part 2.* 1983.

160 A. Rezus. *Abstract AUTOMATH.* 1983.

161 G.F. Helminck. *Eisenstein series on the metaplectic group, an algebraic approach.* 1983.

162 J.J. Dik. *Tests for preference.* 1983.

163 H. Schippers. *Multiple grid methods for equations of the second kind with applications in fluid mechanics.* 1983.

164 F.A. van der Duyn Schouten. *Markov decision processes with continuous time parameter.* 1983.

165 P.C.T. van der Hoeven. *On point processes.* 1983.

166 H.B.M. Jonkers. *Abstraction, specification and implementation techniques, with an application to garbage collection.* 1983.

167 W.H.M. Zijm. *Nonnegative matrices in dynamic programming.* 1983.

168 J.H. Evertse. *Upper bounds for the numbers of solutions of diophantine equations.* 1983.

169 H.R. Bennett, D.J. Lutzer (eds.). *Topology and order structures, part 2.* 1983.

CWI TRACTS

1 D.H.J. Epema. *Surfaces with canonical hyperplane sections.* 1984.

2 J.J. Dijkstra. *Fake topological Hilbert spaces and characterizations of dimension in terms of negligibility.* 1984.

3 A.J. van der Schaft. *System theoretic descriptions of physical systems.* 1984.

4 J. Koene. *Minimal cost flow in processing networks, a primal approach.* 1984.

5 B. Hoogenboom. *Intertwining functions on compact Lie groups.* 1984.

6 A.P.W. Böhm. *Dataflow computation.* 1984.

7 A. Blokhuis. *Few-distance sets.* 1984.