TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Processing networks : introduction and basis structure

Citation for published version (APA):
Koene, J. (1981). *Processing networks : introduction and basis structure*. (Memorandum COSOR; Vol. 8106). Technische Hogeschool Eindhoven.

Document status and date:
Published: 01/01/1981

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 24. Sep. 2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

STATISTICS AND OPERATIONS RESEARCH GROUP

Memorandum COSOR 81-06

Processing networks: Introduction and basis structure

by

J. Koene

Eindhoven, march 1981

The Netherlands

PROCESSING NETWORKS: INTRODUCTION AND BASIS STRUCTURE.

by

J. Koene
Eindhoven University of Technology
Department of Mathematics
P.O. Box 513
5600 MB Eindhoven
the Netherlands

Abstract

A processing network problem is a network flow problem with the following characteristics:

1. conservation of flow in nodes and on arcs.
2. capacity bounds on arcs.
3. there is a non empty subset PN of the node set, such that for each node in PN the flows on the outgoing arcs (or on the incoming arcs) are proportional to each other.

A mathematical formulation of the min cost flow problem in a processing network will be given. The structure of a basis will be discussed. This structure can be used in specifications of the primal simplex algorithm for the min cost flow problem, in analogy with other type of network flow problems (pure, generalized and multicommodity networks).

1. Introduction

Besides the usual requirements of a pure network flow problem - conservation of flow and capacity bounds on arcs - a processing network has an additional property: there is a non empty subset PN of the node set, such that for each node in PN the flows on the outgoing arcs (or on the incoming arcs) are proportional to each other.

The concept of processing networks comes from process industry where refining processes (some commodity splits up in several components in given proportions) and blending processes (several components are blended in given proportions) play an important role. Processing networks can be used to model input-output type of problems [17], but in fact, since any LP-problem can be transformed to a processing network problem [14], processing networks could prove to be a good modelling tool for a large class of practical problems.

A mathematical formulation of the min cost flow problem in a processing network is given in the next chapter.

In network flow problems a basis (in the common sense of linear programming) can be characterized as a subgraph with special structure of the original network. For pure, generalized and multicommodity networks the basis structure is well known and has been exploited in a number of specifications of the primal simplex algorithm [1,3,4,5,7,8,9,10,11,16] yielding great advantages both in solution times and memory requirements. In chapter 3 a summary is given of the basis structure in pure and generalized networks. The basis structure for processing networks will be discussed in chapter 4. A specification of the primal simplex method which exploits this structure is in development.

2. Mathematical formulation

A directed and connected graph G(N,A) is considered. N is the set of nodes, A the set of arcs (i,j), $i,j \in N$. The number of nodes in N is given by n. Each arc (i,j) has capacity bounds ℓ_{ij} and u_{ij}, the flow level is given by

x_{ij}. For each node $i \in N$ the following sets are defined:

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

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

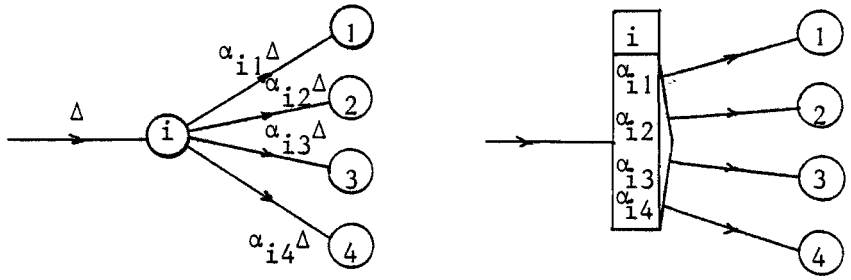
The set N is partitioned into three subsets:

RN : refining nodes

BN : blending nodes

TN : transportation nodes

A *refining node* is a node with two or more outgoing arcs. The flow on each outgoing arc (i,j), $j \in A(i)$ of such a node i is required to be a given fraction α_{ij} ($0 < \alpha_{ij} < 1$) of the total flow entering node i (fig. 1a)



(a) (b)

fig. 1. A refining node

Let r be some node in A(i), then the proportionality demands can be formulated as:

$$(1) \qquad \frac{\alpha_{ij}}{\alpha_{ir}} x_{ir} - x_{ij} = 0 \qquad i \in RN,\ r \in A(i),\ j \in A(i) - \{r\}\ .$$

It is assumed that:

1. $\alpha_{ij} > 0 \qquad j \in A(i)\ ,\ i \in RN$

2. $\sum_{j \in A(i)} \alpha_{ij} = 1 \qquad i \in RN$

Arc (i,r) is called the *representative arc* of processing node i (or process i) because if the flow in (i,r) is known according to (1) flows in all (i,j), $j \in A(i)$ are known.

A *blending node* is a node with two or more incoming arcs. The flow on each incoming arc (j,i), $j \in B(i)$, $i \in BN$ is required to be a given fraction α_{ji} of the total flow leaving node i (fig. 2a).

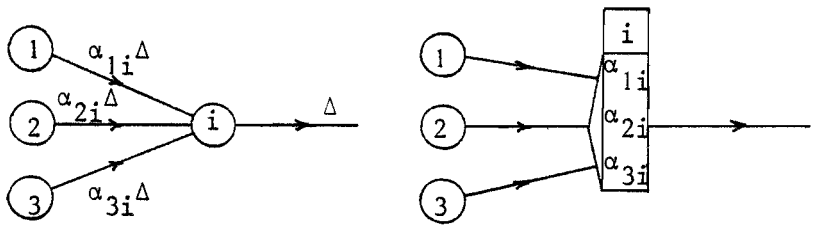


fig. 2. A blending node.

The blending requirements are:

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

in which (r,i) is the representative arc of blending node (process) i. Furthermore $\alpha_{ji} > 0$, $\forall j \in B(i)$, $i \in BN$ and $\sum_{j \in B(i)} \alpha_{ji} = 1$, $\forall i \in BN$.
The set of nodes:

$$PN := \{i \mid i \in RN \cup BN\}$$

is called the set of *processing nodes*. It consists of p nodes of which p_R are refining nodes and p_B are blending nodes. Every node in N - PN is called a *transportation node*. In all $i \in N$ conservation of flow is assumed.

The set of arcs A is partitioned into :

RA: refining arcs

BA: blending arcs

TA: transportation arcs

A *refining arc* is an arc $(i,j) \in A$ in which $i \in RN$, a *blending arc* (j,i) is one in which $i \in BN$. All other arcs are called *transportation arcs*. The set:

$$PA := \{(i,j) | (i,j) \in RA \cup BA\}$$

is called the set of *processing arcs*. With PA(i) will be denoted the set of processing arcs leaving $i \in RN$ or entering $i \in BN$. The number of arcs in PA(i) is given by m_i and is called the *order* of process i. The set N(i) is defined as:

$$N(i) := \{j | (i,j) \cup (j,i) \in PA(i)\} + \{i\}, \quad i \in PN.$$

A directed graph with one or more processing nodes will be called a *processing network*.

Throughout it is assumed that a node can not be a refining node and a blending node at the same time. If some node i has blending requirements on incoming arcs and refining requirements on outgoing arcs this can be established by introduction of an extra node i' and a transportation arc (i,i'), taking all outgoing arcs of i as outgoing arcs of i'.
Similarly it is assumed that some arc (i,j) can not be both a refining arc of node i and a blending arc of node j. The remedy in this situation would be to introduce an additional transportation node i' and replace arc (i,j) by arcs (i,i') and (i',j).
In drawing diagrams of processing networks it is convenient to distinguish refining nodes, blending nodes and transportation nodes. Refining nodes and blending nodes will be presented as in fig. 1b and 2b, a transportation node is given by a simple circle.

The minimal cost flow problem can be formulated as follows:

$$(3)\quad \text{minimize} \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$(4)\quad \sum_j x_{ij} - \sum_j x_{ji} = b_i \qquad i \in N$$

$$(5)\quad \frac{\alpha_{ij}}{\alpha_{ir}} x_{ir} - x_{ij} = 0 \qquad i \in RN,\ r \in A(i),\ j \in A(i) - \{r\}$$

$$(6)\quad \frac{\alpha_{ji}}{\alpha_{ri}} x_{ri} - x_{ji} = 0 \qquad i \in BN,\ r \in B(i),\ j \in B(i) - \{r\}$$

$$(7)\quad \ell_{ij} \le x_{ij} \le u_{ij} \qquad (i,j) \in A\ .$$

Constraints (4) are the conservation of flow equations, in which b_i denotes the (external) demand or supply in node i, (5) and (6) are the refining and blending requirements; capacity bounds are given by (7).

The coefficient matrix of the left hand sides of (4) - (6) is called the technology matrix T. The structure of this matrix is illustrated in fig. 3.

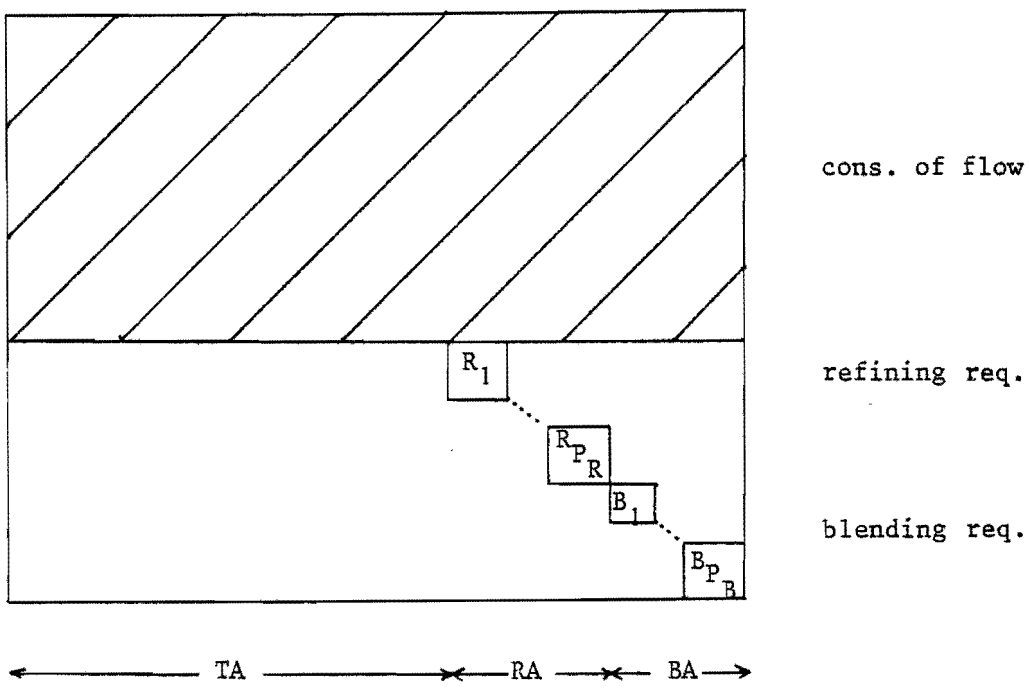


fig. 3. Structure of technology matrix.

In this matrix R_i $(i = 1,\ldots,P_R)$ and B_i $(i = 1,\ldots,P_B)$ are $(m_i - 1) \times m_i$ matrices with the following structure:

$$R_i = \begin{bmatrix} \frac{\alpha_{ij}^{\,1}}{\alpha_{ir}} & -1 & & \\ \frac{\alpha_{ij}^{\,2}}{\alpha_{ir}} & & -1 & \\ \vdots & & & \ddots \\ & & & -1 \end{bmatrix} \qquad B_i = \begin{bmatrix} \frac{\alpha_{j}^{\,1}{}_{i}}{\alpha_{ri}} & -1 & & \\ \frac{\alpha_{j}^{\,2}{}_{i}}{\alpha_{ri}} & & -1 & \\ \vdots & & & \ddots \\ & & & -1 \end{bmatrix}$$

It is possible to give a more compact formulation for the min cost flow problem. This formulation is achieved by substituting the expressions for x_{ij} in (5) and x_{ji} in (6) into (4). Doing so each refining process and each blending process is represented by a single column in the technology matrix T^*. T^* has n rows and each row i of T^* can be identified by node i of the network. In this setting it seems natural to say that each column (i,j) of T^* describes some kind of process. Three types of processes (columns) can be distinguished:

1. *transportation processes*. Column (i,j) has a +1 in row i and a - 1 in row j. All other elements in the column are zero.
2. *refining processes*. The elements in column (i,j) are $1/\alpha_{ij}$ in row i and $-\alpha_{ik}/\alpha_{ij}$ in row k for every $k \in A(i)$. All other elements are zero.
3. *blending processes*. There is a $-1/\alpha_{ji}$ in row i of column (j,i) and α_{ki}/α_{ji} in row k, $k \in B(i)$. Other elements in the column are zero.

It is observed that matrix T^* has the following qualities:

- the column sum of each column in T^* is zero.
- if there are more than one negative (positive) elements in some column of T^* there is only one positive (negative) element.

Note that the columns can be scaled such that the only positive element is +1 (or the only negative element is -1). As an illustration for the processing network of fig. 4 the two discussed types of technology matrices are presented in fig. 5 and 6.

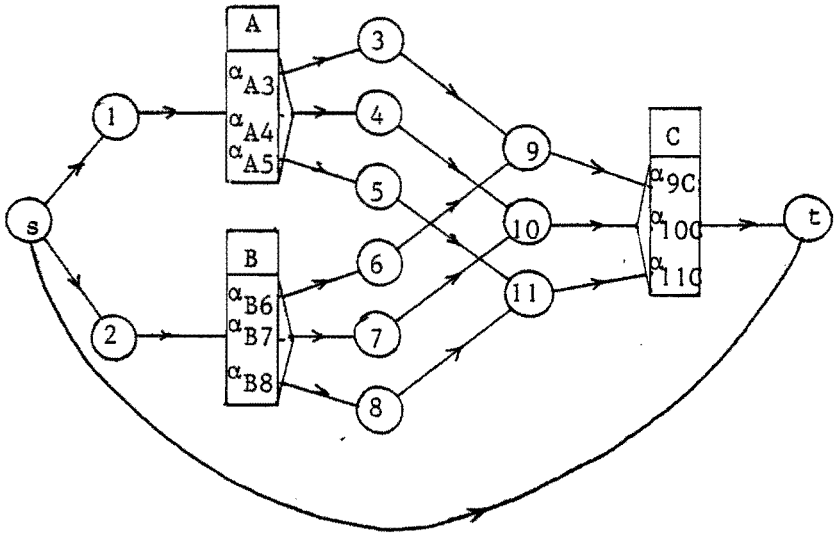


fig. 4. Example of processing network

3. Basis structure in pure and generalized networks

For pure and generalized networks the basis structure is well described in graphical terms [2,6,7,9,16]. A summary is given here.

A basis in a pure network is a (rooted) spanning tree. The reverse is true too. Every spanning tree is a (not necessarily feasible) basis [2].

In a generalized network the basis structure is somewhat more complex: A quasi-tree is a tree with one additional arc. This arc may be a self-loop. To put it in other words: a quasi-tree is a connected graph with exactly one cycle. A basis in a generalized network is a set of quasi-trees, such that every node of the network is contained in some quasi-tree

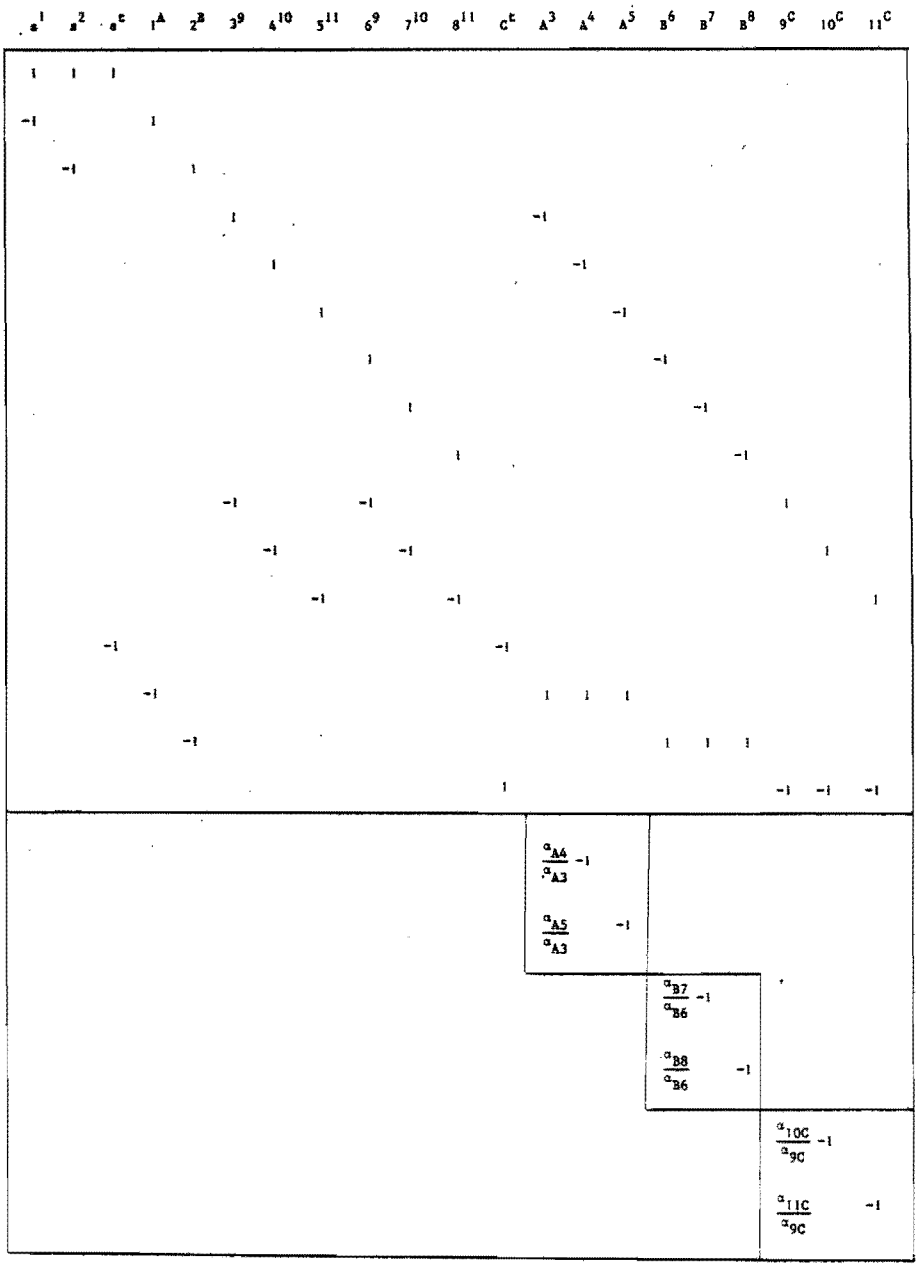


fig. 5. Technology matrix, first version.

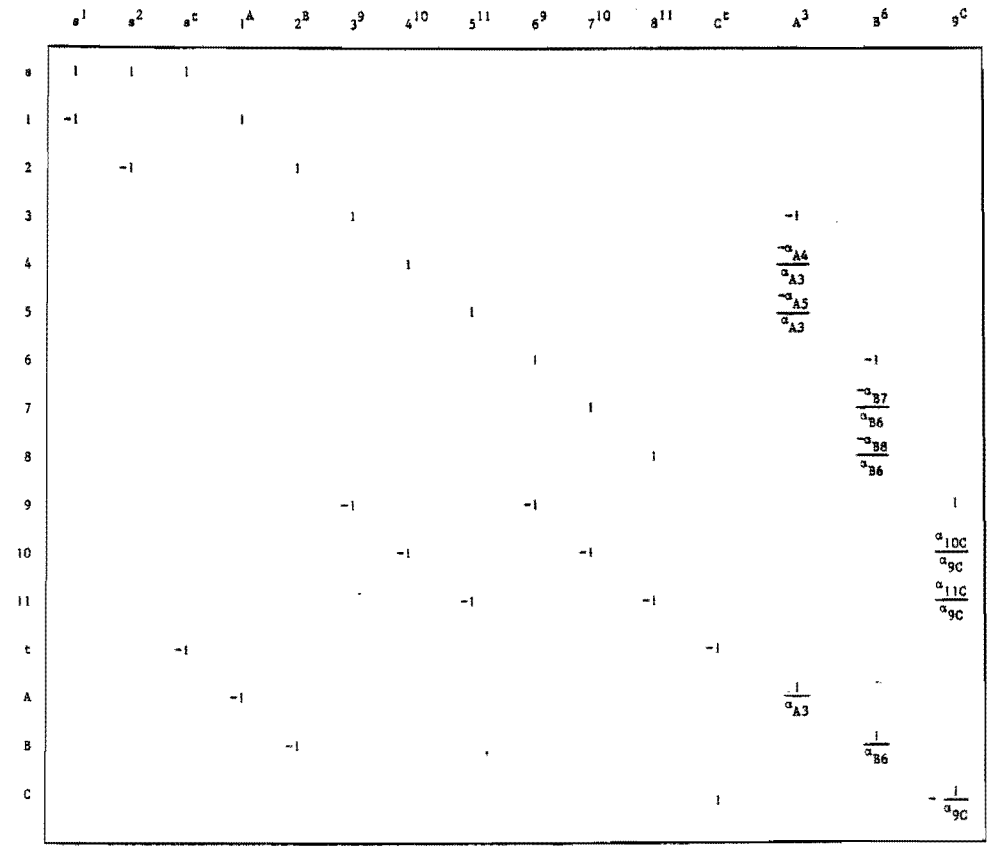


fig. 6. Technology matrix, compact version, representative arcs A3,B6,9C.

(a so called forest of quasi-trees). The reverse is in general not true. Suppose a quasi-tree has a cycle C consisting of at least two arcs and nodes (so it is not a self-loop). Let the product of gains of all clockwise oriented arcs in C be given by k_1 ($k_1 = 1$ if no such arcs are present in C) and let the product of all gains of all counter clockwise oriented arcs be given by k_2 ($k_2 = 1$ if none are present). A forest of quasi-trees is a basis if and only if in the cycles of the quasi-trees consisting of two or more arcs: $k_1 \neq k_2$. The proof can easily be established by considering theorem 1 (iv) and (v) of [18]. The conclusion can be drawn that the basis structure in a generalized network does not only depend on the topology of the network but also on the values of the gains.

4. Basis structure in processing networks

For the purpose of describing a basis in terms of the processing network formulation (3) - (7) is most adequate. This formulation will be used here. However the results and the concepts defined in the sequel can easily be adapted to suit the compact way of describing.

A subgraph of G(N,A) whose arcs can be identified with a basis will be called a *basis graph*.

Before discussing the basis structure one important assumption is made. Consider the graph $G^*(N,TA)$ consisting of all nodes $i \in N$ and all transportation arcs. It is assumed that $G^*(N,TA)$ is connected. This assumption makes it easier to describe a basis, but is not really restrictive. If $G^*(N,TA)$ would consist of several connected components $C_1, C_2, \ldots, C_k$ it can be made connected by introducing "bridge" arcs (s,j), where s is some node in C_1, j some node in C_i ($i = 2,\ldots,k$) and $\ell_{sj} = 0$, $u_{sj} = 0$, $c_{sj} = 0$.

4.1. Some properties

A cycle consisting of only transportation arcs is called a *transportation cycle*. A first observation follows directly from the theory of pure networks:

Lemma 1: The basis graph can not contain any transportation cycle

Proof: Suppose on the contrary there is such a transportation cycle. Then dependency between columns in the coëfficient matrix corresponding to the arcs in the cycle is easily established, see for instance [2]. □

The main reason for assuming $G^*(N,TA)$ connected is that it renders an easy expression for the number of arcs in a basis graph.

Lemma 2: The number of arcs in a basis graph is

$$(n - 1) + \sum_{i \in PN} (m_i - 1).$$

Proof: The technology matrix T consists of $n + \sum_{i \in PN} (m_i - 1)$ rows and at least as many columns since $G^*(N,TA)$ is connected. Adding up the first n rows of T leaves the zero row, so the rank of T is smaller than or equal to $(n - 1) + \sum_{i \in PN} (m_i - 1)$. It is easy to see that the rank must be equal to this number by constructing a spanning tree in $G^*(N,TA)$ and taking all non representative arcs of the refining and blending processes.

The matrix which results can be written as $P = \begin{bmatrix} B & C \\ O & D \end{bmatrix}$, where B is an $(n - 1) * (n - 1)$ matrix which corresponds to a spanning tree and D is a non singular diagonal matrix of order $\sum_{i \in PN} (m_i - 1)$. Since both B and D are non singular P is non singular and thus represents a basis. □

An other straight-forward result is:

Lemma 3: Either all m_i or $(m_i - 1)$ arcs of each set PA(i), $i \in PN$ belong to the basis graph.

Proof: Suppose for some set PA(i), $i \in PN$ two or more arcs are omitted. If (at least) the representative arc and some non representative arc would be left out a zero row would appear in the last $\sum_{i \in PN} (m_i - 1)$ rows. If (at least) two non representative arcs are omitted two rows of the last $\sum_{i \in PN} (m_i - 1)$ rows would be linearly dependent.

Both cases are in contradiction with lemma 2. □

At this stage several concepts are introduced. A processing node i (or process i) is called *active* if all m_i arcs of the set PA(i) are contained in the basis graph. Otherwise (only $(m_i - 1)$ arcs are present in the basis graph) it is called *inactive*. If we would consider the basis graph in which all processing arcs contained in it are left out, this graph would contain several connected components consisting of only transportation arcs. Such a connected component is called a *transportation tree* since it can not contain any cycle. A transportation tree could consist of a single node (including processing nodes).

The next lemma gives a relation between the number of transportation trees and the number of active processes in the basis.

Lemma 4: If there are $(q + 1)$ transportation trees in the basis graph $(q \geq o)$ then there must be q active processes and the other way round.

Proof: Every node $i \in N$ belongs to some transportation tree. If the number of transportation trees is $(q + 1)$, $q \geq 0$ these trees contain $n - (q + 1)$ transportation arcs. From Lemma's 2 and 3 it follows that the number of active processes must be q. The second part of the proof is obtained by reversing the argument. □

A more hidden property of a basis is the following. Denote by APN the set of active processing nodes. Suppose APN $\neq \phi$ and let S be a non empty subset of APN. The number of elements of S is $|S|$. Further let T(S) be the set of transportation trees containing at least one node of $N(i)$, $i \in S$. The number of those trees is given by $|T(S)|$.

Lemma 5: $|T(S)| \geq |S| + 1$.

Proof: Suppose that the transportation trees in T(S) contain in total a nodes. This means that the set T(S) contains $a - |T(S)|$ arcs. The set $T(S) \cup \{ \bigcup_{i \in S} PA(i) \}$ contains $a - |T(S)| + |S| + \sum_{i \in S} (m_i - 1)$ arcs. In matrix terms these arcs have entries unequal to zero in exactly $a + \sum_{i \in S} (m_i - 1)$ rows. Adding up the a rows corresponding to the conservation of flow equations leaves the zero row. Given the fact that the columns in a basis matrix are linearly independent this means that :

$$a - |T(S)| + |S| + \sum_{i \in S} (m_i - 1) \leq a - 1 + \sum_{i \in S} (m_i - 1)$$

or

$$|T(S)| \geq |S| + 1 .$$ □

Equality in lemma 5 holds in any case for S = APN according to lemma 4, but there can be real subsets of APN for which the equality holds too. A consequence of lemma's 4 and 5 is that every transportation tree is incident to at least one active processing node, that is to say every transportation tree contains some node $j \in N(i)$, $i \in$ APN. Note worthy is that lemma 5 also applies to pure networks in the following way. If in a pure network S is defined as a subset of basis arcs and T(S) the set

of nodes incident to those arcs this is evident. Here lemma 5 can be formulated as: there can not be a cycle in a basis graph.

Suppose that a sub graph of G(N,A) satisfies lemma's 1 - 5 then it is not said that it characterizes a basis. This is illustrated in fig. 7.

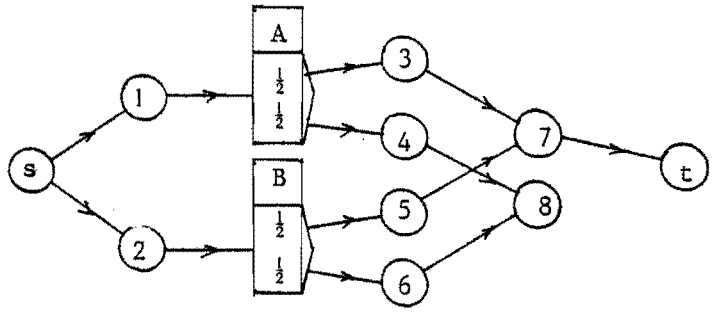


fig. 7. Not a basis.

Let a_{ij} denote the column in the technology matrix corresponding to arc (i,j). It is easily checked that:

$$2a_{s1} - 2a_{s2} + 2a_{1A} - 2a_{2B} + a_{A3} + a_{A4} - a_{B5} - a_{B6} +$$

$$+ a_{37} - a_{57} + a_{48} - a_{68} = 0$$

showing that these columns are linearly dependent. As in the case of generalized networks the basis does not only depend on the topology of the network but also on the coëfficients α_{ij}.

For practical purposes lemma 5 is inadequately formulated. In the next section it will be shown with the help of lemma 5 that the representative arcs of active processes can be chosen in a convenient way.

4.2. Representative arcs of active processes

The main point to be made in this section is:

Lemma 6: The representative arcs of active processes can be chosen in such a way that the arcs contained in the transportation trees plus these representative arcs form a spanning tree in G(N,A).
This spanning tree will be called a *representative spanning tree*.

Proof: Suppose that a basis contains (q + 1) transportation trees $T_1, T_2, \ldots, T_{q+1}$ ($q \geq 0$). If $q = 0$ there is a spanning tree of transportation arcs. Consider the case that $q \geq 1$ and let the representative arcs of the active processes be chosen in an arbitrary way.

It is helpful to consider the following derivated graph: nodes $1,2,\ldots,q+1$ (node i corresponds to transportation tree T_i) and edges (unordered pairs) (u,v) for every representative arc of active processes which has endpoints in T_u and T_v. Notice that one of the nodes incident to a representative arc is the processing node attached to it, which, given a basis belongs to one and the same transportation tree. If it can be proved that the representative arcs of active processes can be chosen in such a way that the derivated graph is connected then also the graph consisting of the transportation trees plus these representative arcs is connected and thus a representative spanning tree since it contains n nodes and (n - 1) arcs.

Suppose that the derivated graph is not connected and consists of $k \geq 2$ connected components. This means that at least one component must contain a cycle. Consider such a cycle C_1 consisting of nodes $1,2,\ldots,\ell$ (numbered so only for convenience) and edges $(1,2),(2,3),\ldots,(\ell-1,\ell),(\ell,1)$. It will be proved that the number of connected components can be reduced to (k - 1) as long as $k \geq 2$. By induction it follows that in the end a connected derivated graph should result.

The ℓ edges in C_1 correspond to ℓ active processes which have "entrances" in at least $(\ell + 1)$ transportation trees according to lemma 5. So there must be at least one node $\notin C_1$, say $(\ell + 1)$ in the derivated graph, such that edge $(i,i + 1)$ can be replaced by $(i,\ell + 1)$. If node $(\ell + 1)$ belongs to an other connected component the number of components is reduced by one.

Otherwise (ℓ + 1) belongs to the same connected component as cycle C_1; when the replacement is carried out a new cycle C_2 is formed with at least one arc which was not contained in C_1. The ℓ edges of C_1 plus the new one(s) correspond to as many active processes. Again at least one of those edges can be chosen differently, such that one endpoint $\notin C_1 \cup C_2$ (Lemma 5). This process can be repeated. In the end there must be some edge which can be replaced by one with one endpoint in an other component. If this edge is contained in some cycle, which can always be (re)established, the number of connected components is then reduced by one. □

An illustration of the concept "derivated graph" is given in fig. 8. The numbers in the nodes say to which transportation tree this node belongs. The active processes are given by A,B and C; only the chosen representative arcs of these processes are drawn. The structure of the derivated graph is given below the basis graph.

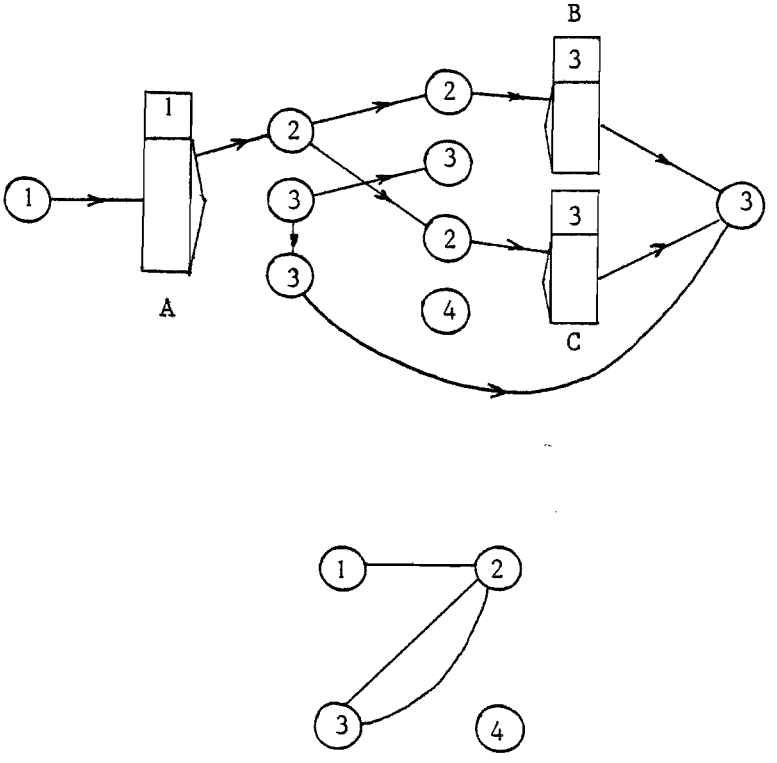


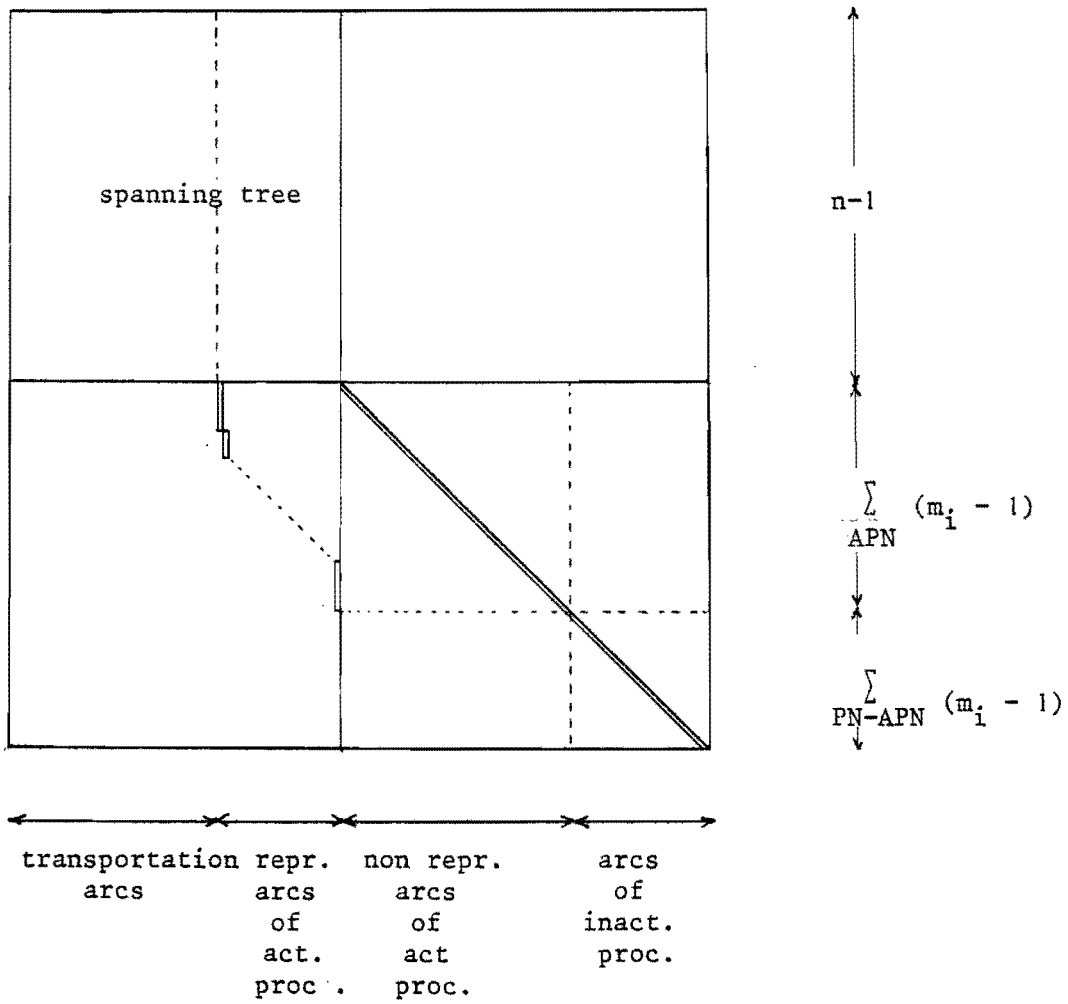
fig. 8, Part of a basis graph with derivated graph.

5. Conclusion

The lemma's in the previous chapter prove:

Theorem: A basis graph in a processing network is a spanning tree consisting of transportation arcs and representative arcs of active processes plus $(m_i - 1)$ additional processing arcs for each set PA(i), $i \in PN$.

In matrix terms the basis structure can be summarized as in the following picture:



References

1. *Barr. R, J. Elam, F. Glover and D. Klingman,* A Network Alternating Path Basis Algorithm for Transshipment Problems, Lecture Notes in Economics and Mathematical Systems, No. 174, Extremal Methods and System Analysis, A. Fiacco and K. Kortanek Eds.,Springer Verlag, Berlin 1980.

2. *Bazaraa. M.S. and J.J. Jarvis,* Linear Programming and Network Flows, John Wiley and Sons, New York, 1977.

3. *Bradley. G.H., G.G. Brown and G.W. Graves,* Design and Implementation of Large Scale Primal Transshipment Algorithms, Management Science 24 (1977) 1-34.

4. *Charnes. A., F. Glover, D. Karney, D. Klingman and J. Stutz,* Past, Present and Future Development, Computational Efficiency and Practical Use of Large Scale Transportation and Transshipment Computer Codes, Comput. and Operations Research 2 (1975) 71-81.

5. *Cunningham. W.H.,* A Network Simplex Method, Math. Programming 11 (1976), 105-116.

6. *Dantzig. G.B.,* Linear Programming and Extensions, Princeton University Press, Princeton, N.J, 1963.

7. *Elam. J, F. Glover and D. Klingman,* A Strongly Convergent Primal Simplex Algorithm for Generalized Networks, Math. of Oper. Res 4 (1979) 39-59.

8. *Glover. F, D. Karney and D. Klingman,* Implementation and Computational Comparison of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Flow Network Problems, Networks 4 (1974) 191-212.

9. *Glover. F., D. Klingman and J. Stutz,* Extensions of the Augmented Predecessor Index Method to Generalized Network Problems, Transportation Science 7 (1973) 377-384.

10. *Hartman. J.K. and L.S. Lasdon,* A Generalized Upperbounding Algorithm for Multicommodity Network Flow Problems, Networks 1 (1972) 333-354.

11. *Helgason. R.V. and J.L. Kennington,* A Product Form Representation of the Inverse of a Multicommodity Cycle Matrix, Networks 7 (1977) 297-322.

12. *Johnson. E.L.,* Networks and Basic Solutions, Operations Research 14 (1966) 619-623.

13. *Koene. J.,* Maximal Flow through a Processing Network with the Source as the only Processing Node, Memorandum COSOR 80-02, Revised version november 1980, Eindhoven University of Technology.

14. *Koene. J.,* Formulating Linear Programming Problems as Processing Network Problems, Eindhoven University of Technology, Eindhoven, to appear as Memorandum COSOR.

15. *Mayeda. W.,* Maximal Flow under Controlled Edge Flows, Proceedings 1968 Int. Conf. Communs, Philadelphia, Pennsylvania, june 1968.

16. *Maurras. J.,* Optimization of the Flow through Networks with Gains, Math. Programming 3 (1972) 135-144.

17. *Schaefer. A.,* Netze mit Verteilungsfactoren, Verlag Anton Hain, Meisenheim am Glan, 1978.

18. *Truemper. K.,* An Efficient Scaling Procedure for Gain Networks, Networks 6 (1976) 151-159.