Minimum Cost Sizing of Rearrangeable Networks with Multi-period Demands

Patrick Rioux

A Thesis

in

The Department

of

Electrical and Computer Engineering

Submitted in Partial Fulfillment of the Requirements
for the degree of Doctor of Philosophy at
Concordia University
Montréal, Québec, Canada

January 1988

ISBN 0-315-49109-4

ABSTRACT

Minimum Cost Sizing of
Rearrangeable Networks with
Multi-period Demands

Patrick Rioux Ph.D.
Concordia University, 1988

In the planning of dynamic architecture networks, considerations of dimensioning of the links lead to a multi-commodity, multi-period flow problem on the underlying graph. The fact that all links of the network are non-directed greatly affects the formulation of this multi-period problem. In this thesis, a formulation based on the *arc-chain* representation is presented. This formulation reveals an especially interesting structure which can be exploited in two ways. First, the *generalized upper bounding* (GUB) technique is used to take advantage of the multi-commodity nature of the problem. GUB reduces the size of the problem to that of a working basis. Then, the multi-period character of the problem is exploited by a decomposed triangular factorization of the working basis. In each step of the algorithm, the problem is divided into as many subproblems as there are periods, where each subproblem has a size equivalent to the size of a single period problem. A column generation algorithm designed in order to avoid the explicit representation of the columns is also presented.

ACKNOWLEDGMENTS

I sincerely thank my supervisors Dr. K. Thulasiraman of Concordia University and Dr. B. Smith of Ecole Polytechnique for their invaluable guidance, patience, and support during the writing of this thesis.

I thank Dr. J.F. Hayes, Chairman of the Department of Electrical and Computer Engineering at Concordia University, for his support and collaboration in coordinating the joint BNR-Concordia project.

A special thanks to Dr. P. Blondeau, Professor at Ecole Polytechnique, who supervised the thesis during the early stages, and for giving me the opportunity to join Bell-Northern Research in the graduate students formation program.

I express my gratitude to Bell-Northern Research for believing in the value of an industry-university collaboration.

I would also like to thank the Natural Sciences and Engineering Research Council of Canada for the generous financial support received for the period from January 1984 to January 1988, and Concordia University for the University Graduate Fellowship awarded to me for the academic year 1984-1985.

A mes parents,

Anne-Marie et Charles-Albert.

TABLE OF CONTENTS

LIST OF NOTATIONS

A'	Arc-chain matrix representing the network.
A^h	Arc-chain matrix after scaling by the demand vector of period h.
$\bar{a}^{leav}$	Column leaving B.
$\bar{a}^{ent}$	Column entering B.
a^{ent}	Column entering WB: first HM elements of $\hat{a}^{ent}$ - KEY($\hat{a}^{ent}$)
$\bar{a}^j$	Column of the coefficient matrix.
a^j	First HM elements of $\hat{a}^j$ - KEY($\hat{a}^j$)
B	Basis; square matrix of size $HM + HK$.
$\hat{B}$	Last HM columns of the first HM rows of the basis B.
C_m	Capacity variables.
C	Last HM columns of the last HK rows of the basis B.
C^h	Sub-matrix of C associated with period h.
d^h	Demands for period h.
E	Demand constraints matrix
f^h	Flow variables for period h (fraction of the demand satisfied by each of the L routes).
f^h_{ml}	Flow on arc m due to the flow variable corresponding to route l at period h.
H	Number of periods.
$\widehat{LP}^h$	Factorization matrix associated with Q^h.
LP^h	First q^h rows of $\widehat{LP}^h$.
LP^X	Factorization matrix associated with X.
K	First HK columns of the first HM rows of the basis B.
K^h	Sub-matrix of K associated with period h.

K^{ent}	Sub-matrix of K associated with the period of the entering column.
K^{leav}	Sub matrix of K associated with the period of the leaving column.
K_k^h	Column of K^h representing OD-pair k.
K	Number of OD-pairs in the network $(= 0.5 \times N(N - 1))$.
KEY(a^j)	Key column associated with route a^j.
L	Number of routes in the network.
M	Number of arcs in the network.
N	Number of nodes in the network.
OD(a^j)	OD-pair of route a^j.
PER(a^j)	Period of route a^j.
Q^h	Sub-matrix of WB associated with period h.
q^h	Number of columns in Q^h.
S	Set of the columns of B having $OD(a^{leav})$ and PER(a^{leav}).
S^h	Slack variables associated with period h.
S_m^h	Slack variable for period h corresponding to arc m.
T	GUB transformation matrix for the basis.
U	The near upper-triangular matrix resulting from the decomposed factorization of WB.
$\widehat{U}^h$	Matrix resulting from the factorization of Q^h.
U^h	First q^h rows of $\widehat{U}^h$.
U^X	Matrix resulting from the factorization of X.
V	Permutation matrix allowing for the decomposition of WB.
X	Right bottom corner matrix of U.
x^B	Basic variables.
x_{ent}^B	Variable entering the basis B.
x^{ent}	Column entering X.

x^{leav}	Column leaving X.
WB	Working basis, resulting from the application of the *generalized upper-bounding* technique to $\hat{B}$.
$\$$	Cost vector; unit costs of the arcs.
$\$^{B}$	Cost vector; unit costs of the variables associated with the columns of B.
$\$^{WB}$	Cost vector; unit costs of the variables associated with the columns of WB.
π	Dual variables associated with the capacity constraints.
$\pi_{t'}$	Transformed dual variables.
π^{h}	Dual variables associated with the capacity constraints of period h.
μ	Dual variables associated with the demand constraints.
μ^{h}	Dual variables associated with the demand constraints of period h.
σ	Representation of the $\hat{a}^{ent}$ in terms of the columns of B.
σ_1	First HK elements of σ after the transformation T.
σ_2	Last HK elements of σ after the transformation T.

LIST OF FIGURES

Chapter 1 Introduction

Network planning methods are now increasingly applied to a great variety of problems. Today's society depends heavily on reliable telecommunication networks for voice and data transmission as well as on transportation networks for passengers, freight or others goods.

The advent of powerful and specialized computers has turned the quiet evolution of telecommunication networks into a race to provide new and better services [1]. The North American telephone network, with more than 100 million access lines [2] and with its increasing diversity of services, is among the most sophisticated networks in the world. The introduction of new technologies, new services or new concepts in such a large network must be carefully studied taking into account the average lifetime of the equipment (more than 20 years in some cases) and the cost involved.

Prior to digitization in the 1960's, planners were simply provisioning for the expected growth given by more or less accurate demand forecasts. Cost minimization of the growth based on economic methods such as PWAC (Present Worth of Annual Charges) was the chief concern.

During the last 20 years, proliferating technological developments have made possible a number of new concepts of operation providing new types of services to customers. Planners have been unable to predict their effects on the network.

In fact, they have never kept pace with the impressive evolution in concepts and services required. The short time horizon type of planning advocated until quite recently for *circuit-switched networks* appeared to be no longer valid.

At the same time, planners of *computer communication networks* [3] outdistanced technological improvements. New protocols and concepts succeded one another at an impressive rate, only moderated by the computational power available. Many reasons favoured planners of *packet-switched* types of network. First, the networks considered were much smaller than the telephone networks. Very often, they were plannning without any, or with very little, restrictions on the structure, whereas *circuit-switched* network planners had to deal with the evolution of a huge, existing infrastructure. Obviously, it was impossible to consider removing the current network and starting afresh with a new network and new concepts. Finally *computer communication networks* have been planned as a function of the upcoming technologies on a long term basis. Until recently, telephone plannners who were unable to follow the technological development had to work on a short term basis. Most of the efforts were directed towards development of new and more powerful equipment. There was very little cooperation with planners to create long term objectives and to define the equipment necessary for the evolution of the network in the most promising direction.

The first organized attempt at planning has become necessary with the current trend to *integrate services* [4] which are completly different in nature all in one network, the Integrated Service Digital Network (ISDN). This integration in a single network is not an easy task if we look at the list of services considered:

circuit-switched facilities for services requiring a dedicated circuit for an undetermined period of time. Voice communication is the best known example in this category.

Packet-switched facilities for communication of digital signals with intermittent transmission. An interactive session between a terminal and a mainframe is representative of this type of services.

Special services capabilities to satisfy all other types of demands. For example, non-switched networks in which circuits are generally dedicated for long periods of time.

Common Channel Signalling (CCS) facilities utilized by the control unit to govern all the communication among the equipment in the network.

The ISDN network [5] also needs the flexibility of making possible the introduction of services required in the future with a minimum impact on the network itself.

At present, the CCITT (International Telegraph and Telephone Consultative Committee) is responsible for defining the standards and protocols necessary for an organized evolution of the ISDN concept. Based on the CCITT reports [6,7,8] the first field trials started last year and will continue until 1988. The first totally operational ISDN network is not expected before early 1990.

A key problem with Integrated Services Networks is the non-coincidence of the demands. Non coincident demands occur in presence of demands at different periods in time and when all links are not dominated by the demand of the same period. There are many reasons for non-coincidence: traffic patterns which are different during business hours compared to evenings ; time zones (there is a time difference of 3 hours between peak traffic in Montréal and Vancouver), etc. The presence of non-coincident demands may become an important problemwhen the network does not have sufficient flexibility to react accordingly. The solution to this problem is to use a network with *dynamic architecture*. By Dynamic Network Architecture (DNA) we mean the ability of a network to allocate the spare capacity where it is needed. Traditional telephone networks are currently unable to adapt their architecture to the state of the network.

Another important factor is that a cost effective implementation of ISDN relies on the principle of *dynamic architecture*. It is inconceivable to implement all the capabilities of ISDN at every node. Using DNA, it will be possible to reconfigure the network in real time and give the impression of being directly connected to all services from every node in the network.

The prime objective of DNA is the dynamic allocation of resources and services to customers. The second goal is the survivability of the network. It does not have the spectacular effect of the first goal but its importance is far from negligeable. The surviviability is the ability to maintain services under link or node failures. Obviously, concepts such as DNA and *rearrangeable networks* are important steps ahead in the consideration of survivability problems.

What made possible the concept of *dynamic architecture* is a new piece of equipment called DXCS (Digital Cross Connect System) or *slow-switches*. The DXCS can rearrange the interconnection of the circuits at its ports via remote control command [9]. It enables reconfiguring the circuit pattern of the network under the command of a central control unit. The presence of DXCS in the network and the concept of *dynamic architecture* are relatively new facts for planners. No planning methods for networks using DNA are known so far. A few heuristic algorithms have been published on multi-commodity,multi-period sizing and routing, but none of them looks precisely at our problem. Most of them are direct adaptations of planning methods currently used for *circuit-switched* or *packet-switched* networks. At present, telephone companies in their field trials are systematically oversizing the network until an adequate method is made available.

The sizing of a DNA network with optimal routing is a minimum cost multi-commodity,multi-period flow problem on a non-directed graph. The multi commodity aspect arises because of the presence of a different demand for each pair of nodes. We must mention that for a given period there is only one type of commodity per pair of nodes. This last observation and consideration of the non-directed characteristic of the network permit the use of the arc-chain formulation presented in Chapter 2. Demands are given in numbers of circuits required between each pair of nodes for each of several periods. The reason for a demand in circuits instead of traffic units (Erlang) is the presence of multiple services on the same network. Demands for many types of services in ISDN are given in circuits whereas the demands for voice communications are in traffic units. The best method to standardize the demands for all types of services is to convert every demand into circuit requirements.

The key difference between the problem addressed here and most of the work done in the area of network planning is the presence of non-coincidence in the demands. Though only a few papers have considered the joint problem of multi-commodity flows with multi-period demands, extensive research has been done on the multi-commodity flow problem for a single period [10]. The case of directed graphs is especially well covered in the literature [11,12, 13,14,15,16.17,18]. Aronson and Chen [19,20] considered a multi-period problem on a directed network. But they were concerned with the minimal cost routing rather than minimal cost sizing.

Our formulation of the multi-commodity, multi-period problem is based on the arc-chain representation. Only a few authors have published using the arc-chain representation. Most of them [21,22,23] have based their studies on the *Dantzig-Wolfe Decomposition* algorithm [24] which is constructed to consider problems having a set of general constraints (master problem) and a *block diagonal* set of constraints (the subproblems) . In 1967, Dantzig and Van Slyke [25] published a new algorithm called *Generalized Upper Bounding* (GUB). This algorithm is designed to consider problems where the *block diagonal* constraints are linear convex combinations. As we shall see, this algorithm is perfectly suitable for our formulation and it will be the basis for the algortithm developed in this thesis.

The multi-period aspects of the problem have not been studied as thouroughly as the multi-commodity aspects during the last 20 years. When considering the non-coincidence of demands. planners were mainly interested in routing techniques. Sizing methods were simple extensions of the algorithms used for routing. A paper published in 1971 by Y. Rapp[26] was the first significant effort in the field of network planning dealing with non-coincident busy hours. At that time, the ISDN and DNA concepts were unknown. Considerations of the multi period aspect were restricted to dynamic routing schemes because of the underlying static network. The solution to the problem of non-coincidence appeared to be the use of dynamic routing algorithms [27.28,29, 30,31]. The idea was good for a *circuit-switched* network almost exclusively dedicated to voice communication. The reality is now totally different. It would be impossible to implement a dynamic routing scheme sufficiently powerful to support ISDN. We need a network which is capable of operating on much broader objectives than *dynamic routing* of calls.

Consideration of circuit requirements instead of traffic demands for a multi period network was discussed first by Zadeh [32] and Yaged [33,34,35] in the early 70's. The multi-period aspect studied is the growth over time of the network, given dynamic routing capabilities and circuit demands for the next several years. Although we are concerned with non-coincidence, which is a totally different element of the multi-period problem, those papers are important in the sense that they were the first to examine circuit requirements in a multi-period context.

The mention of *rearrangeable networks* and *slow-switches* in the literarture is recent. Most of the papers on the subject address only the simpler parts of the problem. In the case of *slow-switches* (DXCS), the possibilities enabled by their use in the network are not fully known, and consequently very little information on the subject has been made public by telephone companies.

Rearrangeable networks have not recieved all the attention expected precisely because of the lack of knowledge outside the private sector on the equipment available to support the concept. Akiyama [36] in 1979 and Winnicki and Paczinski [37] in 1980 proposed similar rearrangement procedures. The performances of rearrangement algorithms are compared in a report by M. Larocque [38] published in 1985. In both cases, the prime interest was the rearrangement and not the sizing of the network. It is possible to derive a sizing method from a routing algorithm but the quality of the result is not always good. In the case of rearrangeable networks, the configuration will not be rearranged often, the process is sufficiently slow to allow the time for an optimal routing. The question that remains is to find the minimal cost network that satisfies the demand of each period.

It would be impossible to make the last assumption at all levels in the network. For instance, it is necessary to take into account the type of routing (hierarchical or dynamic) to determine the number of circuits required to respect a traffic demand. The routing of calls has to be done so fast that we cannot assume optimal routing. Different approaches to this problem with traffic demands and hierarchical routing are given in a paper by A. Girard [39].

One approach in an unpublished paper by K. Kortanec and G. Polak [40,41] has attracted our attention. The authors use a formulation very close to the formulation developed in this thesis. The difference is due to the type of demands. Because of a demand in traffic units they must consider the effects of the routing. The extra constraints generated prevent them from using the GUB method to solve the problem. The *block diagonal* part of the constraints, which is composed of linear convex combinations in our formulation, has a general *block diagonal* form in their formulation. This generalization of the constraints forces the use of a less specialized technique called *Dantzig-Wolfe Decomposition* technique [42,43].

We present in the following seven chapters a method based on the Revised Simplex Method (RSM) to find the minimum cost rearrangeagble undirected network with multi-period demands. The problem presents the multi-commodity aspect as well as the multi-period aspect. The combined effect of the presence of both aspects results in a problem having a large number of constraints and variables.

In Chapter 2, we address the problem of finding an adequate mathematical formulation for the problem. We discuss a formulation based on the arc-chain representation of the network. This formulation presents many advantages. First, the number of constraints is relatively small and a large part of these constraints will be considered implicitly by the Generalized Upper Bounding technique. A second advantage of the formulation is the structure obtained. The structure will allow decomposing the basis by period and convert the problem into the equivalent of as many one period problems as there are periods.

In Chapter 3, we review the Revised Simplex Method and the Generalized Upper Bounding technique. This permits us to introduce the notation for the following chapters and also to present some of the properties of the different steps of the RSM. Then we show how the GUB technique considers implictly the demand constraints. Finally, each step of the GUB is briefly described.

The structure of the problem is discussed in more detail in Chapter 4. The effect of the multi commodity and multi-period aspects on the structure of the basis is shown. Two subjects closely related to the structure of the basis are presented. First, the problem of finding an initial basic solution is solved. Then we prove how the structure of the working basis (WB) can be preserved by keeping certain variables (capacity variables) in WB.

In Chapter 5, we present the decomposed factorization procedure where we start with an independent factorization of the matrices associated with the different periods. The following step consists of permuting the working basis in order to establish a square sub-matrix associated with the capacity variables. The second part of the chapter is the update of the factorization which requires considering the position of the leaving column, the type of the entering column (route or slack) and the period of the entering and leaving columns.

In Chapter 6, we discuss the solution method of two linear systems encountered in the RSM. The first step of the RSM consists of finding the dual variables. It is equivalent to solving a linear system involving the basis with the costs of the dual variables on the right-hand side. The factorization procedure affects the solution method since the factorization plays the role of the inverse when solving the system. The second linear system finds the representation of the entering column in terms of the columns in the basis. Again, it is equivalent to a linear system with the basis as the coefficient matrix, but with the entering column as the right-hand side. We show the reduction in computational complexity of the decomposed factorization method as compared to the method using the direct factorization of the working basis.

In Chapter 7, we present a column generation method. We start by pointing out the necessity of a column generation method in this problem. Then we present a method by which we decompose the problem into as many sub problems of one period size as there are periods.

The last chapter summarizes the results obtained on the different aspects of the problem. The results are compared with other formulations and other approaches. We conclude by presenting possible extensions to this thesis.

Chapter 2 Formulation

The choice of a formulation is crucial for a problem having very distinct features. Characteristics such as multi-commodity flows, multi-period demands and the use of an undirected graph can be exploited in a formulation to reveal an interesting structure. Starting with these characteristics, we may obtain considerably different formulations. The efficiency of the algorithms to solve the different formulations of the same problem may vary radically. For this reason, it is necessary to find a formulation that allows the use of an efficient algorithm and permits us to fully exploit the structure of the problem.

2.1 Preliminary Notions

Before we present our formulation, we will illustrate in more detail the two concepts introduced in Chapter 1. The first is the Digital Cross Connect System (DXCS). It has the ability to reconfigure the interconnection of the circuits at its ports. The second is the concept of Dynamic Network Architecture (DNA) using DXCS to produce a network in which the architecture is adapted to the requirements of the moment.

2.1.1 Digital Cross Connect System (DXCS)

The role of the slow-switching equipment is totally different from the role accomplished by the switching equipment currently used in networks. Currently, switches are working on a real time basis. When a call arrives, the routing is immediately determined and the switches on the route establish a circuit. This is done in a network having a static and well defined architecture. The role of the switches is not changed by the introduction of slow-switches; only the routing procedure will be required to take into account the frequently changing architecture. Contrary to switches, DXCS does not have the capability to process a call. It is intended to give flexibility to the architecture of the network seen by the switches.

The structure of a network which incorporates DXCS's and switches cannot be compared to the structure of a network using only switches. First, there are no direct links allowed between switches, all switches are connected to one and only one DXCS. This is known as a hub structure, and an example is given in Figure 2.1 (switches and slow-switches are represented, respectively, by circles and squares). In this configuration all the circuits provided to a switch are routed through its corresponding DXCS.

The principle of reconfiguration made possible by the DXCS is illustrated in Figure 2.2. In this example we are using the simplest possible network. Only one slow-switch replaces the entire network. The circuit demands for each pair of nodes are given in the table and the resulting interconnection patterns for periods T and T' are shown in the network representation.

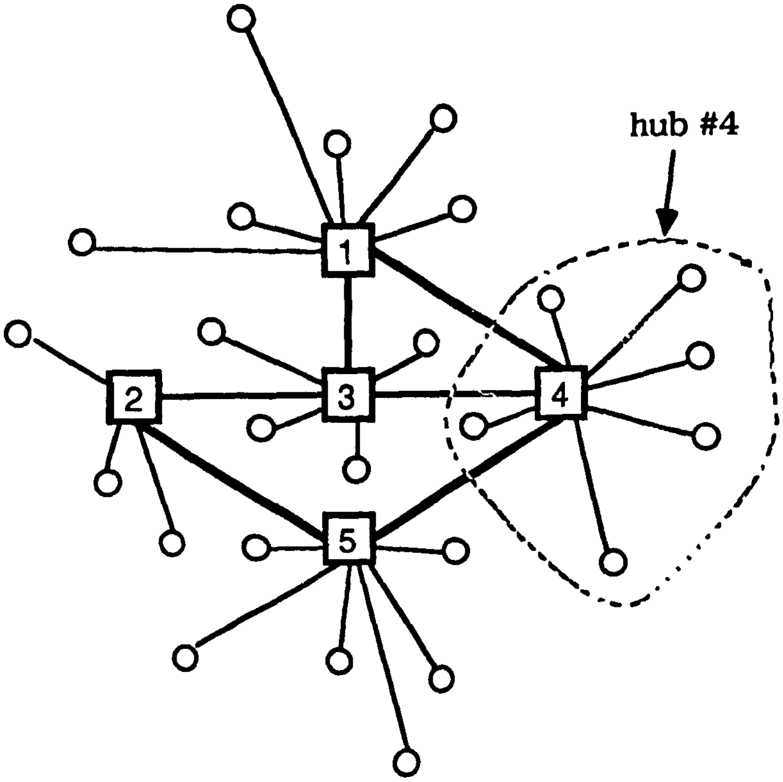
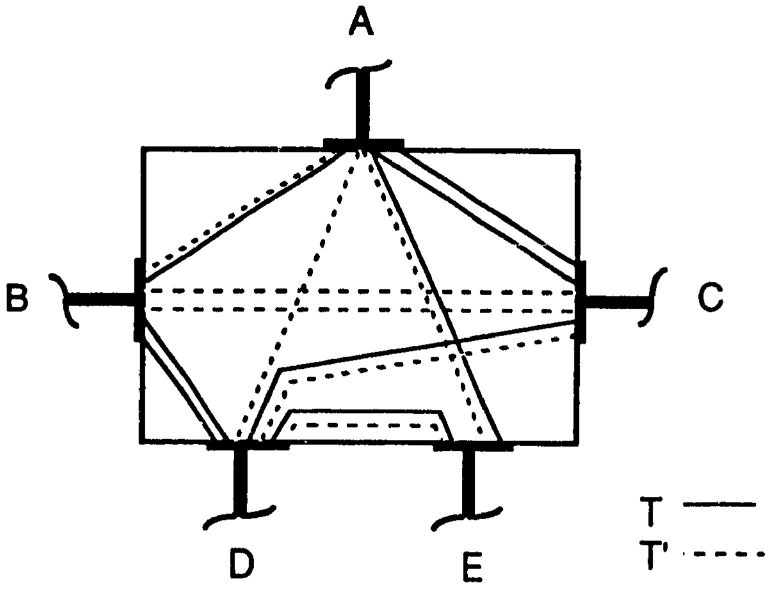


Fig 2.1 Hub structure



# circuits between	period T	period T'
A-B	1	1
A-C	2	0
A-D	0	1
A-E	1	1
B-C	0	2
B-D	2	0
B-E	0	0
C-D	1	1
C-E	0	0
D-E	1	1

Fig 2.2 DXCS

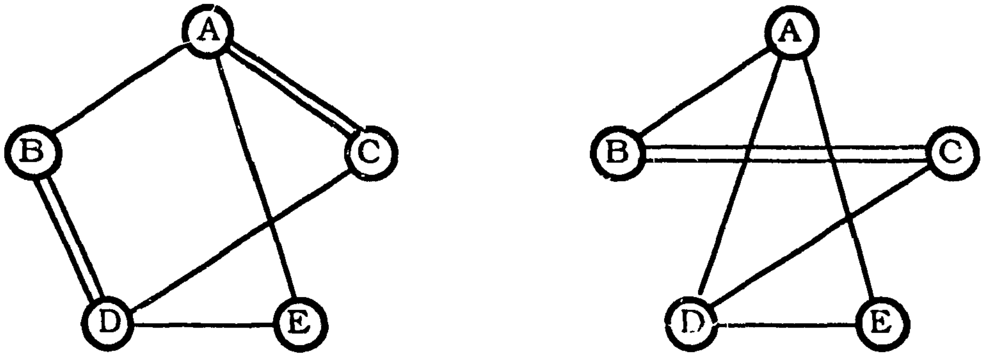
By varying the configuration of the DXCS, we are in fact changing the architecture of the network seen by the switches. In Figure 2.3, we present the resulting architecture for periods T and T' of Figure 2.2.

As we can see, the replacement of the network by a slow-switch permits radically changing the structure of the network. In the event of large non-coincidence in the demands over several periods, the possibility of rearranging the network with all the flexibility shown in Figures 2.2 and 2.3 is undoubtedly an advantage.

2.1.2 Dynamic Network Architecture (DNA)

Obviously, one cannot think of a network consisting of a single slow-switch as in Figure 2.2 to replace the entire telephone network. Economic and technical feasibilities would be unacheivable. The solution proposed in Figure 2.1 is the hubbing configuration. The network is replaced with a simpler network built with slow-switches. In this configuration each node is connected to exactly one DXCS. The slow-switch becomes the entry point to the network for all the switches clustered around it. It is called the DNA network because of its *dynamic architecture* capabilities made possible by the DXCS.

Before we present the implications of using a DNA network to replace the static networks currently in place, there is one assumption we can make without changing the original problem : all demands between switches of the same cluster are satisfied directly at their common DXCS without using the DNA network.



period T

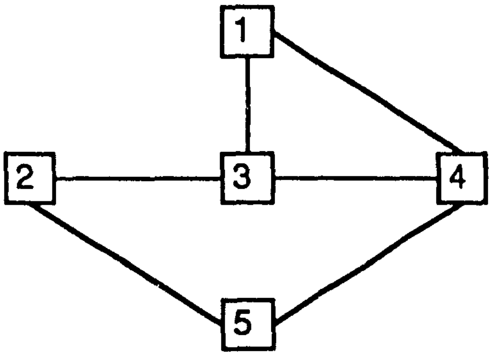
period T '

Fig 2.3 Example of rearrangement

Our prime objective is to minimize the size of the DNA network. Because they are considered automatically satisfied, demands within a cluster are not included in the demand matrices. Another important remark concerning circuit demands is that we group all the demands between the switches of DXCS i and the switches of DXCS j in a single demand d_{ij}. This permits considering only the DNA network without the switches and one demand in circuits for each pair of DXCS's. For example, a three period situation is illustrated in Figure 2.4.

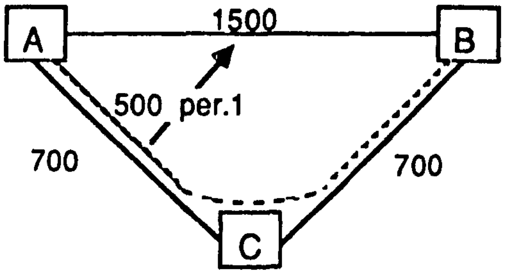
We have shown in Figure 2.2 the impressive flexibility which can be produced by a DXCS. This flexibility is preserved with DNA networks but appears in a different way. In the case of a DXCS, the flexibility is the result of its ability to connect a circuit between two ports whenever it is needed. With DNA, the multitude of routes available to establish a circuit between two DXCS's is the source of flexibility. For example, there are 4 possible routes,or ways, to assign a circuit between node 1 and node 2 in Figure 2.4: 1-3-2, 1-4-3-2, 1-4-5-2, 1-3-4-5-2. Obviously, we are interested only in routes that do not repeat nodes.

Figure 2.5 illustrates how DNA can be used to reduce the size of a network with non-coincident demands over two periods. The network considered here is simple but it demonstrates how the total cost (size) of a network can be reduced. The circuit requirements of period 1 is dominant on link A-B with a demand of 2000 circuits compared to 1500 circuits for period 2. The requirements of period 2 are dominant on links A-C and B-C with demands of 200 and 700 circuits respectively for periods 1 and 2.



$$\underset{\textit{period 1}}{\begin{bmatrix} \bullet & d_{12} & d_{13} & d_{14} & d_{15} \\ \bullet & \bullet & d_{23} & d_{24} & d_{25} \\ \bullet & \bullet & \bullet & d_{34} & d_{35} \\ \bullet & \bullet & \bullet & \bullet & d_{45} \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}} \quad \underset{\textit{period 2}}{\begin{bmatrix} \bullet & d_{12} & d_{13} & d_{14} & d_{15} \\ \bullet & \bullet & d_{23} & d_{24} & d_{25} \\ \bullet & \bullet & \bullet & d_{34} & d_{35} \\ \bullet & \bullet & \bullet & \bullet & d_{45} \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}} \quad \underset{\textit{period 3}}{\begin{bmatrix} \bullet & d_{12} & d_{13} & d_{14} & d_{15} \\ \bullet & \bullet & d_{23} & d_{24} & d_{25} \\ \bullet & \bullet & \bullet & d_{34} & d_{35} \\ \bullet & \bullet & \bullet & \bullet & d_{45} \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}}$$

Fig. 2.4 Multi-period rearrangeable network



	DEMAND		DIMENSION	
Link	Per. 1	Per. 2	current	DNA
A-B	2000	1500	2000	1500
A-C	200	700	700	700
B-C	200	700	700	700
		TOTAL:	3400	2900

Fig. 2.5 Example of cost reduction

In a network having no DNA capabilities it is very difficult to take advantage of the fact that all links are not dominated by the demands of the same period. One way of handling this situation in a non-DNA environment is to require that the capacity on each link be given by the circuit requirement of the dominant period on each link. In the case of our example (Figure 2.5), the capacities of links A-B, A-C and B-C will be respectively 2000, 700 and 700. This gives a total capacity of 3400 circuits. This method was proposed by R. Horn [30] in the context of network planning without *dynamic architecture*. Its weakness is the inability to exploit the spare capacity introduced by the presence of non-coincidence of demands over different periods.

With DNA it is possible to take advantage of the spare capacity in the network. Returning to the example of Figure 2.5, consider period 1 where there is spare capacity on links A-C and B-C whereas link A-B is saturated. Consequently, for each circuit A-C-B established at period 1 using the spare capacity on links A-C and B-C we can reduce the capacity on link A-B by 1 circuit. As a result, the use of the 500 free circuits on A-C and B-C does not increase the capacity of these circuits because period 1 is not dominant for these arcs, but the reduction of 500 circuits on A-B decreases the total number of circuits since it is dominated by period 1. The total size of the network is now 2900 circuits (1500, 700, 700) and the demands of both periods are satisfied. An important remark is that the circuit A-B has a capacity of 1500 circuits which is less than the demand of 2000 circuits for period 1. The flexibility of DNA permits allocating the 500 remaining circuits from the spare capacity of the network.

2.2 Type of Formulations

In this section we present the different formulations for solving the minimum cost sizing problem in the context of Dynamic Network Architecture (DNA) and a detailed description of the formulation selected, namely, the arc-chain representation. Clearly we are concerned with a multi-commodity flow problem. The impact of the multi-commodity flow aspect on the formulation is largely reduced by the presence of one and only one commodity per Origin-Destination pair (OD pair), allowing a single arc-chain matrix to represent all OD pairs of a given period.

The main feature of this problem is the existence of demands at different periods in time. As we shall see, it has considerable effect on the structure of the various formulations. First, flow conservation forces the addition of the equivalent of a complete single period multi-commodity flow problem for each period added. Since all periods share the same network, we must ensure that the capacity variables of the new period are integrated with the capacity variables of other periods to produce a consistent formulation over all periods.

Another very important characteristic of our problem is the presence of undirected arcs in the network. When we consider a single period problem, undirected arcs can be replaced by a pair of directed arcs in opposite directions. The demand of the OD pair k is then included in the formulation by defining one of the end nodes as the supply node and the other as the sink node. The effect on the formulation for a single period problem can be considered negligeable but in the context of multi-period demands the influence of two-way arcs cannot be neglected. We shall see in Section 2.4 that the orientation of the spare capacity caused by non-coincident demands is an important factor when rearranging the network. The node-arc formulation is especially affected by this problem whereas the arc-chain formulation in which the arcs are not explicitly represented in terms of their end nodes is not affected.

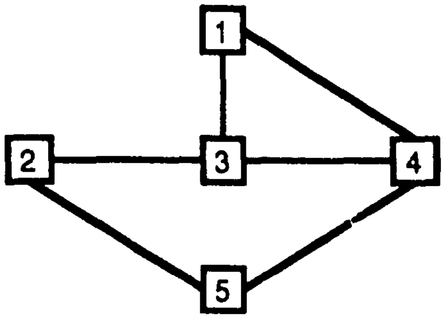
We will now present our formulation which is based on the arc-chain representation of the underlying network. The following section will show an alternative formulation using the node-arc representation of the network.

2.3 Arc-Chain Representation

The arc-chain representation of the network is based on a matrix A' having M rows and L columns. Each of the L columns of A' is associated with a route in the network. A column l is a 0-1 vector of length M with an entry 1 in rows corresponding to all arcs in route l and an entry 0 in all rows corresponding to the arcs which are not part of l. The enumeration of all possible routes is certainly not the most intuitive approach when representing a network. The node-arc representation presented in the next section is by far the most common and usually the most effective approach to network flow problems. For this reason, very few people have been interested by the arc-chain representation when solving the general multicommodity network flow problem. In fact, its use seems to be advantageous only on problems having very special characteristics such as the existence of a different commodity for almost every pair of nodes, with not more than one commodity per pair of nodes. Another important point is that the number of routes grows very fast with the size of the network. In the event that we cannot generate the routes only when they are needed (column generation) then the arc-chain formulation becomes nearly impossible to use. A column generation algorithm for the formulation we use will be presented in Chapter 7.

The first appearance of the arc-chain representation for the multi-commodity flow problem was in a paper by Ford and Fulkerson [23] in 1958. They were addressing a single period problem, namely, the maximal multi-commodity flow problem. The formulation of the maximal flow problem is simpler than the minimum cost problem because it requires only information about the structure of the network. In the minimum cost problem another set of constraints is required to force the flow of each commodity at the desired level. These constraints are not necessary in the maximal flow problem since the flow of each commodity is not given but is part of the objective function to optimize. The demand constraints insure that at a period h the sum of the flows on the routes connecting two nodes is equal to the demand between that particular Origin-Destination pair at period h. In our formulation all the demand constraints of a period are grouped in a single matrix called E. The matrix E is a block diagonal matrix in which each block is a row vector of length l_k, where l_k is the number of routes between the Origin-Destination pair k. This formulation has been used by McCallum [16] for solving the minimum cost *capacity expansion* problem in the single period context.

Figure 2.6 shows matrices A' , E, and the corresponding network for $N = 5$ nodes, $M = 6$ arcs, $K = \frac{N(N-1)}{2} = 10$ OD pairs, and all the $L = 32$ routes.



N = 5
M = 6
K = 10
L = 32

routes connecting OD-pair 13

$$A' = \begin{array}{c}\left[\begin{array}{cccc|ccc|ccccccccccccccccccccccccc}
1&0&0&1&1&0&0&0&1&1&0&1&1&0&0&0&1&0&0&1&0&0&1&0&1&0&0&0&1&0&0&1\\
0&1&1&0&0&1&1&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&1&0&0&0&1&0&0&1\\
1&1&0&0&0&0&1&0&0&1&0&0&1&1&1&0&0&1&0&1&0&1&1&0&0&1&1&0&0&0&1&1\\
0&0&1&1&0&0&1&0&0&1&0&0&1&1&0&1&1&0&1&0&1&0&0&0&0&1&1&0&0&0&1&1\\
0&1&0&1&0&1&0&0&1&0&0&1&0&1&0&1&0&1&0&0&0&1&0&1&0&0&0&1&0&0&1&0\\
0&0&1&1&0&0&1&0&0&1&1&1&0&0&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&1&0&0
\end{array}\right]\end{array}
\begin{array}{c}\text{arcs}\\ 1\text{-}3\\ 1\text{-}4\\ 2\text{-}3\\ 2\text{-}5\\ 3\text{-}4\\ 4\text{-}5\end{array}$$

$$E = \begin{array}{c}\left[\begin{array}{cccc|ccc|ccccccccccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1
\end{array}\right]\end{array}
\begin{array}{c}\text{O-D}\\ 1\text{-}2\\ 1\text{-}3\\ 1\text{-}4\\ 1\text{-}5\\ 2\text{-}3\\ 2\text{-}4\\ 2\text{-}5\\ 3\text{-}4\\ 3\text{-}5\\ 4\text{-}5\end{array}$$

Fig. 2.6 Arc-chain and demand matrices

Our formulation of the multi-period problem is based on the two matrices A' and E, both of which appear once for each period. In addition to this we need capacity variables that will provide the coupling between all periods. We have already assigned to E the role of insuring that the circuit requirements of each OD pair are satisfied for all periods. The matrix A' will represent the other set of constraints, the capacity constraints. The role of these constraints is to insure that the capacity of an arc is not exceeded at any period by the sum of the flows over all the routes using this arc. In the following sections, we present formal definitions of the two types of constraints.

2.3.1 Capacity Constraints

From a given set of flows g^h , $h = 1...H$, where g^h is a column vector of length L, we can compute the total flow on a given arc m for all H periods. In order to satisfy the requirements of every period, the value of the capacity on arc m has to be at least equal to the maximal value of flow on the arc over all H periods. In fact the sum of the flows must be less than or equal to the capacity. Thus

$$A'g^h \leq C, \qquad h = 1...H, \tag{2.1}$$

where C is the M-vector of capacity variables and $A' = (A'_{ml})$ is the $M \times L$ arc-route matrix and is defined as follows:

$$A'_{ml} = \begin{cases} 1, & \text{if arc}(m) \in \text{Route}(l), \\ 0, & \text{otherwise .} \end{cases} \tag{2.2}$$

Slack variables S^h are added to transform Equation (2.1) into standard form. The interpretation of the slack variables in terms of the network sizing problem is simple; they represent the spare capacity of each arc for the corresponding period. Thus,

$$A'g^h - IC + IS^h = 0, \qquad h = 1 \ldots H. \tag{2.3}$$

where the two identity matrices have size M.

2.3.2 Demand Constraints

The second set of constraints insures that the demand for each period is satified between every OD pair. Each column of $E_{K \times L} = (E_{kl})$ is associated with a route of A' and contains just one non-zero element corresponding to the end nodes (OD pair) of the route.

$$Eg^h = d^h, \tag{2.4}$$

where

$$E_{kl} = \begin{cases} 1, & \text{if route } l \text{ connects OD pair } k, \\ 0, & \text{otherwise,} \end{cases} \tag{2.5}$$

and d^h is the K-vector of OD-pair demands for period h.

The resulting minimal cost problem contains one matrix A', one matrix E, and two identity matrices per period.

$$\min \sum_{m=1}^{M} \$_m C_m$$

$$\left[\begin{array}{cccccc} A' & 0 & \cdots & 0 & & -\mathrm{I} \\ 0 & A' & & 0 & \mathrm{I}_{HM \times HM} & -\mathrm{I} \\ & \vdots & \ddots & & & \vdots \\ 0 & 0 & & A' & & -\mathrm{I} \\ E & 0 & \cdots & 0 & & \\ 0 & E & & 0 & 0 & \\ & \vdots & \ddots & & & \\ 0 & 0 & & E & & \end{array}\right] \left[\begin{array}{c} g^1 \\ g^2 \\ \vdots \\ g^H \\ S^1 \\ S^2 \\ \vdots \\ S^H \\ C \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ d^1 \\ d^2 \\ \vdots \\ d^H \end{array}\right], \qquad (2.6)$$

$$C, S^h, g^h \geq 0,$$

where $\$_m$ is the cost per circuit on arc m,

g^h are the flow variables, for period h,

C_m is the capacity variable of arc m.

S^h are the spare capacity variables for period h,

d^h are the demands for all OD pairs in period h,

M is the number of arcs in the network,

N is the number of nodes in the network,

L is the number of routes,

K is the number of Origin-Destination pairs ($\frac{K=N(N-1)}{2}$), and

H is the number of periods.

2.3.3 Scaling of the Constraints and Variables

The Generalized Upper Bounding technique (GUB) is the basis of our solution method for the minimal cost sizing problem of a rearrangeable network with multi period demands. The GUB is intended for the solution of linear programming problems having a very specific structure. It takes advantage of a formulation in which a part of the constraints has a block diagonal structure with each block being a linear convex combination of flow variables. The formulation of our problem presented in Equation 2.6 does not correspond exactly to the structure required by the GUB. It differs by the presence of demand vectors d^h on the right-hand side of the matrix E whereas the GUB requires the presence of ones on the right-hand side.

We achieve the GUB structure by scaling the flow variables. The scaling is carried out in two steps. First, the *demand constraints* (Figure 2.7a) are divided by the corresponding demand d_k^h (Figure 2.7b). Then the *columns* associated with routes connecting the Origin-Destination k in period h are multiplied by d_k^h (Figure 2.7c). Our objective is to exchange the d_k^h for 1 without affecting the block diagonal structure of unit vectors in E. By dividing each demand constraint by its d_k^h in step one we are replacing the right-hand side by a unit vector and altering E slightly. The matrix E remains in the block diagonal form but the blocks are now vectors of $\frac{1}{d_k^h}$.

The second step of the scaling process consists of restoring the ones in the matrix E. It is done by replacing the flow variables g_l^h with new variables f_l^h such that $g_l^h = f_l^h d_{\mathrm{OD}(l)}^h$. It is equivalent to multiplying each column of A' for period h by its corresponding d_k^h. The new arc-chain matrices now include the circuit demands of each period and must be labeled with the superscript h indicating the period they represent.

The new matrix $A^h = (A_{ml}^h)$ is defined as follows:

$$A_{ml}^h = \begin{cases} d_{\mathrm{OD}(l)}^h, & \text{if arc } m \in \text{Route } l, \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

$$\begin{array}{c} \begin{array}{cccc|cc} \multicolumn{4}{c}{\text{routes 1-2}} & & \end{array} \\ \left[\begin{array}{cccc|ccc} 1&0&0&1&1&0& \\ 0&1&1&0&0&1& \\ 1&1&0&0&0&0& \\ 0&0&1&1&0&0&\cdots \\ 0&1&0&1&0&1& \\ 0&0&1&1&0&0& \end{array}\right] \\ \left[\begin{array}{cccc|ccc} 1&1&1&1&0&0& \\ 0&0&0&0&1&1& \\ 0&0&0&0&0&0&\cdots \\ &&\vdots&&&& \end{array}\right] \end{array} g \quad = \quad \begin{bmatrix} 0\\0\\0\\0\\0\\0\\d_{12}\\d_{13}\\d_{14}\\\vdots \end{bmatrix}$$

Fig. 2.7 a) Arc-chain formulation before scaling

$$\begin{array}{c} \begin{bmatrix} 1&0&0&1&1&0& \\ 0&1&1&0&0&1& \\ 1&1&0&0&0&0& \\ 0&0&1&1&0&0&\cdots \\ 0&1&0&1&0&1& \\ 0&0&1&1&0&0& \end{bmatrix} \\ \begin{bmatrix} \frac{1}{d_{12}}&\frac{1}{d_{12}}&\frac{1}{d_{12}}&\frac{1}{d_{12}}&0&0& \\ 0&0&0&0&\frac{1}{d_{13}}&\frac{1}{d_{13}}&\cdots \\ 0&0&0&0&0&0& \\ &&&\vdots&&& \end{bmatrix} \end{array} g \quad = \quad \begin{bmatrix} 0\\0\\0\\0\\0\\0\\1\\1\\1\\\vdots \end{bmatrix}$$

Fig. 2.7 b) After dividing the demand constraints by the demands

$$\begin{array}{c} \begin{bmatrix} d_{12}&0&0&d_{12}&d_{13}&0& \\ 0&d_{12}&d_{12}&0&0&d_{13}& \\ d_{12}&d_{12}&0&0&0&0& \\ 0&0&d_{12}&d_{12}&0&0&\cdots \\ 0&d_{12}&0&d_{12}&0&d_{13}& \\ 0&0&d_{12}&d_{12}&0&0& \end{bmatrix} \\ \begin{bmatrix} 1&1&1&1&0&0& \\ 0&0&0&0&1&1& \\ 0&0&0&0&0&0&\cdots \\ &\vdots&&&&& \end{bmatrix} \end{array} f \quad = \quad \begin{bmatrix} 0\\0\\0\\0\\0\\0\\1\\1\\1\\\vdots \end{bmatrix}$$

Fig. 2.7 c) Final arc-chain formulation

2.3.4 Final Formulation

The final formulation satisfies the structure requirements of the GUB technique. The right-hand side of the demand constraints is composed exclusively of ones and the demands d_k^h are now integrated into the structure of the network in the matrices A^h. The matrix E has kept its block diagonal form where each block is a single row of ones, forming mutually exclusive linear convex combinations of flow variables f_l^h. Consequently, in each period h there is one linear convex combination per OD pair k , $k = 1...K$; $\sum_{\{l|\text{OD}(l)=k\}} f_l^h = 1$, where a flow variable f_l^h represents the fraction of $d_{\text{OD}(l)}^h$ serviced by route l. Our final formulation is thus

$$\min \sum_{m=1}^{M} \$_m C_m$$

$$\begin{bmatrix} A^1 & 0 & \cdots & 0 & & -I \\ 0 & A^2 & & 0 & I_{HM\times HM} & -I \\ & \vdots & \ddots & & & \vdots \\ 0 & 0 & & A^H & & -I \\ E & 0 & \cdots & 0 & & \\ 0 & E & & 0 & 0 & \\ & \vdots & \ddots & & & \\ 0 & 0 & \cdots & E & & \end{bmatrix} \begin{bmatrix} f^1 \\ f^2 \\ \vdots \\ f^H \\ S^1 \\ S^2 \\ \vdots \\ S^H \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \qquad (2.8)$$

$$C, S^h, f^h \geq 0,$$

where $\$_m$ is the cost per circuit on arc m,

f^h $= (f_l^h)$ where f_l^h is the fraction of $d_{\text{OD}(l)}^h$ satisfied using route l,

C is the vector of capacity variable,

S^h are the spare capacity variables for period h, M is the number of arcs in the network,

N is the number of nodes in the network,

L is the number of routes,

K is the number of Origin-Destination pairs ($K = \frac{N(N-1)}{2}$), and

H is the number of periods.

Equation 2.8 can be rewritten as follows:

$$\min \sum_{m=1}^{M} \$_m C_m$$

$$\begin{aligned} \sum_{k=1}^{K} \sum_{\{l|\mathrm{OD}(l)=k\}} A_{ml}^{h} f_{kl}^{h} + S_m^h - C_m &= 0, \qquad m = 1...M, h = 1...H, \\ \sum_{\{l|\mathrm{OD}(l)=k\}} f_{kl}^{h} &= 1, \qquad k = 1...K, h = 1...H, \\ C, S^h, f^h &\geq 0, \end{aligned} \tag{2.9}$$

where f_{kl}^h is a flow variable for period h corresponding to a route l connecting OD pair k.

Due to the nature of our problem, the arc-chain representation has shown numerous advantages over other mathematical representations of a network, the most important being the possibility of using the efficient GUB technique. Another significant advantage is that a routing for each period will be readily available at optimality from the flow variables. We will see in the next section that the "compact" formulation obtained is another reason for using the arc-chain representation.

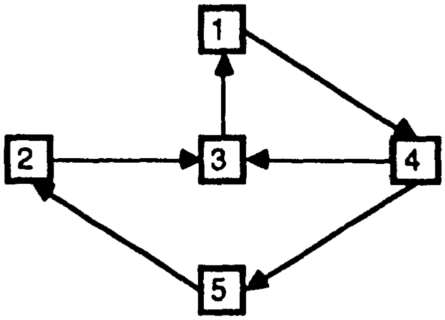
2.4 Node-Arc Representation

The node-arc representation of the network is based on the incidence matrix $F = (F_{nm})$. Each of the M columns of F is associated with one of the M arcs whereas the N rows are associated with the N nodes. A directed arc m is defined in F by its endpoints (t_m, e_m) where node t_m is the tail of arc m and node e_m the head of arc m. The column representing an arc $m = (t_m, e_m)$ contains only two nonzero elements ; the rows associated with nodes t_m and e_m, respectively, contain the elements -1 and 1.

$$F_{nm} = \begin{cases} 1, & \text{if node } n \text{ is the head of arc } m, \\ -1, & \text{if node } n \text{ is the tail of arc } m, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

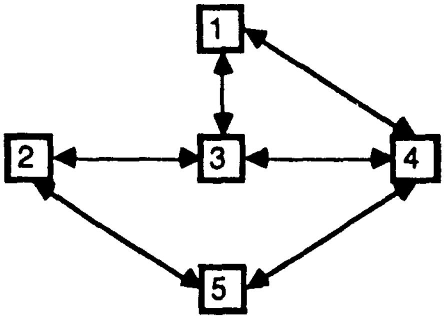
In an undirected network each arc is replaced by a pair of arcs in opposite directions. The node-arc incidence matrix of an undirected network can be given by ($F, -F$) in which $F_{N \times M} = (F_{nm})$ is the incidence matrix of a directed network with the arcs $m = (t_m, e_m)$ such that $t_m < e_m$. Examples of node-arc matrices for undirected and directed networks are given in Figure 2.8.

With this representation, the origin node of a flow arriving at a given node cannot be identified if there is more than one supply node. It would be difficult to determine the number of circuits between each OD pair if there were more than one Origin-Destination pairs represented by a single node-arc matrix. Consequently, *at least* one node-arc matrix is required per OD pair at each period and the total number of constraints becomes considerably large.



$$\text{node-arc incidence matrix } F = \begin{array}{c|cccccc|l} & 31 & 14 & 23 & 52 & 43 & 45 & \text{arc} \\ \hline 1 & 1 & -1 & 0 & 0 & 0 & 0 & \\ 2 & 0 & 0 & -1 & 1 & 0 & 0 & \\ 3 & -1 & 0 & 1 & 0 & 1 & 0 & \\ 4 & 0 & 1 & 0 & 0 & -1 & -1 & \\ 5 & 0 & 0 & 0 & -1 & 0 & 1 & \\ \hline \text{node} & & & & & & & \end{array}$$

Fig. 2.8 a) Directed graph



$$F = \begin{array}{c|cccccc|l} & 13 & 14 & 23 & 25 & 34 & 45 & \text{arc} \\ \hline 1 & -1 & -1 & 0 & 0 & 0 & 0 & \\ 2 & 0 & 0 & -1 & -1 & 0 & 0 & \\ 3 & 1 & 0 & 1 & 0 & -1 & 0 & \\ 4 & 0 & 1 & 0 & 0 & 1 & -1 & \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 & \\ \hline \text{node} & & & & & & & \end{array}$$

node-arc incidence matrix (F,-F)

Fig. 2.8 b) Undirected graph

2.4.1 Single Period Problem

The minimal cost sizing problem for an undirected network in the context of a single period demand has a trivial solution. It is not necessarily the case for multi-period demands or for directed networks. When we consider just one period there is no spare capacity available to route a circuit without adding one unit of capacity on each arc along a route. Each time we add a route, we are increasing the cost of the network by the sum of the unit costs of the arcs along the route. Thus it is impossible to take advantage of the existing capacity in order to add a route at a cost less than the total cost of the arcs. Since the cost of the network is given by the sum over all routes of the total unit cost of the route multiplied by the flow on the route, the best solution is obtained by routing each demand via the least cost path. Although we know the optimal solution, the formulation of the single period problem has some interesting properties which will reappear in the multi-period problem.

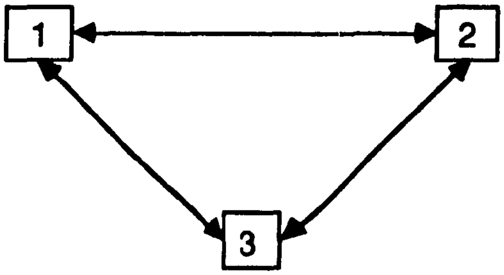
In the single period case, the treatment of the network as directed does not affect significantly the formulation of the problem. In addition to the fact that each arc is replaced by two arcs with opposite directions, only a small change in the demand vectors is necessary. When the demand between nodes i and j is considered, one of the nodes has to be designated as the source and the other as the sink. It introduces a direction on the demand and therefore on the spare capacity available for possible rearrangement. But, since there is no rearrangement of the spare capacity necessary in the single period problem, the introduction of directions does not affect the solution.

2.4.2 Multi-Period Problem

The multi-period problem is much more complex. In fact the trivial solution of the single period case becomes the starting point for the rearrangement procedure that optimizes the cost of the network. The problem arises when a direction is given to the demand, introducing a direction on the spare capacities and thus reducing the number of possible rearrangements.

Figure 2.9 illustrates a situation where it is possible to reduce the size of the network only if the direction of the demand 2 → 3 is reoriented in the other direction. In this example, we start with a spare capacity of 500 circuits from node 1 to node 2 at period 1, the arcs 1 to 3 and 2 to 3 each have 500 free circuits available for period 2. It is now impossible to use the spare capacities of period 2 and reduce the size of the arc 1-2 by establishing links along 1-3-2. At least 500 of the 700 circuits required between node 2 and 3 must be reoriented in the direction 3 → 2 if we want 500 circuits 1-3-2 created with the spare capacity of period 2.

In a larger network we cannot anticipate as to which share of the demand should be sent in each direction. Therefore, we have to formulate the multi-period problem in such a way that the demand is not oriented. For each Origin-Destination pair i and j we define two demand variables b_{ij}^{h} and b_{ji}^{h} representing the demands from i to j and j to i respectively in period h. Another set of constraints will insure that



				DIMENSION	
source → sink	demand Per. 1	demand Per. 2	spare capacity	before reorienting 2 → 3	after reorienting to 3 → 2
1 → 2	1500	2000	500 1 → 2 period 1	2000	1500
1 → 3	700	200	500 1 → 3 period 2	700	700
2 → 3	700	200	500 2 → 3 period 2	700	700
			TOTAL :	3400	2900

Fig. 2.9 Capacity orientation in the node-arc formulation

the sum of the demands in the two directions is equal to the original demand d^h_{ij}.

$$\begin{aligned}(F_{*} - F)f^h_{ij} + b^h_{ij}V_{ij} &= 0, \\ (F_{*} - F)f^h_{ji} + b^h_{ji}V_{ji} &= 0, \qquad (2.11) \\ b^h_{ij} + b^h_{ji} &= d^h_{ij},\end{aligned}$$

where V_{ij} is a column N-vector containing only two non-zero elements; positions i and j have values 1 and -1 respectively. The node-arc formulation requires HK $\frac{HN(N-1)}{2}$ copies of (2.11), one per OD pair at each period. In addition to the demand constraints, there are H sets of M constraints each of which insures that the capacity is not exceeded on any arc.

2.5 Comparison Between the Number of Constraints in the Node-Arc and Arc-Chain Formulations

The total number of constraints with the node-arc formulation is $H(N^3 - N^2 + M)$. If we consider a network with $H = 20$ periods, $N = 10$ nodes and $M = 25$ arcs we have almost **20000 constraints** in the node-arc formulation. The problem can be treated using the Dantzig-Wolfe Decomposition algorithm. It is then reduced to roughly 1000 problems of 20 constraints each and one master problem of 500 constraints.

The arc-chain formulation contains $H(M + \frac{N(N-1)}{2})$ or 1400 constraints for the previous example. Because of the structure of the arc-chain formulation the last 900 constraints will be considered implicitly by the Generalized Upper Bounding Technique. Thus, the solution of the problem is equivalent to solving a problem with only **500 constraints**.

Chapter 3 Review of the Revised Simplex and GUB Algorithms

All optimization techniques based on the simplex method proceed by improving the current basic feasible solution from one iteration to the next. The basis matrix, which is non-singular, defines a solution as well as a value to the objective function which is unique to this basis. At each iteration the simplex identifies an entering and a leaving column such that the exchange of the two columns leads to a new unique solution having an improved value of the objective function. The simplex is applied to problems of the following form:

$$\max z = \$x$$

$$Ax = b, \tag{3.1}$$

$$x \geq 0,$$

where z is the objective value associated with the solution x as evaluated by the objective function $z = \$x$.

The linear system $Ax = b$ and $x \geq 0$ is composed of a set of constraints which determines whether a solution is feasible or not. Since the region of feasibility is defined by *linear* constraints, the set of the feasible solutions is convex. This means that if two points a and b are feasible solutions then all the points on the line segment joining them are also feasible solutions. The resulting multidimensional region is called a polytope. An example of a polyhedron (a bounded polytope) arising from a problem having three variables and eight constraints (including the three non-negativity constraints) is presented in Figure 3.1.

We note that the number of variables determines the dimension of the Euclidean space in which the polytope resides whereas the number of constraints indicates the number of facets of the polyhedron (assuming no redundancy in the constraints).

The geometric interpretation of the objective function is a series of parallel planes, each corresponding to a different value z of the objective function. The vector \$ (the gradient of the objective function) composed of the unit cost of each variable indicates the direction of increase of the objective value. All the points on a plane orthogonal to the vector \$ have identical objective values (see Figure 3.2). The simplex searches for the tangent plane in the direction of \$.

The basic feasible solutions are the vertices of the polyhedron. The simplex procedure maximizes the value of the objective function by moving from vertex to vertex along the edges of the polytope defined by $Ax = b$, $x \geq 0$. If we suppose that we start at the point a in Figure 3.1 and that the corresponding objective function is the one illustrated in Figure 3.2 then a likely path followed by the simplex is the sequence of vertices $a \rightarrow b \rightarrow c \rightarrow d$.

The geometric interpretation of the simplex algorithm can be divided into two parts. First, we determine the possible edges to leave the current vertex. All edges directed on the same side of the current objective value plane as the gradient vector \$ will increase the value of the objective function if used. The best edge would be the one that is the most orthogonal to the current objective value plane, in other words, the edge that is the most parallel to the direction of increase of the objective function. The second step consists in determining the length of the displacement

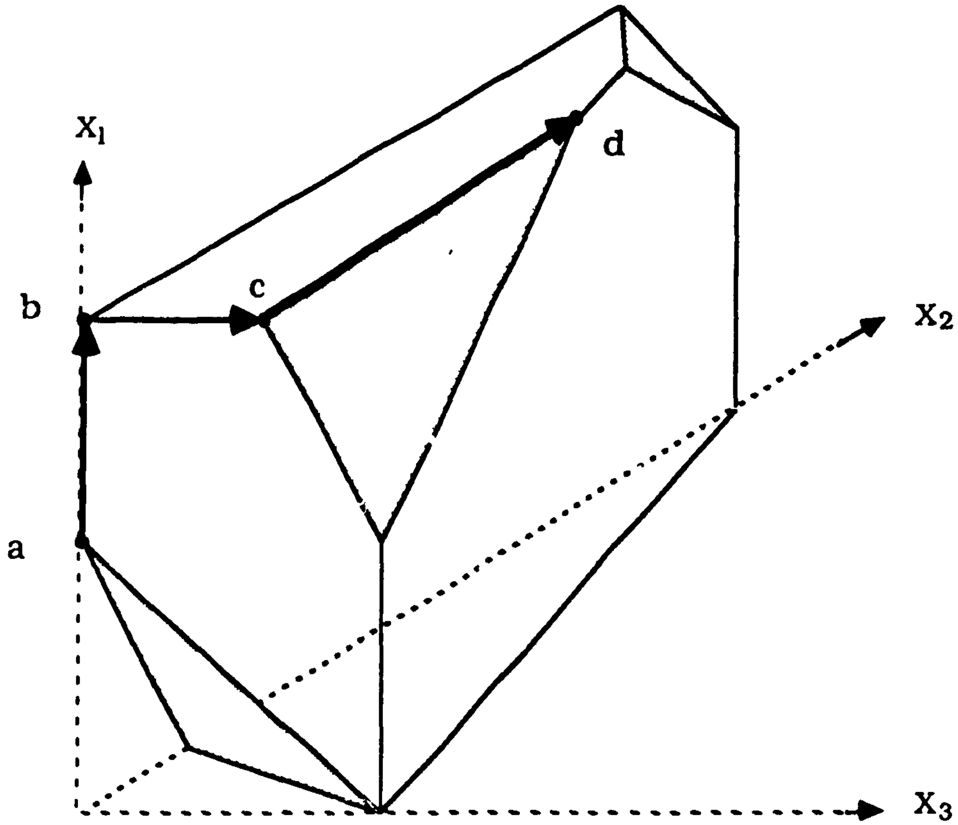


Fig. 3.1 Representation of a constraint set

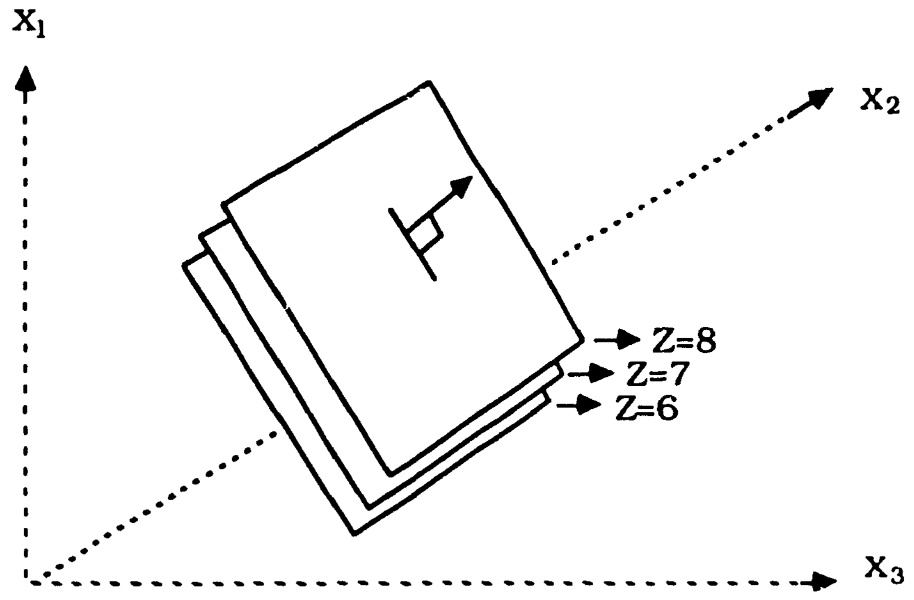


Fig. 3.2 Representation of the objective function

along the chosen edge. We repeat these two steps until we reach a vertex on the tangent plane in the direction $\$$. This point is the optimal solution to the linear program since there is no possibility of improving the value of $z = \$x$.

3.1 Revised Simplex Method

The introduction to this chapter may give the impression that the solution of a linear program is relatively easy. The reality is totally different. As the number of variables and constraints increases the size of the problem becomes too large for an efficient solution by the standard simplex method. This problem can be avoided by using the Revised Simplex Method (RSM), a more structured version of the simplex algorithm. Basically, the RSM is divided into six steps based on the solution of two linear systems involving the basis matrix B. The RSM is presented in Figure 3.3.

The first two steps define an entering variable (most promising edge). In the event that there is no variable satisfying the criteria for entering the basis, the current basis is declared optimal. Step 1 computes the dual variables μ associated with the current basis. In step 2 we verify if all the constraints of the dual problem are respected. Since there is a one-to-one correspondance between the dual constraints and the variables of the primal problem, the variables eligible to enter the basis are associated with the violated dual constraints. In Figure 3.3, we have used the *largest-coefficient* rule to determine the entering column; the entering column a^{ent} is defined as the non-basic column whose corresponding dual constraint is the most violated. In fact, $\$^j - \mu a^j$ is the rate at which z increases when the value of the

RSM

1. Find dual variables μ by solving $\mu B = \$^B$.

2. Choose the entering column a^{ent} corresponding to

$$\max_{a^j \notin B}(\$^j - \mu a^j).$$

   If $\$^j - \mu a^j \leq 0$ for all j, the current solution is optimal.

3. Find the representation σ of a^{ent} in terms of the columns of B.

$$B\sigma = a^{ent}.$$

4. Identify the most restricting constraint.

$$\min_{\sigma_i > 0} \frac{x_i^B}{\sigma_i} = \Theta.$$

5. Update $X_B = B^{-1}b - \Theta\sigma$.

6. Update the basis B .

Fig. 3.3 Revised Simplex Method

entering variable increases. The simplex algorithm always preserves primal feasiblity and complementary slackness [42]. When all dual constraints are respected we are at optimality.

The following two steps determine the rate of variation of the basic variables when the value of the entering variable is increased, and the leaving column a^{leav}. In step 3, we compute the representation of the entering column a^{ent} in terms of the columns of the basis. The elements of the resulting vector σ, associated with the columns of B, indicate the rate at which the values of the basic variables change when the value of the entering variable increases by 1. Since all variables in the simplex have a lower bound of 0, we increase the value of the entering variable until one of the basic variables reaches the value 0. This is done in step 4 by identifying the most restricting constraint. If all $\sigma_i \leq 0$, then the linear program is unbounded. In our problem, this cannot occur due to demand constraints.

In Steps 5 and 6, the different variables and matrices are updated to reflect the changes to the basic variables and the basis. The new basic variable associated with the entering column takes the value Θ whereas the other basic variables x_i^B take the value $x_i^B - \Theta\sigma_i$. In theory, the update of the basis is limited to the replacement of the leaving column in B by the entering column. In practice. those columns can be replaced either in the inverse of B or in a triangular factorization of B, since these are two computational devices which can be used to solve the two linear systems of Step 1 and 3 of the RSM. In the preface to his book, Chvátal [42] declares that "the inverse of the basis is an anachronism that has no place in modern versions of the revised simplex method.".

There are various reasons why the use of triangular factorization is superior to the use of the inverse.

- The most important reason is the possibility of exploiting a particular structure of the basis when using triangular factorization. The inverse usually does not preserve the structure of the original matrix. For our problem, this is an important consideration.
- The triangular factorization is (at least empirically) faster to compute than the inverse of the basis, and consequently round-off errors are more serious in the case of the inverse. In our case, the increase in calculation speed can be theoretically established.
- The solution of linear systems based on back substitution in triangular matrices is theoretically equivalent to solution using the inverse. In our case, one can show that back substitution is indeed faster.

The update of a triangular factorization is a complex problem, and will be covered in Chapter 5.

3.2 Generalized Upper Bounding Technique

For most large scale linear programs one must use the RSM as it becomes impossible to store in computer memory all the columns of the constraint matrix. Indeed, in some cases these columns are not immediately available, but are implicitly defined and must be generated during the course of the algorithm. For certain structured problems, techniques such as Dantzig-Wolfe decomposition or Generalized Upper Bounding (GUB) technique use the RSM after first reducing the number of rows of the constraint matrix.

The following formulation presented in Chapter 2, based on the *arc-chain* representation, has the structure required by the GUB.

$$\min \sum_{m=1}^{M} \$_m C_m$$

$$\left[\begin{array}{cccccc} A^1 & 0 & \cdots & 0 & & -1 \\ 0 & A^2 & & 0 & I_{HM\times HM} & -1 \\ & \vdots & \ddots & & & \vdots \\ 0 & 0 & & A^H & & -1 \\ E & 0 & \cdots & 0 & & \\ 0 & E & & 0 & 0 & \\ & \vdots & \ddots & & & \\ 0 & 0 & \cdots & E & & \end{array}\right] \begin{bmatrix} f^1 \\ f^2 \\ \vdots \\ f^H \\ S^1 \\ S^2 \\ \vdots \\ S^H \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{3.2}$$

$$C, S^h, f^h \geq 0,$$

where $\$_m$ is the cost per circuit on arc m,

A^h is the arc-chain matrix associated with period h,

E is a matrix indicating the OD-pair of each route ,

$f^h = (f_j^h)$ where (f_j^h) is the fraction of $d^h_{\mathrm{OD}(a^j)}$ satisfied using route a^j,

C is the vector of capacity variables,

S^h are the spare capacity variables for period h,

M is the number of links in the network, and

H is the number of periods.

Recall that N denotes the number of nodes in the network and $K = \frac{N(N-1)}{2}$ denotes the number of origin-destination pairs.

3.2.1 Properties of the Basis B

Each of the last HK constraints in 3.2 corresponds to a period and an OD-pair. Since these constraints are in fact linear convex combinations, then in any feasible solution at least one flow variable from each period must have a non zero value. Therefore all bases, which necessarily contain all the columns associated with non zero variables, are composed of at least one route from each origin-destination pair and period. We group one column from each period and OD-pair in the first HK columns of the basis; these columns are called the *key columns*. The square matrix in the bottom part of the key columns is an identity matrix of size $HK \times HK$, as the key columns comprise one column for every OD-pair and period. The upper part is composed of H matrices K^h, each of size $M \times K$, comprising the key columns of their corresponding period. Thus, B is given by

$$\begin{bmatrix} \mathrm{K} & \hat{B} \\ I & 0 \end{bmatrix} = \left[\begin{array}{cccc:c} \mathrm{K}^1 & 0 & \cdots & 0 & \\ 0 & \mathrm{K}^2 & \cdots & 0 & \hat{B}_{HM \times HM} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & \mathrm{K}^H & \\ \hdashline & \mathrm{I}_{\mathrm{HK} \times \mathrm{HK}} & & & C_{HK \times HM} \end{array} \right] \tag{3.3}$$

GUB transforms the basis using a matrix T defined as

$$T = \begin{bmatrix} I & -C \\ 0 & I \end{bmatrix}. \tag{3.4}$$

The effect of post-multiplication of B by T is the elimination of the matrix C and the subtraction of KC from $\hat{B}$.

$$BT = \begin{bmatrix} \mathrm{K} & \mathrm{WB} \\ I & 0 \end{bmatrix}, \tag{3.5}$$

where WB$=\hat{B} - \mathrm{K}C$ is a non-singular matrix called the *working basis.* The product KC, which is subtracted from $\hat{B}$, is never carried out in practice. In fact, the columns of WB are defined as follows:

- If a column is associated with a slack or a capacity variable then the column appears in WB unchanged.
- A column of WB representing a route is obtained by subtracting from the corresponding column of $\hat{B}$ the key column associated with the OD-pair and period of the route.

We shall demonstrate that the two linear systems solved in Steps 1 and 3 of the RSM can be reduced to systems based only on the working basis when using the GUB technique. Another important characteristic of WB is that the M capacity variables remain in the working basis throughout the entire execution of the algorithm. This is proved in Chapter 4. The other $(H-1)M$ columns of the working basis are grouped by period into submatrices Q^h so that

$$WB = \begin{bmatrix} Q^1 & 0 & \cdots & 0 & -\mathrm{I} \\ 0 & Q^2 & \cdots & 0 & -\mathrm{I} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & Q^H & -\mathrm{I} \end{bmatrix}, \tag{3.6}$$

where Q^h has size $M \times q^h$. Since there is a one-to-one correspondence between the columns of WB and the columns of $\hat{B}$ and C, they take the following form after permuting the columns as in WB:

$$\widehat{B} = \begin{bmatrix} \widehat{B}^1 & 0 & \cdots & 0 & 1 \\ 0 & \widehat{B}^2 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \widehat{B}^H & 1 \end{bmatrix}, \qquad (3.7)$$

$$C = \begin{bmatrix} C^1 & 0 & \cdots & 0 & 0 \\ 0 & C^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & C^H & 0 \end{bmatrix}. \qquad (3.8)$$

The matrix $\widehat{B}^h$ contains non-key columns from A^h and the unit M-vectors associated with the slack variables.

We have mentioned that for various reasons we opted to base the solution of the two linear systems on a factorization method rather than on the calculation of the inverse. Since the linear systems depend ultimately only on WB, we need to factorize only WB. The structure of WB allows for a very effective method of solution of these systems using principles taken from factorization and decomposition methods. This is discussed in Chapter 5.

3.2.2 The GUB Algorithm

Basically, the six steps of the GUB technique are identical to those of RSM. The difference lies in the way the two linear systems are solved using systems of size $HM \times HM$ based on the matrix WB. We now present the most important points of each step along with references to the chapter or section covering the subject in more detail.

3.2.2.1 Step 1. Finding Dual Variables (π, μ).

To find the dual variables we must solve the system

$$(\pi, \mu)B = (0, \ldots, 0, \$_1, \ldots, \$_M). \tag{3.9}$$

Both sides of the above equation are post-multiplied by the matrix T, giving

$$(\pi, \mu)\begin{bmatrix} \mathrm{K} & \mathrm{WB} \\ I & 0 \end{bmatrix} = (0, \ldots, 0, \$_1, \ldots, \$_M). \tag{3.10}$$

This system can be decomposed into the following two subsystems:

$$\pi \mathrm{WB} = (0, \ldots, 0, \$_1, \ldots, \$_M), \tag{3.11}$$

$$\mu = -\pi \mathrm{K}, \tag{3.12}$$

where the right-hand side of Equation 3.11 is composed of the last HM elements of the right-hand side of Equation 3.10. By further exploiting the structure of WB and K it is possible to considerably reduce the amount of work involved in solving the two subsystems. The details of the solution are given in Chapter 6.

3.2.2.2 Step 2. Choice of the Entering Column $\hat{a}^{ent}$.

The entering column is determined by

$$\min_{\hat{a}^j \notin B}(\$^j - (\pi, \mu)\hat{a}^j), \tag{3.13}$$

where $\hat{a}^j$ is a column of the constraint matrix shown in Equation 3.2.
Only the capacity variables have unit costs different from zero. and since they remain in the basis we have $\$^j = 0$, $\forall a^j \notin B$. Thus,

$$-(\pi, \mu)\hat{a}^{ent} = \min_{\hat{a}^j \notin B}(-(\pi, \mu)\hat{a}^j). \tag{3.14}$$

The value $(\pi,\mu)\hat{a}^{ent}$ is the rate at which the value of the objective function decreases when the entering variable increases by one unit. If $-(\pi,\mu)\hat{a}^{ent} \geq 0$, then it is impossible to reduce the current value of the objective function and, therefore, the current solution is optimal.

Equation 3.14 is not solved by enumerating all the routes and the slack variables, and then computing $-(\pi,\mu)\hat{a}^{j}$ for each one of them. We present in Chapter 7 a column generation technique which solves Equation 3.14 using the structure of the underlying network. In fact, the product $-\pi\hat{a}^{j}$ is the cost of the route $\hat{a}^{j}$ in a network having the $-\pi$ as costs on the arcs. By finding the shortest path and adding the corresponding μ we can determine the best variable eligible to enter the basis.

3.2.2.3 Step 3. Finding the Representation of $\hat{a}^{ent}$ in Terms of B.

The representation of the entering column in terms of the columns of the basis is found by solving the linear system

$$B\sigma = \hat{a}^{ent}. \tag{3.15}$$

In order to exploit the structure of the basis, we make a change of variables using the matrix T. Thus,

$$\sigma = T\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix}. \tag{3.16}$$

Substituting Equation 3.16 in 3.15, we obtain the following linear system:

$$BT\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix} = \begin{bmatrix}\mathrm{K} & \mathrm{WB}\\ I & 0\end{bmatrix}\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix} = \hat{a}^{ent}. \tag{3.17}$$

From the last HK equations of this system we conclude that σ_1 is equal to the last HK elements of $\hat{a}^{ent}$.

$$\sigma_1 = \begin{bmatrix} \hat{a}^{ent}_{HM+1} \\ \vdots \\ \hat{a}^{ent}_{HM+HK} \end{bmatrix}. \tag{3.18}$$

We shall demonstrate in Chapter 6 that $\mathrm{K}\sigma_1$ is, in fact, the key column associated with the OD-pair and period of column $\hat{a}^{ent}$. Therefore σ_2 is obtained by solving the following linear system:

$$\mathrm{WB}\sigma_2 = a^{ent}, \tag{3.19}$$

where a^{ent} is composed of the first HM elements of the column vector $\hat{a}^{ent} - \mathrm{KEY}(\hat{a}^{ent})$. Here $\mathrm{KEY}(\hat{a}^{ent})$ is the key column associated with route $\hat{a}^{ent}$.

An efficient procedure for solving this system, using the decomposed factorization, is given in Section 6.3.

3.2.2.4 Step 4. Identification of the Leaving Column $\hat{a}^{leav}$

The leaving column is associated with the most restricted variable currently in the basis, that is to say, it corresponds to the variable first attaining the value 0 when the value of the entering variable is increased from 0. It is the basic variable x^B_{leav} such that

$$\frac{x^B_{leav}}{\sigma_{leav}} = \min_{\substack{\sigma_i > 0 \\ 1 \leq i \leq HK+(H-1)M}} \frac{x^B_i}{\sigma_i} = \Theta. \tag{3.20}$$

In Theorem 4.1 of Section 4.2, it will be proved that if a column corresponding to a capacity variable is to be chosen for leaving the basis then there is a route or a slack variable which is also eligible to leave the basis. We shall retain the capacity variables in the basis throughout the execution of the algorithm so that the structure of $\widehat{B}$ and hence WB is preserved. This is reflected in Equation 3.20 where the last M basic variables are excluded from eligibility for leaving the basis.

Since all unit costs are greater than or equal to zero and all flows are upper-bounded by the demand constraints, it is not possible to have an unbounded solution ($\sigma_i \leq 0,\ \forall i = 1 \ldots HK + HM$). In other words, $\sigma_i > 0$ for at least one value of $1 \leq i \leq HK + HM$.

3.2.2.5 Step 5. Update of the Basic Variables x^B

The basic variables could be computed at each iteration as follows:

$$Bx^B = \begin{bmatrix} 0_{HM \times 1} \\ 1_{HK \times 1} \end{bmatrix}. \tag{3.21}$$

The row vector x^B is obtained much more easily by simply updating the current values of the basic variables. The basic variable associated with the leaving column is removed and replaced with the variable corresponding to the entering column, namely x^B_{ent}, which has a value of Θ. The other variables are updated by subtraction of the product $\Theta\sigma_i$.

$$\begin{aligned} x^B_{ent} &= \Theta, \\ x^B_i &= x^B_i - \Theta\sigma_i, \qquad i \neq ent. \end{aligned} \tag{3.22}$$

3.2.2.6 Step 6. Update of the Basis

The update of the working basis WB becomes simpler if the leaving column of B is also from WB. So our update procedure is in two parts. First we examine if the leaving column is also a key column. If so, it is brought into WB. Then the leaving and entering columns are exchanged in WB.

The details of the update procedure along with the update of the factorized form of WB are discussed in Chapter 5.

Chapter 4 Structure of the Problem

In this chapter, we study the structure introduced by the presence of multi period demands in a rearrangeable network. Two subjects strongly related to the mathematical structure of the problem are discussed: the construction of an initial basic solution and the possibility of preserving the structure of the basis from one iteration to the next by keeping the capacity variables in the basis.

In Chapter 3, we have shown that the solution of this minimum cost problem by the GUB technique depends mainly on the two linear systems (Equations 3.10 and 3.15) involving the working basis WB. The part of the basis containing the key columns is not used when solving the linear systems and this considerably reduces the size of the problems solved in steps 1 and 3 as compared to the RSM applied to the original problem. Although the basis used by GUB is the same as the one used by the simplex, a transformation of variables through the matrix T permits the decomposition of the basis and simplifies the solution of the two linear systems.

$$BT = \left[\begin{array}{cccc|c} \mathrm{K}^1 & 0 & \cdots & 0 & \\ 0 & \mathrm{K}^2 & \cdots & 0 & \mathrm{WB} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & \mathrm{K}^H & \\ \hline & & \mathrm{I}_{\mathrm{HK}\times\mathrm{HK}} & & 0 \end{array}\right], \tag{4.1}$$

$$\text{where} \qquad \mathrm{WB} = \begin{bmatrix} Q^1 & 0 & \cdots & 0 & -\mathrm{I}_\mathrm{M} \\ 0 & Q^2 & \cdots & 0 & -\mathrm{I}_\mathrm{M} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & Q^H & -\mathrm{I}_\mathrm{M} \end{bmatrix}.$$

GUB groups its basis BT into two sets of columns corresponding to the *key columns* and the *working basis* (Equation 4.1). The first set contains exactly one route from each OD pair at every period forming H matrices K^h of K columns each. Since each key column represents a different OD pair and period the result is an identity matrix $\mathrm{I}_{\mathrm{HK}\times\mathrm{HK}}$ in the last HK rows corresponding to the key columns. The second set is the last HM columns of BT composed of slack variables, capacity variables and routes. The last HK rows of these columns contain only zero values and the first HM rows contain the *working basis* WB.

4.1 Structure of the Working Basis

The presence of multi-period demands is also reflected in WB. The routes and slack variables are grouped by period forming H matrices Q^h of q^h columns and the last M columns are the capacity variables (Equation 4.1). The working basis will be used when solving the systems $\pi\mathrm{WB} = \$^{\mathrm{WB}}$ and $\mathrm{WB}\sigma_2 = a^{ent}$. These two systems could be solved much faster if it were possible to decompose WB by period, but the presence of the capacity variables does not permit a real decomposition into H subsystems each of size $M \times M$. In the next chapter, we show that WB can be decomposed into $H + 1$ parts corresponding to the H matrices Q^h and to a matrix X extracted from the last M columns. Each of these parts will be factorized separately and used highly interdependently to solve steps 1 and 3 of the RSM. The resulting factorization of WB is highly dependent on the structure of our formulation, characterized by the the presence of all the capacity variables in the basis.

In order to prove that the capacity variables can be made to remain in the basis we first show how an initial basic feasible solution containing all the capacity variables can be obtained (Section 4.1.1). Then we prove that the choice of a capacity variable as the leaving variable can be avoided (Section 4.2).

4.1.1 Initial Basic Solution

The problem of finding an initial basis has a trivial solution and can be solved in two steps. In this solution all the demands will be satisfied by the key columns. The working basis is composed exclusively of slack and capacity variables. The first step consists of choosing the key routes. The capacity of each arc is determined in the second step and finally the slack variables are selected to complete WB.

4.1.1.1 Initial Basis: Key Columns K^h

Each key column is associated with the demand of a given OD pair k and period h. In the initial basis the routes forming the key columns will be the only routes in the basis. Consequently all the demands d_k^h of period h and OD pair k will be routed by the key variable represented by the k$^{\text{th}}$ column of K^h. Any route connecting OD pair k is acceptable as the key column but in order to reduce the number of iterations we would like to have a route that will probably stay in the optimal basis. A good candidate for this role is the least cost route, as shown in 2.4: this solution is in fact optimal for a single period. Using the unit costs per arc $\$_m$ as lengths, we find the shortest paths between each of the K OD pairs. These K routes will form the key matrix K^h. The variables associated with these routes, having been scaled, assume the value 1. All other route variables have value 0. Since the unit costs of the arcs are the same for all periods, these least cost paths are the same for all periods; thus the initial key matrices K^h are the same for all periods. As an example, a network and the corresponding initial matrix K^h is presented in Figure 4.1.

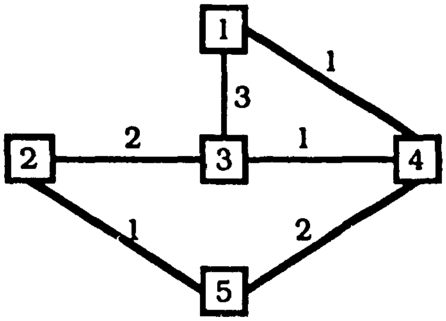
4.1.1.2 Initial Basis: Working Basis WB

The demand constraints having being satisfied by the variables corresponding to the key columns, we now have to satisfy the capacity constraints. In this initial solution the number of circuits necessary on arc m at period h is $\sum_{k=1}^{K} K_{mk}^h$. The capacity of arc m has to satisfy the requirement of every period, so that we must set

$$C_m = \max_{h=1...H} \{ \sum_{k=1}^{K} K_{mk}^h \}, \qquad m = 1, ..., M \ . \tag{4.2}$$

This gives M capacity variables in the basis. The basis is completed with slack variables. There are HM slack variables, one for each arc at every period. However only $(H-1)M$ columns are needed to complete the basis. At least one of the H slack variables associated with any arc m does not have to be in the basis (has the value 0). This is because from Equation 4.2 we know that in one of the periods, say h^*, capacity C_m is equal to the number of circuits used by this period on arc m. The slack variable value (spare capacity) is defined as the difference between the capacity and the number of circuits used, that is,

$$S_m^h = C_m - \sum_{k=1}^{K} K_{mk}^h \ . \tag{4.3}$$



$$K^h = \begin{array}{c|cccccccccc|c}
\text{OD pair} & 12 & 13 & 14 & 15 & 23 & 24 & 25 & 34 & 35 & 45 & \text{arcs} \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 \\
 & d^h_{12} & d^h_{13} & d^h_{14} & d^h_{15} & 0 & 0 & 0 & 0 & 0 & 0 & 14 \\
 & d^h_{12} & 0 & 0 & 0 & d^h_{23} & d^h_{24} & 0 & 0 & d^h_{35} & 0 & 23 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & d^h_{25} & 0 & d^h_{35} & 0 & 25 \\
 & d^h_{12} & d^h_{13} & 0 & 0 & 0 & d^h_{24} & 0 & d^h_{34} & 0 & 0 & 34 \\
 & 0 & 0 & 0 & d^h_{15} & 0 & 0 & 0 & 0 & 0 & d^h_{45} & 45
\end{array}$$

$$h = 1 \ldots H$$

Fig. 4.1 Initial key matrix

Therefore for period h^* the slack variable $S_m^{h^*}$ has a value zero and can be left out of the basis. The same is true for all M arcs, leaving $M(H - 1)$ slack variables for the completion of WB.

An example based on the network of Figure 4.1 ($M = 6$ and $H = 4$) is given in Figure 4.2. The number of circuits needed in each arc in order to satisfy the demand constraints using the least cost path at every period is shown in Figure 4.2a. From the same figure we determine the capacity of each arc and encircle its value. Figure 4.2b shows the value of the slack variables S_m^h that will be in the initial basis. The symbol $*$ indicates the dominant period h^* for each arc. The corresponding slack variable will be non-basic. The resulting working basis and the values of the basic variables are given in Figure 4.3.

4.2 Preservation of the Structure of WB

The structure of WB is characterized by the presence of all M capacity variables. We have shown that there is an initial basic feasible solution with this structure. We now prove that the structure is preserved from one iteration to the next by preventing the choice of a capacity variable as leaving variable. The leaving variable is determined in Steps 3 and 4 of the algorithm. Recall that in Step 3 we find the candidate to leave the basis as a function of the entering variable by first solving the linear system $B\sigma = \widehat{a}^{ent}$. All columns j such that $\sigma_j > 0$ are eligible to leave the basis. In Step 4 the leaving variable is identified as the first one whose value attains the value 0 when the value of the variable associated with $\widehat{a}^{ent}$ increases.

$$\sum_{k=1}^{K} K_{m\,k}^{h}$$

$m \backslash h$	1	2	3	4	C_m
13	$\textcircled{0}$	0	0	0	0
14	70	70	$\textcircled{90}$	80	90
23	$\textcircled{90}$	90	70	60	90
25	70	60	$\textcircled{80}$	60	80
34	$\textcircled{80}$	80	70	80	80
45	60	70	80	$\textcircled{90}$	90

Fig. 4.2 a) Determining the initial value of the capacity variables

Value of slack variables S_{l}^{h}.

$m \backslash h$	1	2	3	4
13	×	0	0	0
14	20	20	×	10
23	×	0	20	30
25	10	20	×	20
34	×	0	10	0
45	30	20	10	×

Fig. 4.2 b) Assigning initial value to the slack variables

$$WB = \left[\begin{array}{cccccccccccccccccccccccc}
\multicolumn{18}{c}{\longleftarrow (H\text{-}1)M \longrightarrow} & \multicolumn{6}{c}{\longleftarrow M \longrightarrow} \\
0&0&0&&&&&&&&&&&&&&&&-1&0&0&0&0&0 \\
1&0&0&&&&&&&&&&&&&&&&0&-1&0&0&0&0 \\
0&0&0&&&&&&&&&&&&&&&&0&0&-1&0&0&0 \\
0&1&0&&&&&&&&&&&&&&&&0&0&0&-1&0&0 \\
0&0&0&&&&&&&&&&&&&&&&0&0&0&0&-1&0 \\
0&0&1&&&&&&&&&&&&&&&&0&0&0&0&0&-1 \\
&&&1&0&0&0&0&0&&&&&&&&&&-1&0&0&0&0&0 \\
&&&0&1&0&0&0&0&&&&&&&&&&0&-1&0&0&0&0 \\
&&&0&0&1&0&0&0&&&&&&&&&&0&0&-1&0&0&0 \\
&&&0&0&0&1&0&0&&&&&&&&&&0&0&0&-1&0&0 \\
&&&0&0&0&0&1&0&&&&&&&&&&0&0&0&0&-1&0 \\
&&&0&0&0&0&0&1&&&&&&&&&&0&0&0&0&0&-1 \\
&&&&&&&&&1&0&0&0&&&&&&-1&0&0&0&0&0 \\
&&&&&&&&&0&0&0&0&&&&&&0&-1&0&0&0&0 \\
&&&&&&&&&0&1&0&0&&&&&&0&0&-1&0&0&0 \\
&&&&&&&&&0&0&0&0&&&&&&0&0&0&-1&0&0 \\
&&&&&&&&&0&0&1&0&&&&&&0&0&0&0&-1&0 \\
&&&&&&&&&0&0&0&1&&&&&&0&0&0&0&0&-1 \\
&&&&&&&&&&&&&1&0&0&0&0&-1&0&0&0&0&0 \\
&&&&&&&&&&&&&0&1&0&0&0&0&-1&0&0&0&0 \\
&&&&&&&&&&&&&0&0&1&0&0&0&0&-1&0&0&0 \\
&&&&&&&&&&&&&0&0&0&1&0&0&0&0&-1&0&0 \\
&&&&&&&&&&&&&0&0&0&0&1&0&0&0&0&-1&0 \\
&&&&&&&&&&&&&0&0&0&0&0&0&0&0&0&0&-1
\end{array}\right]
\begin{array}{c} \Big\uparrow \\ HM \\ \Big\downarrow \end{array}
\quad X^{WB} = \left[\begin{array}{c}
20 \\ 10 \\ 30 \\ \hline
0 \\ 20 \\ 0 \\ 20 \\ 0 \\ 20 \\ \hline
0 \\ 20 \\ 10 \\ 10 \\ \hline
0 \\ 10 \\ 30 \\ 20 \\ 0 \\ \hline
0 \\ 90 \\ 90 \\ 80 \\ 80 \\ 90
\end{array}\right]$$

Fig. 4.3 Initial working basis

Theorem 4.1: At each iteration there are at least H routes or slack variables eligible to leave the basis if a capacity variable is eligible to leave the basis.

Proof: We first prove that if the capacity variable associated with arc m^* is eligible to leave the basis, then there are at least H other candidates. From Figure 4.4 we see that C_{m^*} is present in one constraint per period. In equation form these H contraints are as follows:

$$\widehat{a}^{ent}_{(h-1)M+m^*} = \sum_{k=1}^{K} \mathrm{K}^{h}_{m^*k}\ \sigma^{\mathrm{K}^h}_{k} + \sum_{q-1}^{q^h} \widehat{\mathrm{B}}^{h}_{m^*q}\ \sigma^{\widehat{\mathrm{B}}^h}_{q} \qquad \sigma^{\mathrm{C}}_{m^*} \qquad h = 1 \ldots H\ ,$$

$$\widehat{a}^{ent}_{(h-1)M+m^*} + \sigma^{\mathrm{C}}_{m^*} = \sum_{k=1}^{K} \mathrm{K}^{h}_{m^*k}\ \sigma^{\mathrm{K}^h}_{k} + \sum_{q=1}^{q^h} \widehat{\mathrm{B}}^{h}_{m^*q}\ \sigma^{\widehat{\mathrm{B}}^h}_{q} \qquad h = 1 \ldots H\ . \tag{4.4}$$

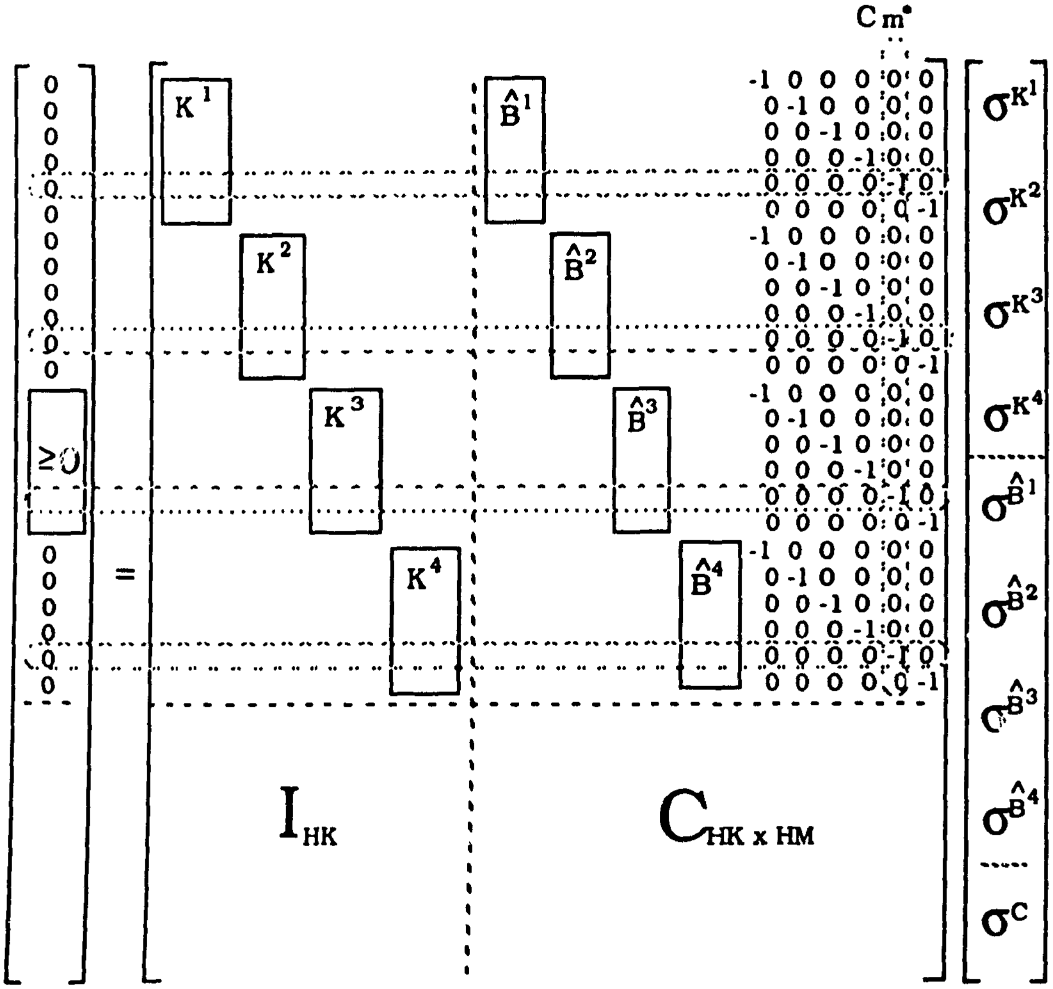
We know that $\widehat{a}^{ent} \geq 0$, $\mathrm{K}^{h}_{m^*k} \geq 0$, $\widehat{\mathrm{B}}^{h}_{m^*k} \geq 0$ and that if C_{m^*} is eligible to leave the basis then $\sigma^{\mathrm{C}}_{m^*} > 0$. Therefore according to Equation 4.4 there must be a route or a slack variable using arc m^* in each period h with $\sigma^{\mathrm{K}^h}_{k} > 0$ or $\sigma^{\widehat{\mathrm{B}}^h}_{q} > 0$ respectively.

We now have H candidates to leave the basis in addition to C_{m^*}. We complete the proof by showing that these H candidates attain a value 0 before or at the same time as the capacity variable C_{m^*}. In fact we will prove that it is impossible to have a non-zero value for the candidates when $C_{m^*} = 0$.

In the simplex procedure the primal constraints are not violated and the non-basic variables are all 0 leading to the following equation for our problem.

$$\begin{aligned} Bx^{B} &= \begin{bmatrix} 0_{HM\times 1} \\ 1_{HK\times 1} \end{bmatrix}, \\ \text{where} \qquad x^{B} &= \left[x^{\mathrm{K}^1}, x^{\mathrm{K}^2} \ldots, x^{\mathrm{K}^H}, x^{\widehat{\mathrm{B}}^1}, x^{\widehat{\mathrm{B}}^2} \ldots, x^{\widehat{\mathrm{B}}^H}, \mathrm{C}\right]^{\mathrm{T}}. \end{aligned} \tag{4.5}$$

$\hat{a}^{ent} = B\,\sigma$ (step 3)



Each $K^h \geq 0$ and each $\hat{B}^h \geq 0$ $\quad h=1...4$

Fig. 4.4 Proof of Theorem 1

We now extract the H equations associated with m^*

$$\sum_{k=1}^{K} \mathrm{K}_{m^*k}^{h} \, x_k^{\mathrm{K}^h} + \sum_{q=1}^{q^h} \hat{\mathrm{B}}_{m^*q}^{h} \, x_q^{\hat{\mathrm{B}}^h} - \mathrm{C}_{m^*} \quad 0, \qquad h \quad 1 \ldots H \, . \tag{4.6}$$

If the capacity variable C_{m^*} were to assume a value 0, then according to Equation 4.6 all route and slack variables using arc m^*, including the H candidates, must be equal to 0 forcing the H candidates to attain a value 0 before or at the same time as C_{m^*}. We conclude that they are all eligible to leave the basis and can be chosen instead of the capacity variable.//

Chapter 5 Factorization and Update

At each iteration, the *generalized upper bounding* technique solves two linear systems involving the working basis WB, in Steps 1 and 3. The use of the inverse WB ^{1} is not recommended for many reasons. First, the sparsity of a matrix is rarely preserved through the inversion process as shown in the example below due to Chvátal [42].

$$\begin{bmatrix} 1 & & & & 1 \\ 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix}.$$

As a consequence, it would be difficult to fully exploit the structure of WB when WB^{-1} is used for solving the linear systems. Another outcome of using the inverse is a possible poor accuracy of the solution obtained, round off errors being non-negligeable even with periodic reinversion of the matrix [44].

A more efficient approach is the factorization of the working basis which consists of a sequence of pivot matrices defined by a pre-processing of WB[45,46]. These matrices triangularize WB as is done in Gaussian elimination. When this factorization is used, the structure of WB permits its decomposition into much smaller submatrices which can be treated separately. The factorization can be updated at each iteration instead of being refactorized from scratch which takes much more time.

We present in the next section a factorization method that exploits the structure of WB and reduces the order of complexity for the solution of systems involving WB as well as the update of WB. The section following explains the update process of the factorization. The solution of steps 1 and 3 of the revised simplex using this method will be presented in Chapter 6.

5.1 Triangular Factorization

Given a basis matrix B, triangular factorization produces an upper triangular matrix U and a series of matrices $L_J, P_J, ..., L_1, P_1$such that

$$L_J P_J ... L_1 P_1 B = U. \tag{5.1}$$

The lower triangular matrices L_j differ from the identity matrix I only in the presence of non-zero elements below the diagonal and are of the form

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & \alpha_4 & 1 & & \\ & & \alpha_5 & & 1 & \\ & & \alpha_6 & & & 1 \end{bmatrix}.$$

The $L_j{}'s$ carry out Gaussian elimination on B in order to obtain the upper triangular matrix U. The permutation of the rows is done using permutation matrices P_j, obtained by permuting the rows of an identity matrix in the same order as the rows of the partially triangularized matrix should be permuted. This moves a proper non-zero element to the pivot position for the next elimination of below-diagonal non-zero elements.

The main objective of the factorization is to speed up the solution of the two linear systems: the computation of the dual variables in Step 1 and the representation of the entering column in terms of the basic columns in Step 3. In Step 1, we solve the system $(\pi,\mu)B = \$^B$ which is done by defining a new vector of variables Z such that

$$(\pi,\mu) = ZL_J P_J ... L_1 P_1. \tag{5.2}$$

Note that $L_J P_J ... L_1 P_1$ is non-singular, so that the variables Z are well-defined. The dual variables are computed by first solving

$$ZU = \$^B \tag{5.3}$$

and then by applying Equation 5.2.

In Step 3, the linear system $B\sigma = \hat{a}^{ent}$ is solved by replacing $\hat{a}^{ent}$ with $L_J P_J ... L_1 P_1 \hat{a}^{ent}$ and B with U. Then we get

$$U\sigma = L_J P_J ... L_1 P_1 \hat{a}^{ent}. \tag{5.4}$$

As we will see in Chapter 6 the solution of linear systems involving an upper triangular matrix is also very fast; only $\frac{M(M-1)}{2}$ multiplications and additions are necessary for a system of order M.

Although at first the factorization may seem to require more storage than the inverse, this is not necessarily the case. Indeed, we need only store the non-zero column of L_j; the rest is simply an identity matrix. The permutation matrices P_j will also be kept in memory as a vector, indicating in which order the permutation should be done.

5.1.1 Factorization of WB

In the previous section we assumed that the matrix B had no particular structure and we proposed to factorize the entire matrix without trying any decomposition. In fact, the matrix WB in the two linear systems that we have to solve has a very special structure that allows a compromise between a complete decomposition and a complete factorization. For each period the matrices Q^h will be factorized separately so that

$$L_J^h P_J^h \ldots L_1^h P_1^h Q^h = \hat{U}^h, \qquad h = 1 \ldots H \ . \tag{5.5}$$

Before we reveal the complete factorization method of WB, we show how the matrices L_J^h and P_J^h can be computed for a given Q^h.

5.1.1.1 Computation of the L_J^h and P_J^h

The upper traingular matrix $\hat{U}^h$ is obtained by Gaussian elimination on Q^h. It is important to note here that the $M \times q^h$ matrix Q^h is not necessarily a square matrix ($q^h \leq M$), and thus there will be $M - q^h$ rows of zeroes at the bottom of $\hat{U}^h$ whereas the first q^h rows will constitute a square matrix in upper triangular form.

The following three-step procedure is used iteratively to calculate the triangular factorization of $Q^h = (Q^h_{mq})$. In the literature, this procedure is known as QR factorization. 'n the first step, we define t^* as the position in the matrix of the leftmost column with non-zero elements below the diagonal. In the second step we permute row t^* with row m^*, where m^* is the row with the largest absolute value in column t^* excluding the rows above t^*. That is,

$$Q^h_{m^*t^*} = \max_{m=t^* \ldots M} \{|Q^h_{mt^*}|\} \ . \tag{5.6}$$

The permutation of rows t^* and m^* is represented by the matrix P_1^h. This matrix is the $M \times M$ identity matrix with the rows t^* and m^* interchanged. The resulting matrix $Q^{h'} = P_1^h Q^h$ does not have any non-zero entry below the diagonal in the first $t^* - 1$ columns and has $Q^h_{m^*t^*}$ as the diagonal element in column t^*.

$$Q^{h'} = \begin{bmatrix} - & - & - & - & & - \\ 0 & - & - & - & & - \\ 0 & 0 & Q^h_{m^*t^*} & - & \cdots & - \\ 0 & 0 & - & - & & - \\ & & \vdots & & & \\ 0 & 0 & - & - & & - \end{bmatrix}.$$

In the third step, we use the matrix L_1^h to eliminate all the non-zero elements below the diagonal of column t^*. L_1^h is an identity matrix except for non-zero values below the diagonal of column t^*.

$$L_1^h = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & \cdots & \\ & & \alpha_{t^*+1} & 1 & & \\ & & \vdots & & \ddots & \\ & & \alpha_M & & & 1 \end{bmatrix}, \tag{5.7}$$

$$\text{where } \alpha_i = -\frac{Q^{h\,'}_{it^*}}{Q^{h\,'}_{t^*t^*}} \qquad i = t^* + 1, \; ..., \; M \, .$$

The matrix $Q^{h''} = L_1^h P_1^h Q^h$ is obtained by multiplying $Q^{h'}$ on the left by L_1^h. It has zeroes below the diagonal in the first t^* columns.

We repeat these three steps until the matrix is upper triangular. Assuming that J iterations are necessary, we have

$$L_J^h P_J^h ... L_1^h P_1^h Q^h = \hat{U}^h. \tag{5.8}$$

The total number of additions or multiplications for the factorization of an $M \times M$ matrix Q^h is given by

$$\sum_{m=1}^{M} m(m-1) = \frac{M^3}{3} - \frac{M}{3}. \tag{5.9}$$

This value constitutes an upper bound for an $M \times q^h$ matrix. We conclude that the order of complexity for factorizing WB is $O(HM^3)$.

Example: We present an example where $M = 6$ and $q^h = 4$.

$$Q^h = \begin{bmatrix} -2 & 0 & -2 & 2 \\ 1 & 5 & 3 & 4 \\ -2 & 0 & -2 & 0 \\ 4 & 0 & 4 & 4 \\ 1 & 0 & 4 & 3 \\ 2 & 0 & 2 & 1 \end{bmatrix}.$$

The first column with a non-zero element below the diagonal is column $t^* = 1$, and the entry with the largest absolute value in this column is in row $m^* = 4$. P_1^h permutes rows 1 and 4 of the matrix Q^h. Thus,

$$P_1^h = \begin{bmatrix} & & & 1 & & \\ & 1 & & & & \\ & & 1 & & & \\ 1 & & & & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}.$$

The result of the permutation is the matrix $Q^{h'}$.

$$Q^{h'} = \begin{bmatrix} 4 & 0 & 4 & 4 \\ 1 & 5 & 3 & 4 \\ -2 & 0 & -2 & 0 \\ -2 & 0 & -2 & 2 \\ 1 & 0 & 4 & 3 \\ 2 & 0 & 2 & 1 \end{bmatrix}.$$

The elimination of the non-zero elements of columns t^* is done by the matrix L_1^h in which the α_i are equal to $-\frac{Q^{h}_{it^*}{}'}{Q^{h}_{t^*t^*}{}'}$. Thus

$$L_1^h = \begin{bmatrix} 1 & & & & & \\ -0.25 & 1 & & & & \\ 0.5 & & 1 & & & \\ 0.5 & & & 1 & & \\ -0.25 & & & & 1 & \\ -0.5 & & & & & 1 \end{bmatrix}$$

and

$$Q^{h''} = \begin{bmatrix} 4 & 0 & 4 & 4 \\ 0 & 5 & 2 & 3 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

The first column of the matrix $Q^{h''} = L_1^h P_1^h Q^h$ is now in upper triangular form and the leftmost column that does not respect the upper triangular form is $t^* = 3$. The element with the largest absolute value is in the fifth row, that is, $m^* = 5$. P_2^h will therefore permute rows 3 and 5.

$$P_2^h = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & & & 1 & \\ & & & 1 & & \\ & & 1 & & & \\ & & & & & 1 \end{bmatrix}$$

and

$$Q^{h'} = \begin{bmatrix} 4 & 0 & 4 & 4 \\ 0 & 5 & 2 & 3 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Column 3 already respects the upper triangular form. So, no elimination is necessary and we have $L_2^h = \mathrm{I}_{M \times M}$.

The last column is now the only one with non-zero elements below the diagonal and we have $t^* = 4$. The element with the largest absolute value in column t^* is in the fourth row so that $m^* = 4$. P_3^h is equal to $I_{M \times M}$ since $t^* = m^* = 4$; thus no permutation is required. The final step in the factorization is the elimination of the elements below the diagonal of column 4 of $Q^{h'}$.

$$L_3^h = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & 0.5 & 1 & \\ & & & 0.25 & & 1 \end{bmatrix}.$$

The factorized matrix $\widehat{U}^h = L_3^h P_3^h L_2^h P_2^h L_1^h P_1^h Q^h$ is obtained by multiplying the last $Q^{h'}$ on the left by L_3^h and is given below.

$$\widehat{U}^h = \begin{bmatrix} 4 & 0 & 4 & 4 \\ 0 & 5 & 2 & 3 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

5.1.1.2 Decomposition of the Factorization

In Section 5.1.1.1, we have shown how to factorize a matrix of size $M \times q^h$, $q^h < M$. We now present the method used to decompose WB into $H + 1$ such matrices which then can be factorized separately. This will reduce considerably the amount of work necessary when updating WB and solving steps 1 and 3 because it is easier to solve and update H subsystems of size $M \times M$ than it is to do the same for an $HM \times HM$ system.

The H matrices, Q^h, are readily available in WB. From the factorization of each Q^h we obtain the matrices $\widehat{U}^h, \widehat{LP}^h, L_J^h, P_J^h, ..., L_1^h, P_1^h$, where $\widehat{LP}^h$ is the triangularization matrix of Q^h defined as follows:

$$\widehat{LP}^h = L_J^h P_J^h ... L_1^h P_1^h. \tag{5.10}$$

This first step in the factorization of WB can be represented in matrix form by a multiplication of WB on the left by a block diagonal matrix LP, containing the $\widehat{LP}^h$ as diagonal blocks.

$$LP = \begin{bmatrix} \widehat{LP}^1 & & & \\ & \widehat{LP}^2 & & \\ & & \ddots & \\ & & & \widehat{LP}^H \end{bmatrix}. \tag{5.11}$$

The resulting matrix keeps the same form as WB, except that the $I_{M \times M}$ matrices are replaced by the $\widehat{LP}^h$ and the Q^h by the $\hat{U}^h$.

$$LP \cdot \mathrm{WB} = \begin{bmatrix} \hat{U}^1 & & & & -\widehat{LP}^1 \\ & \hat{U}^2 & & & -\widehat{LP}^2 \\ & & \ddots & & \vdots \\ & & & \hat{U}^H & -\widehat{LP}^H \end{bmatrix}. \tag{5.12}$$

We note that the $\hat{U}^h$ are rectangular matrices of size $M \times q^h$ having $M - q^h$ rows of zeroes at the bottom. We know that $\sum_{h=1}^{H} q^h = (H-1)M$ since only the last M of the HM columns in WB are not in the Q^h matrices. Therefore, the total number of rows having only zeroes up to column $(H-1)M$ in $LP \cdot \mathrm{WB}$ is given by

$$\sum_{h=1}^{H} (M - q^h) = HM - (H-1)M = M. \tag{5.13}$$

The second step in the factorization is the permutation of these M rows to the bottom part of the matrix using a permutation matrix V.

$$U = V \cdot LP \cdot \mathrm{WB} = \begin{bmatrix} U^1 & & & & -LP^1 \\ & U^2 & & & -LP^2 \\ & & \ddots & & \vdots \\ & & & U^H & -LP^H \\ & & & & X^T \end{bmatrix}. \tag{5.14}$$

The first $(H-1)M$ columns are now upper triangular. In fact, the matrix U has a block diagonal structure in which each block is in upper triangular form. An important aspect related to the storage of V is that the entire permutation matrix is not kept in memory but rather only a vector of length HM indicating in which order the rows should be permuted.

Even though the transformation of WB into U is expressed as the multiplication of WB by the matrices V and LP, wnich represents a considerable amount of computation time, an implementation would not require these multliplications to be carried out. In fact, after the first step when the Q^h are factorized, the matrices U^h, LP^h, X^T can be established simply by assigning the proper rows of $\widehat{U}^h$ and $\widehat{LP}^h$ to the matrices U^h, LP^h and X^T. The matrices U^h are upper triangular square matrices composed of the first q^h rows of $\widehat{U}^h$ just as $-LP^h$ is composed of the first q^h rows of $-\widehat{LP}^h$. The last $M - q^h$ rows of each $-\widehat{LP}$ are grouped by the permutation matrix V in an $M \times M$ matrix named X^T. An example of this procedure is given in Figure 5.1

The matrix U is almost upper triangular; only X^T is not factorized yet. In order to simplify the update of the factorization we will not factorize X^T but its transpose X. The reason for this is that if a column is changed in a matrix only its corresponding column of the associated upper triangular factorization is altered. However, when a row is changed, the result could be a complete fill-in of its associated upper triangular factorization, as shown in Figure 5.2. Consequently, the matrix X will be defined such that changes will be done on its columns and not on its rows. If we look more closely at X^T, we see that it is composed of rows coming from the different $-\widehat{LP}^h$. The changes to the $-\widehat{LP}^h$ appear as changes to the rows of the matrix in the bottom right-hand corner of U. By defining this matrix as the transpose of X, the changes now appear as changes to the columns of X and the

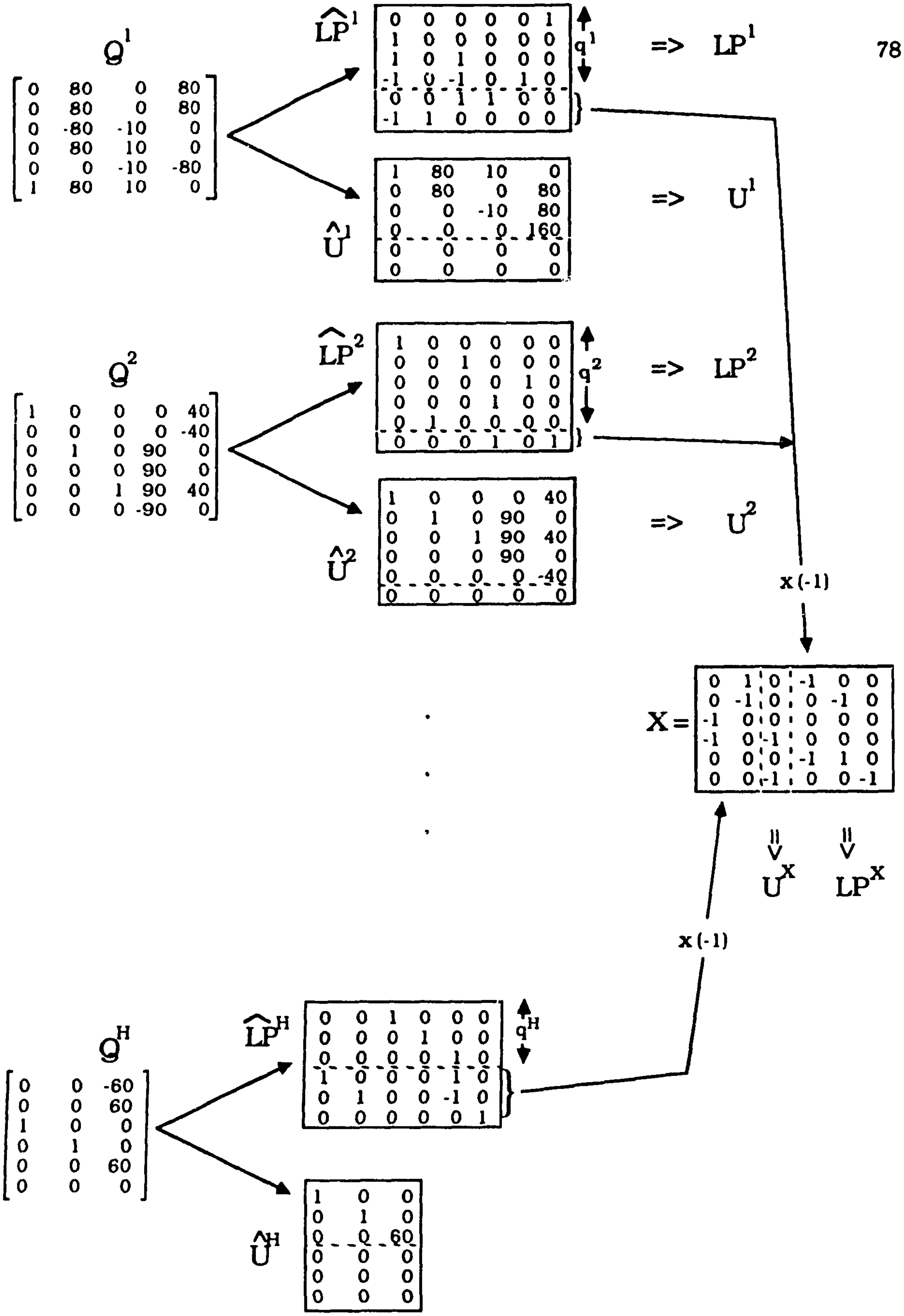


Fig. 5.1 Example of a decomposed factorization

update of the factorization can be accomplished more easily this way.

The last step is the factorization of the matrix X by the same method we used to factorize the matrices Q^h. Since X is a square matrix, no row of zeroes will appear under the upper triangular part. The result is an upper triangular square matrix U^X such that

$$L^X_{M-1}P^X_{M-1}\dots L^X_2 P^X_2 L^X_1 P^X_1 X = LP^X X = U^X . \tag{5.15}$$

5.1.1.3 Complexity Analysis

The factorization of WB can be divided into three parts:

1) Each Q^h is factorized to obtain the $\widehat{LP}^h$ and $\widehat{U}$.

2) Given the q^h, the matrices LP^h, U^h, X^T are established from $\widehat{LP}^h$ and $\widehat{U}^h$.

3) X is factorized by LP^X into U^X

We have demonstrated at the end of Section 5.1.1.1 that the number of additions and multiplications in the factorization of an $M \times M$ matrix is $O(M^3)$ [47,48]. Part 1) contains H factorizations of this type for a complexity of $O(HM^3)$. In part 2) we start by establishing a permutation matrix V. In fact, we only need a vector of length HM that indicates in which order the rows of $LP \cdot WB$ should be permuted; thus a complexity of $O(HM)$. The LP^h and U^h are then defined as the first q^h rows of $\widehat{LP}^h$ and $\widehat{U}^h$, respectively and this does not require any computation. The same is also true for the construction of X^T from the last $M - q^h$ rows of $\widehat{LP}^h$.

$$X = \begin{bmatrix} 0 & 80 & 80 & 0 \\ 0 & 80 & 80 & 0 \\ 0 & -80 & 0 & -30 \\ 0 & 80 & 0 & -30 \\ 0 & 0 & -80 & 30 \\ 1 & 80 & 0 & 30 \end{bmatrix}$$

$$LP^{X} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\overset{X}{U} = \begin{bmatrix} 1 & 80 & 0 & 30 \\ 0 & 80 & 80 & 0 \\ 0 & 0 & 80 & -30 \\ 0 & 0 & 0 & -60 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

after row 1 of the matrix X is replaced by [1 -80 0 30]

$$\begin{bmatrix} 1 & -80 & 0 & 30 \\ 0 & 80 & 80 & 0 \\ 0 & -80 & 0 & -30 \\ 0 & 80 & 0 & -30 \\ 0 & 0 & -80 & 30 \\ 1 & 80 & 0 & 30 \end{bmatrix}$$

identical

$$\begin{bmatrix} 1 & 80 & 0 & 30 \\ 1 & -80 & 0 & 30 \\ 1 & -160 & 0 & 0 \\ 1 & -160 & -80 & -30 \\ 1 & -160 & -80 & 30 \\ -1 & 160 & 80 & -30 \end{bmatrix}$$

Fig. 5.2 Effect of changing a row

The last step is a factorization of the $M \times M$ matrix X which is an operation of order $O(M^3)$. Clearly the dominant operation is part 1) with a complexity of $O(HM^3)$.

The factorization of WB without decomposing the working basis would be of complexity $O(H^3M^3)$. This is the work necessary for the Gaussian elimination of an $HM \times HM$ matrix. The decomposed factorization is superior to the direct factorization by a factor H^2. In addition to this, we shall see that the new approach considerably reduces the amount of work required to solve the two linear systems and to carry out the Column Generation.

5.1.2 Factorization of the Initial Basis

In Section 4.1.1 we presented the special structure of the initial basis. Its sparsity allows for a much simpler factorization procedure.

Due to the nature of the initial Q^h, composed exclusively of slack variables, there is just one non-zero element per column. The factorization of such matrices requires only the use of permutation ma'rices which can be defined as we construct the initial Q^h. For the basis illustrated in Figure 4.3, the permutation of the H initial Q^h is done as follows.

The matrix P_1^1 associated with Q^1 permutes the rows into the order 2,4,6,1,3.5. The rows of period 3 are interchanged into the order 1,3,5,6.2,4. Periods 2 and 4 do not require any permutation since they already are diagonal, hence $P_1^2 = P_1^4 = I$. The resulting matrix LP WB is given in Figure 5.3, where we ascertain that the first q^h rows of each Q^h are in upper triangular form and that the $\widehat{LP}^h = P_1^h$ are readily available from the last M columns.

After the factorization of the initial Q^h, we have H identity matrices I in the first $(H - 1)M$ columns of LP WB. Since WB has HM rows there are M rows of zeroes (Figure 5.3). In fact, there are $M - q^h$ such rows in period h for a total of M rows over all periods. We permute these rows to the bottom using the matrix V and in this way establish the matrix X^T. Therefore, the columns of X are the last $M - q^h$ rows of $\widehat{LP}^h$, for all periods $h = 1$ to H (Figure 5.4). As a matter of fact, for this initial factorization all factorization matrices $\widehat{LP}^h$ are actually permutation matrices and all their rows contain just one non-zero element. The matrices WB, LP, V are all non-singular, so the matrix $V \cdot LP \cdot$ WB as well as the matrix X are non-singular. If the square matrix X is non-singular and each column contains only one non-zero element then all rows must also contain one of the non-zero entries and X can be diagonalized using only a permutation matrix P_1^X. The rows of X in the last example have to be permuted into the order 1,3,5,2,4,6.

$$X = \begin{bmatrix} -1 & & & & & \\ & & & -1 & & \\ & -1 & & & & \\ & & & & -1 & \\ & & -1 & & & \\ & & & & & -1 \end{bmatrix} \quad \text{and} \quad LP^X = P_1^X = \begin{bmatrix} 1 & & & & & \\ & & 1 & & & \\ & & & & 1 & \\ & 1 & & & & \\ & & & 1 & & \\ & & & & & 1 \end{bmatrix}.$$

$$\begin{bmatrix} \hat{U}^1 & & & & -\widehat{LP}^1 \\ & \hat{U}^2 & & & -\widehat{LP}^2 \\ & & \hat{U}^3 & & -\widehat{LP}^3 \\ & & & \hat{U}^4 & -\widehat{LP}^4 \end{bmatrix} = \left[\begin{array}{cccccccccccccccccccccccc}
1 & 0 & 0 & & & & & & & & & & & & & & & & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & & & & & & & & & & & & & & & & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & & & & & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & & & & & & & & & & & & & & & & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & & & & & & & & & & & & & & & & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & & & & & & & & & & & & & & & & 0 & 0 & 0 & 0 & -1 & 0 \\
& & & 1 & 0 & 0 & 0 & 0 & 0 & & & & & & & & & & -1 & 0 & 0 & 0 & 0 & 0 \\
& & & 0 & 1 & 0 & 0 & 0 & 0 & & & & & & & & & & 0 & -1 & 0 & 0 & 0 & 0 \\
& & & 0 & 0 & 1 & 0 & 0 & 0 & & & & & & & & & & 0 & 0 & -1 & 0 & 0 & 0 \\
& & & 0 & 0 & 0 & 1 & 0 & 0 & & & & & & & & & & 0 & 0 & 0 & -1 & 0 & 0 \\
& & & 0 & 0 & 0 & 0 & 1 & 0 & & & & & & & & & & 0 & 0 & 0 & 0 & -1 & 0 \\
& & & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & -1 \\
& & & & & & & & & 1 & 0 & 0 & 0 & & & & & & -1 & 0 & 0 & 0 & 0 & 0 \\
& & & & & & & & & 0 & 1 & 0 & 0 & & & & & & 0 & 0 & -1 & 0 & 0 & 0 \\
& & & & & & & & & 0 & 0 & 1 & 0 & & & & & & 0 & 0 & 0 & 0 & -1 & 0 \\
& & & & & & & & & 0 & 0 & 0 & 1 & & & & & & 0 & 0 & 0 & 0 & 0 & -1 \\
& & & & & & & & & 0 & 0 & 0 & 0 & & & & & & 0 & -1 & 0 & 0 & 0 & 0 \\
& & & & & & & & & 0 & 0 & 0 & 0 & & & & & & 0 & 0 & 0 & -1 & 0 & 0 \\
& & & & & & & & & & & & & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
& & & & & & & & & & & & & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
& & & & & & & & & & & & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
& & & & & & & & & & & & & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
& & & & & & & & & & & & & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
& & & & & & & & & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}\right]$$

Fig 5.3 Factorization of the initial Q^h

$$\begin{bmatrix} U^1 & & & & -LP^1 \\ & U^2 & & & -LP^2 \\ & & U^3 & & -LP^3 \\ & & & U^4 & -LP^4 \\ & & & & X^T \end{bmatrix} = \left[\begin{array}{cccccccccccccccccccccccc}
1&0&0&&&&&&&&&&&&&&&&0&-1&0&0&0&0 \\
0&1&0&&&&&&&&&&&&&&&&0&0&0&-1&0&0 \\
0&0&1&&&&&&&&&&&&&&&&0&0&0&0&0&-1 \\
&&&1&0&0&0&0&0&&&&&&&&&&-1&0&0&0&0&0 \\
&&&0&1&0&0&0&0&&&&&&&&&&0&-1&0&0&0&0 \\
&&&0&0&1&0&0&0&&&&&&&&&&0&0&-1&0&0&0 \\
&&&0&0&0&1&0&0&&&&&&&&&&0&0&0&-1&0&0 \\
&&&0&0&0&0&1&0&&&&&&&&&&0&0&0&0&-1&0 \\
&&&0&0&0&0&0&1&&&&&&&&&&0&0&0&0&0&-1 \\
&&&&&&&&&1&0&0&0&&&&&&-1&0&0&0&0&0 \\
&&&&&&&&&0&1&0&0&&&&&&0&0&-1&0&0&0 \\
&&&&&&&&&0&0&1&0&&&&&&0&0&0&0&-1&0 \\
&&&&&&&&&0&0&0&1&&&&&&0&0&0&0&0&-1 \\
&&&&&&&&&&&&&1&0&0&0&0&-1&0&0&0&0&0 \\
&&&&&&&&&&&&&0&1&0&0&0&0&-1&0&0&0&0 \\
&&&&&&&&&&&&&0&0&1&0&0&0&0&-1&0&0&0 \\
&&&&&&&&&&&&&0&0&0&1&0&0&0&0&-1&0&0 \\
&&&&&&&&&&&&&0&0&0&0&1&0&0&0&0&-1&0 \\
&&&&&&&&&&&&&&&&&&-1&0&0&0&0&0 \\
&&&&&&&&&&&&&&&&&&0&0&-1&0&0&0 \\
&&&&&&&&&&&&&&&&&&0&0&0&0&-1&0 \\
&&&&&&&&&&&&&&&&&&0&-1&0&0&0&0 \\
&&&&&&&&&&&&&&&&&&0&0&0&-1&0&0 \\
&&&&&&&&&&&&&&&&&&0&0&0&0&0&-1
\end{array}\right]$$

Fig 5.4 Initial working basis near upper-triangular form U

The resulting factorized matrix is as follows:

$$U^X = LP^X X = \begin{bmatrix} -1 & & & & & \\ & -1 & & & & \\ & & -1 & & & \\ & & & -1 & & \\ & & & & -1 & \\ & & & & & -1 \end{bmatrix}.$$

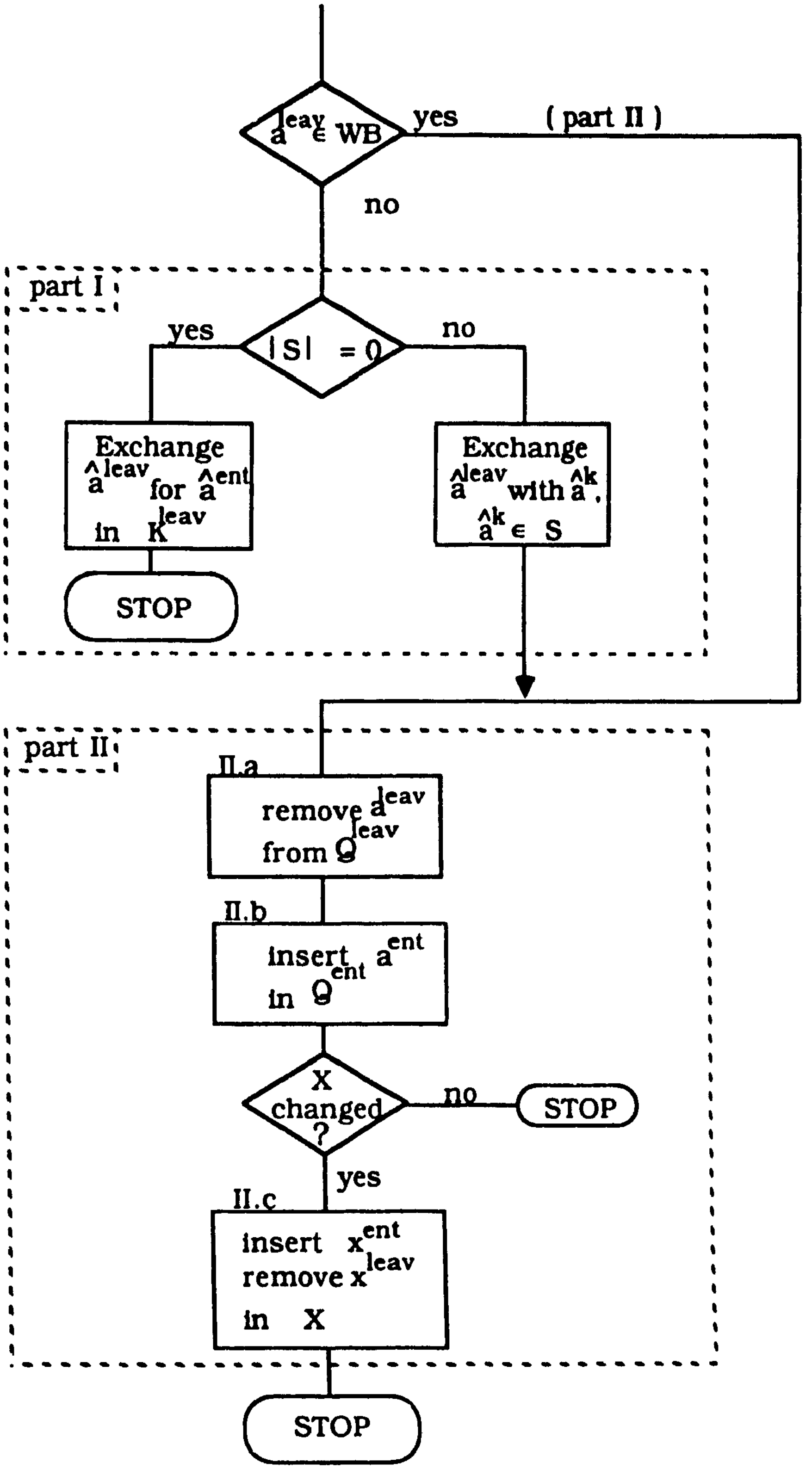
We note that the factorization of the initial WB is greatly simplified by the need for only one permutation matrix to factorize each of the $H+1$ matrices.

5.2 Update of the Factorization

At each iteration the RSM removes one of the columns in the basis B and replaces it with a better column from the constraint matrix but not currently in B. The complete basis B is composed of the key matrices K^h in the first HK columns and of the working basis WB in the last HM columns. We have decomposed the working basis into H matrices Q^h containing routes and slacks of the H periods. The capacity variables are represented by a matrix X constructed from the last M columns of the working basis.

The factorization of the working basis does not have to be done over again at each iteration [49]. The update procedure for the factorization method presented in the previous section is now given with an analysis of the computational complexity of the update. As we shall see, the update has complexity $O(M^2)$ whereas the refactorization would require a time $O(HM^3)$.

A column $a^j \in$ WB is not necessarily equal to the corresponding column in $\widehat{B}$.



$\hat{a}^{leav} \in WB$
yes
(part II)
no
part I
yes
$|S| = 0$
no
Exchange $\hat{a}^{leav}$ for $\hat{a}^{ent}$ in K^{leav}
STOP
Exchange $\hat{a}^{leav}$ with $\hat{a}^{k}$, $\hat{a}^{k} \in S$
part II
II.a
remove a^{leav} from Q^{leav}
II.b
insert a^{ent} in Q^{ent}
X changed ?
no
STOP
yes
II.c
insert x^{ent} remove x^{leav} in X
STOP

Fig. 5.5 Update procedure

In fact, we have shown in Chapter 3 that if $\hat{a}^j$ is a column representing a route then it appears in WB as $a^j = \hat{a}^j - \text{KEY}(\hat{a}^j)$ where KEY($\hat{a}^j$) is the key column associated with the origin-destination and period of route $\hat{a}^j$. From now on, we shall denote by $\hat{a}^j$ a column from the original matrix $\hat{B}$ and by $a^j \in$ WB the same column after the key column has been subtracted. The leaving column a^{leav} can be a key column as well as a column from WB. In general, the exchange of the entering column a^{ent} with a column of WB is much simpler than the exchange with a key column. For this reason *Part I* of the update consists of reducing the case where the leaving column corresponds to a key variable to the problem where it belongs to WB which is treated in *Part II* (Figure 5.5). In *Part II*, we present the details of the update of the different matrices LP^h, U^h... used in the factorization of WB for all the possible cases where a^{leav} is a column of WB. Examples where a^{ent} and a^{leav} come from the same or from different periods with same or different Origin-Destination pairs are given. Figure 5.5 gives the main steps in the update of WB, the details being presented in the following sections. In Figure 5.5, S is the set of all columns in the matrix WB representing the same Origin-Destination pair and period as the leaving column a^{leav}.

5.2.1 Part I: $\hat{a}^{leav}$ is a Key Column

The key variables in the basis B are the set of variables, one for each period and OD pair, representing a route connecting an OD pair. If a key column leaves the basis then there must be another column representing the same period and the same OD pair in WB to replace it or else the entering column necessarily replaces

$\hat{a}^{leav}$. We define S to be the set of all columns eligible to replace $\hat{a}^{leav}$ and currently correspnding to a column in WB. Thus

$$S = \{\hat{a}^j | a^j \in \text{WB}, \text{OD}(a^j) = \text{OD}(\hat{a}^{leav}), \text{PER}(a^j) = \text{PER}(\hat{a}^{leav})\}. \qquad (5.16)$$

After bringing a^{leav} into WB, we go to *Part II* which deals with the exchange of the entering column with a leaving column from WB.

5.2.1.1 |S| = 0.

This is the easiest case, and it occurs when there is no other route in WB connecting OD($\hat{a}^{leav}$) at period PER($\hat{a}^{leav}$). But since the simplex preserves the non-singularity of the basis, the entering column has to connect OD($\hat{a}^{leav}$) in period PER($\hat{a}^{leav}$) in order to be eligible to replace $\hat{a}^{leav}$ as key column. Otherwise, the row associated with this OD pair and period in the demand constraints would consist of zeroes in B, clearly a contradiction of the non-singularity of B.

Update procedure: The column $\hat{a}^{ent}$ replaces column $\hat{a}^{leav}$ in the key matrix K^{leav}. The working basis remains unchanged, and the basis update procedure is terminated for this iteration.

5.2.1.2 |S| > 0.

When there is at least one column in S which can take the place of the leaving key column, we begin the update process by bringing a^{leav} into WB in exchange for one of these columns. Three questions arise when trying to do so: What are the effects of this exchange on the factorization? Considering these effects, which column of S is the best candidate to exchange with a^{leav}? Finally, how can we efficiently restore the factorization following the exchange of the columns?

Effect on the factorization

We interchange column $\hat{a}^{leav} \in K^{leav}$, the matrix containing the key columns of period PER($\hat{a}^{leav}$), with a column $\hat{a}^{j^*} \in S$. The effect on K^{leav} is rather simple. The column $\hat{a}^{leav}$ is removed and replaced by $\hat{a}^{j^*}$. The columns in the matrix Q^{leav} are more affected. Since a^{leav} now takes the place of a^{j^*} in WB and since $\hat{a}^{j^*}$, as the new key column, must be subtracted from all corresponding routes in WB, the column $a^{j^*} = \hat{a}^{j^*} - \hat{a}^{leav}$ has to be replaced by $a^{leav} = \hat{a}^{leav} - \hat{a}^{j^*}$. In replacing the key column with $\hat{a}^{j^*}$ we must change the columns of S in WB from $\hat{a}^{j} - \hat{a}^{leav}$ to a new $a^{j} = \hat{a}^{j} - \hat{a}^{j^*}$, with the exception of column $a^{leav} = \hat{a}^{leav} - \hat{a}^{j^*}$ which is already changed. The new columns $a^{j} = \hat{a}^{j} - \hat{a}^{j^*}$ can be written in terms of the previous columns as $(\hat{a}^{j} - \hat{a}^{leav}) - (\hat{a}^{j^*} - \hat{a}^{leav})$ for all $\hat{a}^{j} \in S, \hat{a}^{j} \neq \hat{a}^{j^*}$.

The changes to the matrix Q^{leav} can be summed up as follows:

1) The key column $\hat{a}^{leav}$ is replaced by $\hat{a}^{j^*}$.

2) The column $a^{j^*} = \hat{a}^{j^*} - \hat{a}^{leav}$ is subtracted from the column $a^{j} = \hat{a}^{j} - \hat{a}^{leav}$, $\hat{a}^{j} \in S$, $a^{j} \neq a^{j^*}$.

3) The column $a^{j^*} = \hat{a}^{j^*} - \hat{a}^{leav}$ is multiplied by -1.

In Figure 5.6.a we illustrate the effect of this process on the matrix U^{leav}, the factorized form of Q^{leav}. The upper triangular form is destroyed to the left of column $j^* = 4$. The amount of work for the update could be considerable, as the elimination of the non-zero elements below the diagonal of the columns on the left of column j^*

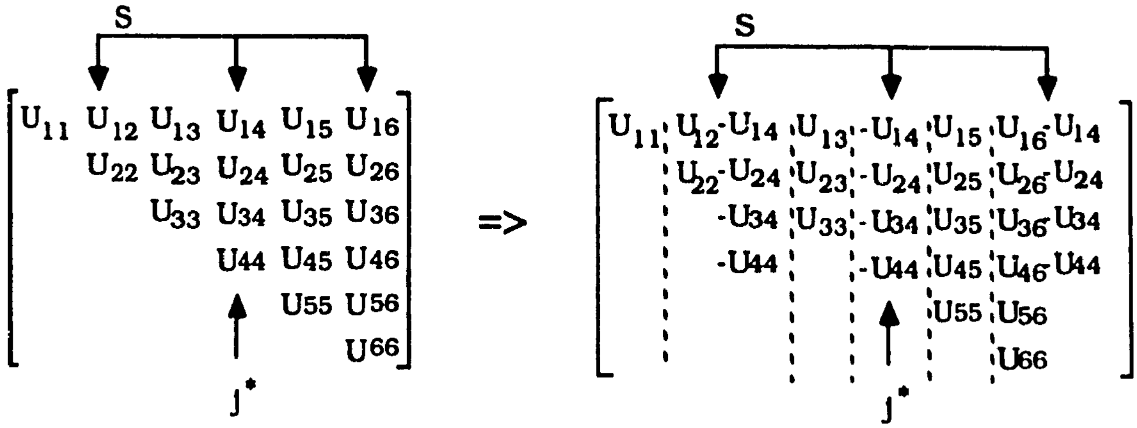


Fig. 5.6.a) Effect on the factorization of exchanging a key column

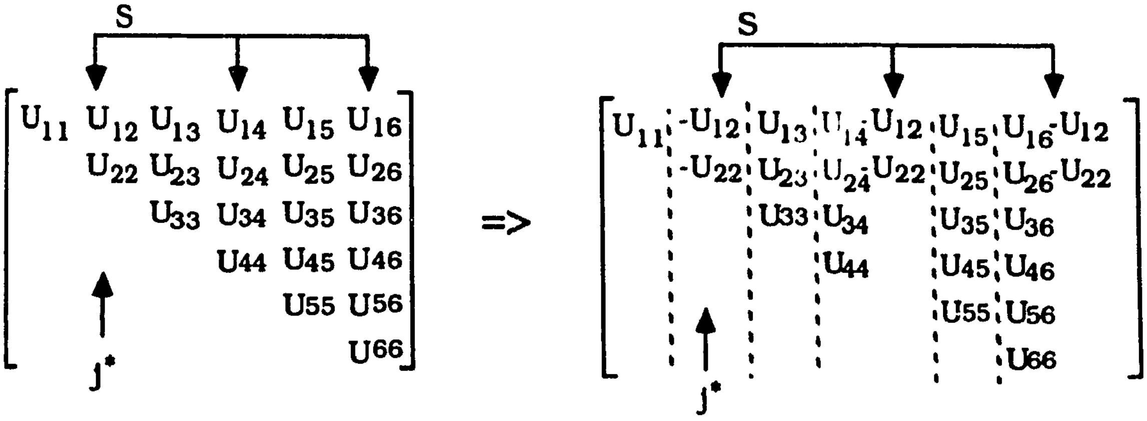


Fig. 5.6.b) Effect of the leftmost column as new key column

will create some fill-in between this column and column j^*. In the worst case, if j^* is the last column of Q^{leav} and if all columns of Q^{leav} correspond to columns of S a complete refactorization of the matrix would be necessary. Therefore, the position of a^{j^*} in Q^{leav} is the key factor in determining how much the triangular form is affected. This brings us to the second question.

How do we choose j^*? The choice of $\hat{a}^{j^*} \in$ S such that a^{j^*} is the leftmost column in the matrix Q^{leav} is the most appropriate. As we have shown, the upper triangular form is not affected by the columns to the right of position j^*. Consequently, the overall upper triangular form of U^{leav} will be preserved although the entries of the columns corresponding to columns of S are changed (see Figure 5.6.b) if we choose a^{j^*} as the leftmost column.

How can we efficiently update the factorization? This last question has already been answered above in this section. First, a^{j^*} such that $\hat{a}^{j^*} \in$ S is chosen as the leftmost column of Q^{leav}, allowing the matrix U^{leav} to keep its upper triangular form. Thus the factorization matrix $\widehat{LP}^{leav}$ does not have to be changed; only the columns of Q^{leav} and U^{leav} are changed as described earlier. All other matrices used in the factorization of WB remain unchanged. A numerical example is given in Figure 5.7. We now have a^{leav} in the working basis and the exchange of a^{ent} with a^{leav} is treated in *Part II.*

5.2.2 Part II: a^{leav} is in the Working Basis

This is the critical part of the update, where the entering and leaving columns

Before the update

$$K^{leav} = \begin{array}{c} \left[\begin{array}{ccccccc} 40 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 0 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 0 & ▨ & ▨ & ▨ & \cdots & ▨ & ▨ & ▨ \\ 40 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 40 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 40 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \end{array}\right] \\ \begin{array}{l} \uparrow \\ \hat{a}^{leav} \end{array} \end{array}$$

$$Q^1 = \begin{array}{ccccc} & \overset{S}{\downarrow} & \downarrow & & \downarrow \\ \hline 60 & 0 & -40 & 1 & -40 \\ -60 & 0 & 40 & 0 & 40 \\ -60 & 40 & 40 & 0 & 0 \\ -60 & -40 & -40 & 0 & 0 \\ 0 & -40 & 0 & 0 & -40 \\ -60 & -40 & -40 & 0 & 0 \\ \hline & \uparrow & & & \\ & \hat{a}^{j*} - \hat{a}^{leav} & & & \end{array}$$

$$\hat{U}^1 = \begin{array}{ccccc} & \overset{S}{\downarrow} & \downarrow & & \downarrow \\ \hline 60 & 0 & -40 & 1 & -40 \\ 0 & 40 & 0 & 1 & -40 \\ 0 & 0 & -80 & 2 & -80 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -80 \\ 0 & 0 & 0 & 0 & 0 \\ \hline & \uparrow & & & \\ & \widehat{LP}^{leav}(\hat{a}^{j*} - \hat{a}^{leav}) & & & \end{array}$$

After the update

$$K^{leav} = \begin{array}{c} \left[\begin{array}{ccccccc} 40 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 0 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 40 & ▨ & ▨ & ▨ & \cdots & ▨ & ▨ & ▨ \\ 0 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 0 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \\ 0 & ▨ & ▨ & ▨ & & ▨ & ▨ & ▨ \end{array}\right] \\ \begin{array}{l} \uparrow \\ \hat{a}^{j*} \end{array} \end{array}$$

$$Q^1 = \begin{array}{ccccc} & \overset{S}{\downarrow} & \downarrow & & \downarrow \\ \hline 60 & 0 & -40 & 1 & -40 \\ -60 & 0 & 40 & 0 & 40 \\ -60 & -40 & 0 & 0 & -40 \\ -60 & 40 & 0 & 0 & 40 \\ 0 & 40 & 40 & 0 & 0 \\ -60 & 40 & 0 & 0 & 40 \\ \hline & \uparrow & & & \\ & \hat{a}^{leav} - \hat{a}^{j*} & & & \end{array}$$

$$\hat{U}^1 = \begin{array}{ccccc} & \overset{S}{\downarrow} & \downarrow & & \downarrow \\ \hline 60 & 0 & -40 & 1 & -40 \\ 0 & -40 & -40 & 1 & -80 \\ 0 & 0 & -80 & 2 & -80 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -80 \\ 0 & 0 & 0 & 0 & 0 \\ \hline & \uparrow & & & \\ & \widehat{LP}^{leav}(\hat{a}^{leav} - \hat{a}^{j*}) & & & \end{array}$$

Fig. 5.7 Example of Part I

are both in WB, which plays the role of the basis in the algorithm. When the basis is not decomposed, the insertion of a^{ent} and the extraction of a^{leav} are made at the same time. In our case, however, the insertion becomes more delicate as we wish to keep the partially decomposed structure of WB, and as the two columns do not necessarily belong to the same period and consequently the same matrix Q^h.

The update is performed in three steps:

Step IIa. The column a^{leav} is removed from Q^{leav}
and the entering row $(x^{ent})^T$ for X^T is defined.

Step IIb. The column a^{ent} is introduced in Q^{ent}.
The leaving row $(x^{leav})^T$ of X^T is identified.

Step IIc. We exchange columns x^{ent} and x^{leav} (if necessary) in the matrix X
associated with the column of the capacity variables.

The complexity analysis of each step is done in each section as we present the details of the update.

5.2.2.1 Step IIa: Extraction of a^{leav} from Q^{leav}

Let a^{leav} be the k^{th} column in Q^{leav}. The update of the factorization for the matrix $\hat{U}^{leav}$ after we removed the column corresponding to a^{leav} is not much different from the standard update procedure. The main difference is that the number of columns in the matrix is reduced. The leaving column is not replaced and the number of columns is therefore reduced to $q^{leav} - 1$ as shown in Figure 5.8a. The k^{th} row in the resulting matrix is permuted to $q^{leav} - th$ position using a permutation matrix P_{J+1}^{leav} (see Figure 5.8b).

$$\hat{U}^{leav} = \begin{array}{cc} \begin{array}{ccccc} & & a^{leav}\downarrow & & \end{array} & \\ \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} & U_{15} \\ & U_{22} & U_{23} & U_{24} & U_{25} \\ & & U_{33} & U_{34} & U_{35} \\ & & & U_{44} & U_{45} \\ & & & & U_{55} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \updownarrow M \\ \longleftarrow q^{leav} \longrightarrow & \end{array} => \begin{array}{c} \begin{bmatrix} U_{11} & U_{12} & U_{14} & U_{15} \\ & U_{22} & U_{24} & U_{25} \\ & & U_{34} & U_{35} \\ & & U_{44} & U_{45} \\ & & & U_{55} \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \leftarrow q^{leav} - 1 \rightarrow \end{array}$$

Fig. 5.8 a) Extraction of a^{leav}

$$U' = P_{J+1}^{leav} \hat{U}^{leav} = \begin{bmatrix} U'_{11} & U'_{12} & U'_{13} & U'_{14} \\ & U'_{22} & U'_{23} & U'_{24} \\ & & U'_{33} & U'_{34} \\ & & & U'_{44} \\ & & U'_{53} & U'_{54} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \longleftarrow \text{row } q^{l} \\ \\ \end{array}$$

Fig. 5.8 b) Retriangularization of U^{leav}

$$\text{new } \hat{U}^{leav} = \overset{L_{J+1}^{leav}}{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha_3 & \alpha_4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}} U' = \begin{bmatrix} U'_{11} & U'_{12} & U'_{13} & U'_{14} \\ & U'_{22} & U'_{23} & U'_{24} \\ & & U'_{33} & U'_{34} \\ & & & U'_{44} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 5.8 c) Removal of line q^{leav}

Finally the non-zero entries in the row q^{leav} of $U' = P_{J+1}^{leav}\widehat{U}^{leav}$ are made zero by a matrix L_{J+1}^{leav} of the form given in Figure 5.8c, where

$$\alpha_i = \frac{U'_{q^{leav}i} + \sum_{j=k}^{i-1} \alpha_j U'_{ji}}{-U'_{ii}}, \qquad i = k, \ldots, q^{leav} - 1. \tag{5.17}$$

We now have a new matrix $\widehat{U}^{leav}$ that is upper triangular in the square matrix formed by its first $q^{leav} - 1$ rows. The matrix Q^{leav} is updated simply by removing column a^{leav}. The new factorization matrix $\widehat{LP}^{leav}$ is obtained by multiplying the previous matrix on the left by $L_{J+1}^{leav} P_{J+1}^{leav}$, that is,

$$\text{new } \widehat{LP}^{leav} = L_{J+1}^{leav} P_{J+1}^{leav} \widehat{LP}^{leav}. \tag{5.18}$$

The matrix X^T is composed of the last $M - q^h$ rows of each matrix $-\widehat{LP}^h$, which are associated with the zero rows of $\widehat{U}^h$. Since row q^{leav} of $\widehat{U}^{leav}$ is now a row of zeroes, the corresponding row of $-\widehat{LP}^{leav}$, defined as $(x^{ent})^T$, will have to be introduced in X^T eventually (see Figure 5.9).

For two reasons we must wait until the row leaving X^T is identified before we introduce $(x^{ent})^T$. The first reason is that the preservation of the square form of X^T is possible only by removing a row at the same time as the row $(x^{ent})^T$ is introduced. There is also a possibility of having both a^{ent} and a^{leav} belonging to the same period in which case $(x^{ent})^T$ might be changed in the process of introducing a^{ent}, and since we want the final row q^{leav} of $-\widehat{LP}^{leav}$ to enter X^T we have to wait until Step IIb is completed where these changes are made.

The only computation in this step is done when we multiply the matrices $\widehat{U}^{leav}$ and $\widehat{LP}^{leav}$ by the factorization matrix L_{J+1}^{leav}, which is $O(M^2)$ because only one row differs from the identity matrix.

$$\text{old } \hat{U}^{leav} = \begin{array}{c} \begin{array}{ccccc} & & \overset{a^{leav}}{\downarrow} & & \end{array} \\ \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} & U_{15} \\ & U_{22} & U_{23} & U_{24} & U_{25} \\ & & U_{33} & U_{34} & U_{35} \\ & & & U_{44} & U_{45} \\ & & & & U_{55} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \quad , \quad \text{old } \widehat{LP}^{leav} \begin{bmatrix} \text{————————} \\ \text{————————} \\ \text{————————} \\ \text{————————} \\ \text{————————} \\ \text{———} \ \epsilon X^T \ \text{———} \end{bmatrix}$$

$$\text{new } \hat{U}^{leav} = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ & U_{22} & U_{23} & U_{24} \\ & & U_{33} & U_{34} \\ & & & U_{44} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad , \quad \text{new } \widehat{LP}^{leav} \begin{bmatrix} \text{————————} \\ \text{————————} \\ \text{————————} \\ \text{————————} \\ \text{———} \ x^{ent} \ \text{———} \\ \text{———} \ \epsilon X^T \ \text{———} \end{bmatrix}$$

Fig. 5.9 Effect of removing a^{leav} on X

Example

$$Q^{leav} = \begin{bmatrix} 60 & 0 & -40 & 1 & -40 \\ -60 & 0 & 40 & 0 & 40 \\ -60 & 40 & -40 & 0 & 0 \\ -60 & -40 & 40 & 0 & 0 \\ 0 & -40 & 80 & 0 & -40 \\ -60 & -40 & 40 & 0 & 0 \end{bmatrix} \quad \text{and} \quad a^{leav} = \begin{bmatrix} 0 \\ 0 \\ 40 \\ -40 \\ -40 \\ -40 \end{bmatrix}.$$

Here $k = 2$ and $q^{leav} = 5$. The upper triangular matrix $\hat{U}^{leav}$ and factorization matrix $\widehat{LP}^{leav}$ associated with Q^{leav} are, respectively,

$$\begin{bmatrix} 60 & 0 & -40 & 1 & -40 \\ 0 & 40 & -80 & 1 & -40 \\ 0 & 0 & -80 & 2 & -80 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -80 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \} \in X^T \quad .$$

The columns corresponding to a^{leav} in Q^{leav} and $\hat{U}^{leav}$ are removed from their respective matrices and the resulting matrices are

$$Q^{leav} = \begin{bmatrix} 60 & -40 & 1 & -40 \\ -60 & 40 & 0 & 40 \\ -60 & -40 & 0 & 0 \\ -60 & 40 & 0 & 0 \\ 0 & 80 & 0 & -40 \\ -60 & 40 & 0 & 0 \end{bmatrix} \text{ and } \hat{U}^{leav} = \begin{bmatrix} 60 & -40 & 1 & -40 \\ 0 & -80 & 1 & -40 \\ 0 & -80 & 2 & -80 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -80 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The second (k^{th}) row of $\hat{U}^{leav}$ is permuted to row q^{leav} using the matrix P^{leav}_{J+1}.

$$U' = P^{leav}_{J+1}\hat{U}^{leav} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \hat{U}^{leav} = \begin{bmatrix} 60 & -40 & 1 & -40 \\ 0 & -80 & 2 & -80 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -80 \\ 0 & -80 & 1 & -40 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Row q^{leav}, is eliminated by the matrix L^{leav}_{J+1} where

$$\begin{aligned}
\alpha_1 &= -\frac{U'_{51}}{U'_{11}} = 0, \\
\alpha_2 &= -\frac{U'_{52} + \alpha_1 U'_{12}}{U'_{22}} = -1, \\
\alpha_3 &= -\frac{U'_{53} + \alpha_1 U'_{13} + \alpha_2 U'_{23}}{U'_{33}} = 1, \\
\alpha_4 &= -\frac{U'_{54} + \alpha_1 U'_{14} + \alpha_2 U'_{24} + \alpha_3 U'_{34}}{U'_{44}} = 0.5 \ .
\end{aligned} \tag{5.19}$$

The new matrix $\widehat{U}^{leav}$ is obtained by multiplying U' on the left by L^{leav}_{J+1}.

$$\text{new } \widehat{U}^{leav} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0.5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} U' = \begin{bmatrix} 60 & -40 & 1 & -40 \\ 0 & -80 & 2 & -80 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -80 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

The matrix $\widehat{LP}^{leav}$ is updated and its new row q^{leav} is defined as $(x^{ent})^T$.

$$\text{new } \widehat{LP}^{leav} = L^{leav}_{J+1} P^{leav}_{J+1} \widehat{LP}^{leav} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 0.5 & 0.5 & -1 & 0.5 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \} = x^{ent^T} \\ \} \in X^T \end{matrix} .$$

5.2.2.2 Step IIb: Addition of a^{ent} to Q^{ent}

The column a^{ent} is added as the last column of Q^{ent}. Since $\widehat{U}^{ent} = \widehat{LP}^{ent} Q^{ent}$, the effect on $\widehat{U}^{ent}$ is the addition of a column $z = \widehat{LP}^{ent} a^{ent}$. An example where $q^{ent} = 3$ and $M = 6$ is given in the Figuie 5.10a.

The matrix X^T does not always contain the last $M - q^{ent}$ rows of $-\widehat{LP}^{ent}$. It contains them only when the entering and leaving columns do not belong to the

$$Q^{ent} = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & a_1^{ent} \\ Q_{21} & Q_{22} & Q_{23} & a_2^{ent} \\ Q_{31} & Q_{32} & Q_{33} & a_3^{ent} \\ Q_{41} & Q_{42} & Q_{43} & a_4^{ent} \\ Q_{51} & Q_{52} & Q_{53} & a_5^{ent} \\ Q_{61} & Q_{62} & Q_{63} & a_6^{ent} \end{bmatrix} \quad \text{and} \quad \hat{U}^{ent} = \begin{bmatrix} U_{11} & U_{12} & U_{13} & z_1 \\ & U_{22} & U_{23} & z_2 \\ & & U_{33} & z_3 \\ & & & z_4 \\ & & & z_5 \\ & & & z_6 \end{bmatrix}$$

Fig. 5.10 a) Addition of the entering column a^{ent}

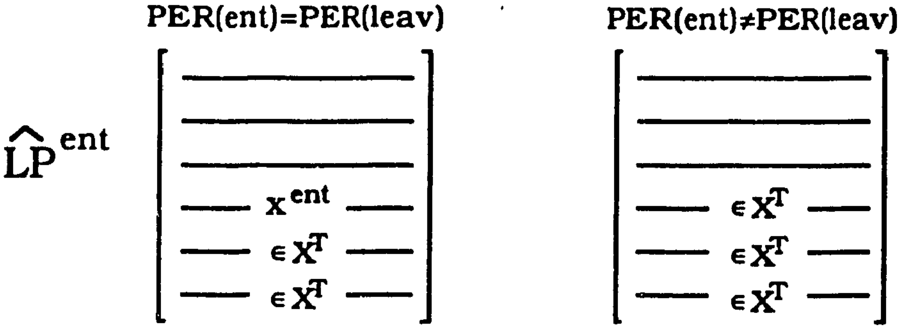


Fig. 5.10 b) Starting condition of LP^{ent}

$$\hat{U}^{ent} = \begin{bmatrix} U_{11} & U_{12} & U_{13} & z_1 \\ & U_{22} & U_{23} & z_2 \\ & & U_{33} & z_3 \\ & & & z_4 \neq 0 \\ & & & z_5 \neq 0 \\ & & & z_6 = 0 \end{bmatrix} \qquad U^x = \begin{bmatrix} U^x_{11} & U^x_{12} & U^x_{13} & U^x_{14} & U^x_{15} & U^x_{16} \\ & U^x_{22} & U^x_{23} & U^x_{24} & U^x_{25} & U^x_{26} \\ & & U^x_{33} & U^x_{34} & U^x_{35} & U^x_{36} \\ & & & U^x_{44} & U^x_{45} & U^x_{46} \\ & & & & U^x_{55} & U^x_{56} \\ & & & & & U^x_{66} \end{bmatrix}$$

column corresponding to → $z_6 = 0$ $\quad z_5 \neq 0$ $\quad z_4 \neq 0$

↑ leftmost ≠0

Fig. 5.10 c) Choice of the pivot element same period. When they are both from the same period we know that Step IIa has been executed on the same matrices as those on which Step IIb is now applied. Thus, only the last $M - q^{ent} - 1$ rows of $\widehat{LP}^{ent}$ are elements of X^T whereas the $(q^{leav} + 1)^{st}$ row defined as the entering column x^{ent} in Step IIa has not entered X^T yet. This is illustrated in Figure 5.10b.

The main objective of Step IIb is to restore the upper triangular form of U^{ent} with a minimal amount of work and minimize the effect on the other part of the factorization. Clearly the factorized matrices U^h of the other periods are not affected by the changes of U^{ent} since the matrices Q^h are totally independent from one another. On the other hand, the update of U^{ent} may affect the factorization of X because of resulting changes in those rows of $-\widehat{LP}^{ent}$ which are in X^T. In some cases the update of U^{ent} could create a complete fill-in of U^X, the matrix obtained from the factorization of X.

Choice of the Pivot Element z_j, $j \geq q^{ent} + 1$

To minimize the effect on U^X, we pivot on z_j, $j \geq q^{ent} + 1$ corresponding to the leftmost column of X and to a row of $-\widehat{LP}^{ent}$ and such that $z_j \neq 0$ (see Figure 5.10c).

<u>Case i:</u> This is the case where there are no candidates respecting the above criteria. This happens when the row corresponding to $z_{q^{ent}+1}$ is not yet in X^T, but all other $z_j = 0$, $j \geq q^{ent} + 2$. Thus, $\widehat{U}^{ent}$ is already upper triangular after the addition of the entering column and both the entering and leaving columns are from the same period. The factorization of X is unchanged since the rows of $\widehat{LP}^{ent}$ in X are not

affected in this case.

There is one more non-zero row in $\widehat{U}^{ent}$, the $(q^{ent}+1)$st one. The corresponding row in $-\widehat{LP}^{ent}$ has to be removed from X^T. We call the transpose of this row x^{leav}. Here the leaving row is the same row that we defined as the entering row x^{ent} in Step IIa. We therefore conclude that the entering and leaving rows of X^T are the same, and hence that no update of X is necessary. The working basis update is therefore terminated for this iteration. An example is given in the next section.

Case ii: There is an element respecting the above criteria, as illustrated in Figure 5.10c. The update begins by permuting the pivot element to the $(q^{ent}+1)$st row using a permutation matrix P^{ent}_{J+1}. The non-zero elements under the diagonal entry of the last column are made zero by multiplication by a matrix L^{ent}_{J+1}. This matrix differs from an identity matrix only in the non-zero values α_i, $i = q^{ent}+2\ldots M$, under the diagonal of the $(q^{ent}+1)$st column (see Figure 5.11a). We define z' as the last column of $\widehat{U}^{ent}$ after the pivot element has been permuted to the proper position by multiplying $\widehat{U}^{ent}$ on the left by P^{ent}_{J+1}. The entries α_i can now be defined as follows:

$$\alpha_i = -\frac{z'_i}{z'_{q^{ent}+1}}, \qquad i = q^{ent}+2\ldots M\,. \tag{5.20}$$

The effect on the matrix X is dictated by the changes made to the last $M - q^{ent} - 1$ rows of $\widehat{LP}^{ent}$. As we shall see, the criterion used to choose the pivot element considerably reduces the effect of the pivoting procedure on X.

The pivot procedure is in fact equivalent to multiplying the $(q^{ent}+1)$st row of $-\widehat{LP}^{ent}$, defined as $(x^{leav})^T$, by α_i and adding it to the i^{th} row, for $i = q^{ent}+2$ to

$$\text{new } \hat{U}^{ent} = \overset{L^{ent}_{J+1}}{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \alpha_5 & 1 & 0 \\ 0 & 0 & 0 & \alpha_6 & 0 & 1 \end{bmatrix}} \cdot P^{ent}_{J+1} \hat{U}^{ent} = \begin{bmatrix} U_{11} & U_{12} & U_{13} & z'_1 \\ & U_{22} & U_{23} & z'_2 \\ & & U_{33} & z'_3 \\ & & & z'_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 5.11a) Retriangularization of $\hat{U}^{ent}$

$$X = \begin{array}{c} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 - \alpha_6 x_4 \end{bmatrix} \\ \begin{array}{cccccc} & \alpha_2 = 0 & & \uparrow & & \alpha_6 \neq 0 \\ & & & \text{pivot column} & & \end{array} \end{array} , \quad U^x = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} & U_{15} & U_{16} - \alpha_6 U_{14} \\ & U_{22} & U_{23} & U_{24} & U_{25} & U_{26} - \alpha_6 U_{24} \\ & & U_{33} & U_{34} & U_{35} & U_{36} - \alpha_6 U_{34} \\ & & & U_{44} & U_{45} & U_{46} - \alpha_6 U_{44} \\ & & & & U_{55} & U_{56} \\ & & & & & U_{66} \end{bmatrix}$$

Fig. 5.11b) Effect on U^x

M. From Figure 5.11b we can see that the addition of a column in U^X to another column on the right does not affect its upper triangular form. This is exactly what happens here. The leftmost column (with $\alpha_i \neq 0$) is multiplied by the associated α_i and added to the columns corresponding to rows belonging to $-\widehat{LP}^{ent}$. The columns to the left of this column are not affected because they necessarily have $\alpha_i = 0$. The matrix X is updated by the same procedure as shown in Figure 5.11b. The only step left is the exchange of x^{ent} for x^{leav} in X which is done in Section 5.2.2.3.

The number of arithmethic operations is greater than in Step IIa but the order of complexity remains $O(M^2)$. In fact these operations are executed when $\widehat{LP}^{ent}$ is multiplied on the left by L_{J+1}^{ent} and when the columns of X and U^X are altered to reflect the changes in $\widehat{LP}^{ent}$. Clearly, the number of operations executed when multiplying a matrix by L_{J+1}^{ent} is $O(M^2)$ since the latter matrix differs from an identity matrix only in one column. When X and U^X are updated, in the worst case, we must add the column associated with the pivot element to all other columns, a $O(M^2)$ operation. Thus, the overall compiexity is $O(M^2)$.

Before we present the last step of the update of WB we give three examples for *StepIIb* : Example 1 treats the case (i), Example 2 addresses case (ii) when a^{ent} and a^{leav} are from the same period, and Example 3 considers case (ii) when a^{ent} and a^{leav} are from different periods.

<u>Example 1: case (i)</u> PER(a^{ent}) = PER(a^{leav}) and $z_i' = 0, \; i = q^{ent} + 2 \ldots M$

The matrix Q^{ent} and the entering column are, respectively,

$$Q^{ent} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } a^{ent} = \begin{bmatrix} 0 \\ 0 \\ 60 \\ 60 \\ 60 \\ 0 \end{bmatrix}$$

and the corresponding matrix $\hat{U}^{ent}$ and $\widehat{LP}^{ent}$ are

$$\hat{U}^{ent} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \widehat{LP}^{ent} = \left[\begin{array}{rrrrrr} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{l} \\ \\ = x^{ent} \\ \} \in X^T \cdot \\ \} \in X^T \\ \} \in X^T \end{array}$$

The column a^{ent} is concatenated to Q^{ent} and $\widehat{LP}^{ent} a^{ent}$ to $\hat{U}^{ent}$. The resulting $\hat{U}^{ent}$ is upper triangular and the $(q^{ent} + 1)$st row of $\widehat{LP}^{ent}$ becomes x^{leav}. Since the same row was already defined as the entering row x^{ent} there is no exchange necessary, the last $M - q^{ent} - 2$ rows of $\widehat{LP}^{ent}$ are not affected, and the matrix X remains unchanged.

$$\hat{U}^{ent} = \left[\begin{array}{rrr} 1 & 0 & 60 \\ 0 & 1 & -60 \\ 0 & 0 & 60 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \text{ and } \widehat{LP}^{ent} = \left[\begin{array}{rrrrrr} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{l} \\ \\ \\ \} \in X^T \cdot \\ \} \in X^T \\ \} \in X^T \end{array}$$

Example 2: case (ii) with $\quad \text{PER}(a^{ent}) = \text{PER}(a^{leav})$

In this case there is a z_i such that $z_i \neq 0$, $i = q^{ent} + 2 \ldots M$. Suppose that Step IIb is entered with the following matrices:

$$Q^{ent} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } a^{ent} = \left[\begin{array}{r} -60 \\ 60 \\ 0 \\ 0 \\ 60 \\ 60 \end{array}\right].$$

$$\hat{U}^{ent} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \widehat{LP}^{ent} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{l} \\ \\ = x^{ent} \\ \} \in X^T \\ \} \in X^T \\ \} \in X^T \end{array}.$$

The matrix X and its factorized form U^X are, respectively,

$$\begin{array}{c} \begin{array}{cccccc} & & z_4 & z_5 & z_6 & \end{array} \\ \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & -1 & 0 \end{bmatrix} \end{array} \text{ and } \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

Now we introduce the entering column and define the pivot element such that the upper triangular form of U^X is not altered.

$$Q^{ent} = \begin{bmatrix} 0 & 0 & -60 \\ 0 & 0 & 60 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 60 \\ 0 & 0 & 60 \end{bmatrix} \text{ and } \hat{U}^{ent} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 60 \\ 0 & 0 & 0 \\ 0 & 0 & -60 \\ 0 & 0 & 60 \end{bmatrix} \begin{array}{c} \underbrace{\text{associated with}} \\ \\ x^{ent} \\ z_4 \\ z_5 \neq 0, \text{ leftmost} \\ z_6 \end{array}.$$

Row 5 is permuted with row $q^{ent} + 1 = 3$ using the matrix P^{ent}_{J+1}. Then, the non-zero elements are made zero by the factorization matrix L^{ent}_{J+1} where $\alpha_4 = 1, \alpha_5 = 0, \alpha_6 = 1$.

$$L^{ent}_{J+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \text{ and } P^{ent}_{J+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The new matrices $\hat{U}^{ent}$ and $\widehat{LP}^{ent}$ are obtained by multiplying the previous matrices on the left by $L^{ent}_{J+1}P^{ent}_{J+1}$.

$$\hat{U}^{ent} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -60 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \widehat{LP}^{ent} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \end{bmatrix} \begin{array}{l} \\ \\ = x^{leav} \\ = x^{ent} \\ \} \in X^T \\ \} \in X^T \end{array} .$$

In X, only the columns from $-\widehat{LP}^{ent}$ have to be updated. α_i times the column associated with the pivot element, represented by z'_3 in this case, is added to the column corresponding to the entry z'_i. The same operations are carried out on the columns of U^X so that it keeps its upper triangular form. Hence, the factorization of X and LP^X does not need to be changed.

$$X = \begin{array}{c} \begin{array}{cccccc} & & z'_5 & z'_3 & z'_6 & \end{array} \\ \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}, \quad U^X = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

Example 3: case (*ii*) with $\quad \text{PER}(a^{ent}) \neq \text{PER}(a^{leav})$

This happens when there is a z_i such that $z_i \neq 0$, $i = q^{ent} + 1 \ldots M$. We enter Step IIb with the following matrices:

$$Q^{ent} = \begin{bmatrix} 0 & 0 & -60 \\ 0 & 0 & 60 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 60 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } a^{ent} = \begin{bmatrix} 30 \\ 30 \\ -30 \\ -30 \\ 0 \\ 0 \end{bmatrix}.$$

$$\hat{U}^{ent} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 60 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \widehat{LP}^{ent} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{l} \\ \\ \\ \} \in X^T \\ \} \in X^T \\ \} \in X^T \end{array} .$$

We note that the entering column of X, x^{ent}, is not involved in this case. The matrix X and its factorized form U^X are respectively

$$\begin{array}{cccccc} & z_6 & & z_5 & z_4 & \\ \left[\begin{array}{c} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right. & \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{array} & \begin{array}{c} 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ -1 \end{array} & \begin{array}{c} 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ 0 \end{array} & \begin{array}{c} -1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{array} & \left.\begin{array}{c} -1 \\ 0 \\ -1 \\ 0 \\ -1 \\ 0 \end{array}\right] \end{array} \text{ and } \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

The entering column, $\widehat{LP}^{ent} a^{ent}$ is added as the last column of $\hat{U}^{ent}$ and the pivot element defined as the non-zero z_i of this column having its corresponding column in X to the left of every other column of X representing the same period.

$$Q^{ent} = \begin{bmatrix} 0 & 0 & -60 & 30 \\ 0 & 0 & 60 & 30 \\ 1 & 0 & 0 & -30 \\ 0 & 1 & 0 & -30 \\ 0 & 0 & 60 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \hat{U}^{ent} = \begin{bmatrix} 1 & 0 & 0 & -30 \\ 0 & 1 & 0 & -30 \\ 0 & 0 & 60 & 0 \\ 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{c} \text{associated with} \\ \overbrace{\hphantom{\text{associated with}}} \\ \\ z_4 \\ z_5 \neq 0, \text{leftmost} \\ z_6 \end{array}.$$

Row 5 of $\hat{U}^{ent}$ is permuted with row $q^{ent} + 1 = 4$ using the matrix P^{ent}_{J+1} and the non-zero elements below the diagonal are made zero with the matrix L^{ent}_{J+1} where $\alpha_5 = -1, \alpha_6 = 0$.

$$L^{ent}_{J+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } P^{ent}_{J+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The new matrices $\hat{U}^{ent}$ and $\widehat{LP}^{ent}$ are obtained by multiplying the previous matrices on the left by $L^{ent}_{J+1} P^{ent}_{J+1}$ to obtain

$$\hat{U}^{ent} = \begin{bmatrix} 1 & 0 & 0 & -30 \\ 0 & 1 & 0 & -30 \\ 0 & 0 & -60 & 0 \\ 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \widehat{LP}^{ent} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{l} \\ \\ \\ = x^{leav} . \\ \} \in X^T \\ \} \in X^T \end{array}$$

The update of X and U^X is accomplished by multiplying the columns in X and U^X associated with the pivot element (z'_4) by α_i and adding them to columns associated with z'_i. The matrix U^X remains upper triangular.

$$X = \begin{array}{c} \begin{array}{cccccc} & z'_6 & & z'_4 & z'_5 & \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & -1 \\ 0 & -1 & -1 & 0 & 0 & 0 \end{bmatrix} \end{array}, \quad U^X = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

5.2.2.3 Step IIc: Exchange of x^{ent} and x^{leav}

This is the easiest of the three parts of Step II. The column x^{leav} is removed from X and x^{ent} is added to the right of X. A similar action is taken in U^X where the column $LP^X x^{leav}$ is removed and $LP^X x^{ent}$ is introduced as the last column (see Figure 5.12a).

The row of U^X corresponding to the column removed is permuted to the bottom of U^X using a permutation matrix P^X_{J+1}. As is shown in Figure 5.12b, the resulting matrix U' has an upper triangular form except for the last line. We can restore its upper triangular form by multiplying it on the left by a matrix L^X_{J+1} of the following form.

$$L^X_{J+1} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{M-1} & 1 \end{bmatrix}, \tag{5.21}$$

$$U^{x}=\begin{array}{c}
\begin{array}{cccccc} & & x^{leav} & & & \\ & & \downarrow & & & \end{array}\\
\left[\begin{array}{cccccc}
U_{11} & U_{12} & U_{13} & U_{14} & U_{15} & U_{16}\\
 & U_{22} & U_{23} & U_{24} & U_{25} & U_{26}\\
 & & U_{33} & U_{34} & U_{35} & U_{36}\\
 & & & U_{44} & U_{45} & U_{46}\\
 & & & & U_{55} & U_{56}\\
 & & & & & U_{66}
\end{array}\right]
\end{array}
\Rightarrow
\left[\begin{array}{cccccc}
U_{11} & U_{12} & U_{14} & U_{15} & U_{16} & |\\
 & U_{22} & U_{24} & U_{25} & U_{26} & |\\
 & & U_{34} & U_{35} & U_{36} & LP^{X}x^{ent}\\
 & & U_{44} & U_{45} & U_{46} & |\\
 & & & U_{55} & U_{56} & |\\
 & & & & U_{66} & |
\end{array}\right]$$

Fig. 5.12a) Exchange of LP $^{X}x^{ent}$ and LP$^{X}x^{leav}$

$$P^{X}_{J+1}\,U^{X}=\left[\begin{array}{cccccc}
U'_{11} & U'_{12} & U'_{13} & U'_{14} & U'_{15} & U'_{16}\\
 & U'_{22} & U'_{23} & U'_{24} & U'_{25} & U'_{26}\\
 & & U'_{33} & U'_{34} & U'_{35} & U'_{36}\\
 & & & U'_{44} & U'_{45} & U'_{46}\\
 & & & & U'_{55} & U'_{56}\\
 & & U'_{63} & U'_{64} & U'_{65} & U'_{66}
\end{array}\right]$$

Fig. 5.12b) Permutation of the non-uppertriangular row where

$$\alpha_i = \frac{U'_{Mi} + \sum_{j=k}^{i-1} \alpha_j U'_{ji}}{-U'_{ii}} \qquad i = k \ldots M - 1 \ , \tag{5.22}$$

and k is the position of x^{leav} in X. Note that $\alpha_i = 0$, $i = 1, \ldots, k-1$. The new matrix LP^X is determined as follows:

$$\text{new } LP^X = L^X_{j+1} P^X_{j+1} LP^X \ . \tag{5.23}$$

The update is completed with this last step. The number of operations is dominated by the computation of the α_i and by the multiplication of LP^X by L^X_{j+1} which are both $O(M^2)$.

Example

$$X = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix} , \ x^{leav} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} , \ x^{ent} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} .$$

The upper triangular matrix obtained from the factorization of X is

$$U^X = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} .$$

The column corresponding to x^{leav} is removed from X and U^X and replaced by x^{ent} and $LP^X x^{ent}$ as the last column in these respective matrices.

$$X = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} , \ U^X = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & 0.5 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} .$$

The 4th row of U^X is permuted to the bottom by P^X_{J+1}.

$$U' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} U^X = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0.5 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 \end{bmatrix}.$$

Then we multiply U' on the left by L^X_{J+1} where $\alpha_1 = \alpha_2 = \alpha_3 = 0$, $\alpha_4 = 2$ and $\alpha_5 = -2$. The result is a new upper triangular matrix U^X.

$$U^X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & -2 & 1 \end{bmatrix} U' = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0.5 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}.$$

Finally LP^X is updated by multiplying it on the left by $L^X_{J+1}P^X_{J+1}$.

5.3 Refactorization

This last section addresses the problem of refactorization made necessary by the accumulation of round-off errors. After many changes to the factorization of a matrix B, the accumulated round-off errors may become non-negligeable and the refactorization of B from scratch becomes necessary. The number of changes can be approximated by the number of matrices L_j used to generate LP such that $LP\ B = U$. We have previously defined LP as $LP = L_J P_J \dots L_1 P_1$, so J would be the number of changes currently used to transform B into U.

Obviously, the optimal number of changes v giving the best trade-off of speed versus accuracy cannot be computed with precision. In fact it appears to be difficult to find a good theoretical approximation. Nevertheless a value of v between 20 and 40 has given good results in practice. Chvátal [42] has mentioned that "the practical success of this policy has been firmly established". From now on we assume that v has a constant value around 30.

If we had ignored the structure of the working basis (size $HM \times HM$), then its refactorization would have a number of operations proportional to H^3M^3 at every 30 iterations for a complexity of $O(H^3M^3)$. The method of factorization presented in this section allows for a faster refactorization. In the worst case, three of the $H+1$ matrices (size $(q^h \leq M) \times M$) composing the factorization of WB ($Q^{ent}, Q^{leav},$ *and* X) change per iteration. Hence a minimum of $\frac{v(H+1)}{3}$ iterations are necessary before each matrix is changed v times. The work involved in refactorizing $(H+1)$ matrices of size M or less is $O(HM^3)$. Since v is a constant, on the average the work involved per iteration for refactorization when decomposition is used is $O(\frac{HM^3}{H+1}) = O(M^3)$.

Chapter 6 Solving $(\pi,\mu)B = \$^B$ and $B\sigma = \hat{a}^{ent}$

An efficient solution of the two linear systems involved in the simplex algorithm, $(\pi,\mu)B = \$^B$ in Step 1 and $B\sigma = \hat{a}^{ent}$ in Step 3, is the reason that has motivated the use of a factorization to represent the working basis WB. The matrix WB appears in the top right-hand corner of the matrix obtained when the basis B is multiplied by the matrix T in the GUB technique. That is,

$$BT = \begin{bmatrix} \mathrm{K} & \hat{B} \\ \mathrm{I} & C \end{bmatrix} \begin{bmatrix} \mathrm{I} & -C \\ 0 & \mathrm{I} \end{bmatrix} = \begin{bmatrix} \mathrm{K} & \mathrm{WB} \\ \mathrm{I} & 0 \end{bmatrix}, \qquad (6.1)$$

where $WB = \hat{B} - KC$. We have seen in Chapter 3 that the solution of the two linear systems amounts to solving the two subsystems using only the matrix WB: $\pi\mathrm{WB} = \$^{\mathrm{WB}}$ and $\mathrm{WB}\sigma_2 = a^{ent}$, where a^{ent} is composed of the first HM elements of the vector $\hat{a}^{ent} - \mathrm{KEY}(\hat{a}^{ent})$. The details of each subsystem as well as the method of solving them using the factorization of WB are presented in the following sections.

6.1 Preliminary Notions

In Chapter 5, WB is factorized from its original form to a near upper triangular form U having H upper triangular matrices U^h and a matrix X^T on the diagonal. This is accomplished by multiplying the matrix WB on the left by two matrices, V

and LP, such that $V\ LP\ \mathrm{WB} = U$. Thus

$$U = V\ LP\ \mathrm{WB} = \begin{bmatrix} U^1 & & & & -LP^1 \\ & U^2 & & & -LP^2 \\ & & \ddots & & \vdots \\ & & & U^H & -LP^H \\ & & & & X^T \end{bmatrix}, \tag{6.2}$$

where

$$\mathrm{WB} = \begin{bmatrix} Q^1 & & & -I \\ & Q^2 & & -I \\ & & \ddots & \vdots \\ & & Q^H & -I \end{bmatrix}. \tag{6.3}$$

Then the matrix X, the transpose of X^T, is factorized into U^X using the matrix LP^X. The solution of the two linear systems involving WB is based on the same decomposition principle that allowed the simplification of the update process. In both cases, the matrix X plays the key role. In fact, after the decomposition of WB, X acts as the basis of the RSM. A similar situation occured in the GUB where WB took the role of the basis, replacing the matrix B. In the following section we present the method of solving the two linear systems with the general triangular factorization method. Then we present the method using the decomposed factorization along with the complexity analysis of each step.

6.1.1 Solution Using the Standard Factorization Method Without Decomposition

In this section we look at the solution of the two linear systems $\pi \mathrm{WB} = \$^{\mathrm{WB}}$ and $\mathrm{WB}\sigma_2 = a^{ent}$ using a triangular factorization applied directly to WB. The matrix WB of size $HM \times HM$ is factorized by a sequence of matrices L_j and P_j into a near upper-triangular matrix U. The factorization matrix is defined as

$$LP = L_J P_J \ldots L_1 P_1 \tag{6.4}$$

and the relation between LP, WB and U is given by

$$LP\ \mathrm{WB} = U. \tag{6.5}$$

6.1.2 Solution of $\pi\mathrm{WB} = \$^{\mathrm{WB}}$

The solution of this system is carried out in three steps. First, there is a transformation of variables using the fact that LP is non-singular. Thus

$$\pi = z\ LP. \tag{6.6}$$

After the transformation, and using Equation 6.5, the system becomes

$$z\ U = \$^{\mathrm{WB}}. \tag{6.7}$$

The solution of this system is relatively easy since U is upper triangular. The number of operations necessary for solving an $HM \times HM$ system based on an upper triangular matrix is proportional to H^2M^2, thus resulting in a complexity of $O(H^2M^2)$. The row vector π is then obtained by substituing z in Equation 6.6 which is also of complexity $O(H^2M^2)$.

6.1.3 Solution of $\mathrm{WB}\sigma_2 = a^{ent}$

The first step is the multiplication of both sides of the system by the factorization matrix LP. Using Equation 6.5, the following system is obtained

$$U\sigma_2 = z', \tag{6.8}$$

where

$$z' = LP\ a^{ent}. \tag{6.9}$$

The multiplication of the matrix LP by a vector is $O(H^2M^2)$ and the solution of a linear system based on an upper triangular matrix is also $O(H^2M^2)$.

Even though the solution of the two linear systems in the GUB is made relatively efficient by the use of a triangular factorization, the structure of WB is not exploited by such a procedure. The overall complexity of the two steps is currently $O(H^2M^2)$. We shall demonstrate in the next two sections that it is possible to reduce the complexity by decomposing the factorized matrix U shown in Equation 6.2.

6.2 Solution of $(\pi,\mu)B = \$^B$ Using the Decomposed Factorization

The solution of $(\pi,\mu)B = \$^B$ corresponds to Step 1 of the revised simplex method. The row vectors π and μ, respectively of length HM and HK, are the dual variables associated with the formulation given in Equation 2.8. The right-hand side of the equation is a row vector of length $HK + HM$ in which the i^{th} element is the cost incurred when the variable associated with the i^{th} column of B is increased by one unit. These unit costs are readily available from the objective function of the problem. $\$^B$ can be divided into two parts, the first $HK + (H-1)M$ elements are zeroes since only the capacity variables have non-zero unit costs. The last M elements which are associated with the capacity variables have a unit cost of $\$_m$, $m = 1 \ldots M$. Thus,

$$(\pi,\mu)B = (\pi,\mu)\begin{bmatrix} \mathrm{K} & \widehat{B} \\ \mathrm{I} & C \end{bmatrix} = (0,\ldots,0,\ \$_1 \ldots \$_M). \tag{6.10}$$

The first step in the solution of Equation 6.10 consists of a transformation of the matrix B in order to eliminate the matrix C in the bottom right-hand corner. This is acheived by a post-multiplication of B by the matrix T as in Equation 6.1 to obtain the required form. When both sides of Equation 6.10 are multiplied on the right by T the result is a linear system in which the matrix BT has a stucture that can be easily exploited. The right-hand side is not affected by T since its last M rows contains only an identity matrix and only the last M elements of $\$^B$ are different from zero. Thus,

$$\begin{gathered} (\pi,\mu)BT = \$^B, \\ (\pi,\mu)\begin{bmatrix} \mathrm{K} & \mathrm{WB} \\ \mathrm{I} & 0 \end{bmatrix} = (0,\ldots,0,\ \$_1 \ldots \$_M). \end{gathered} \tag{6.11}$$

This last system can be divided into two smaller systems:

$$\pi\mathrm{WB} = \$^{\mathrm{WB}} = (0,\ldots,0,\ \$_1 \ldots \$_M), \tag{6.12}$$

and

$$\mu = -\pi\mathrm{K} = -\pi\begin{bmatrix} \mathrm{K}^1 & & & \\ & \mathrm{K}^2 & & \\ & & \ddots & \\ & & & \mathrm{K}^H \end{bmatrix}. \tag{6.13}$$

6.2.1 Solution of πWB = $\$^{\mathrm{WB}}$

Equation 6.12, written more explicitly, is

$$\pi \begin{bmatrix} Q^1 & & & & -I \\ & Q^2 & & & -I \\ & & \ddots & & \vdots \\ & & & Q^H & -I \end{bmatrix} = \$^{\mathrm{WB}} \tag{6.14}$$

and the solution of Equation 6.14 is the essential part of Step 1. The use of the matrix U, obtained by the factorization procedure of Section 5.1, allows for an efficient solution technique. Invertibility of the matrix $V\ LP$ allows us to define a new row vector π_U of length HM such that

$$\pi = \pi_U V\ LP. \tag{6.15}$$

Since $U = V\ LP$ WB, the following system is obtained when Equation 6.15 is substituted in 6.14:

$$\pi_U V\ LP\ \mathrm{WB} = \pi_U \begin{bmatrix} U^1 & & & & -LP^1 \\ & U^2 & & & -LP^2 \\ & & \ddots & & \vdots \\ & & & U^H & -LP^H \\ & & & & X^T \end{bmatrix} = \$^{\mathrm{WB}}, \tag{6.16}$$

where $\$^{\mathrm{WB}} = (0,\ldots,0,\ \$_1 \ldots \$_M)$. We rewrite the vector π_U as (π_U^Q, π_U^X) where π_U^Q represents the first $(H-1)M$ elements and π_U^X, the last M elements of π_U. From Equation 6.16, the triangularity of the U^h, and the definition of $\$^{\mathrm{WB}}$, we conclude that $\pi_U^Q = 0$ and that the solution of linear system 6.16 reduces to solving

$$\pi_U^X X^T = \$^X, \tag{6.17}$$

where $\$^X = (\$_1 \ldots \$_M)$. In fact, we will solve the transpose of this system since we have factorized X instead of X^T.

$$X \pi_U^{X^T} = \$^{X^T}. \tag{6.18}$$

The $M \times M$ matrix X is triangularized into U^X by a multiplication on the left by LP^X. The right-hand side becomes $LP^X \$^{X^T}$, a column vector of length M. The resulting system can be solved easily because of the upper triangular form of U^X.

$$U^X \pi_U^{X^T} = LP^X \$^{X^T}. \tag{6.19}$$

Consequently, the row vector π_U is equal to zero in its first $(H-1)M$ elements and π_U^X, the solution of 6.19, in the last M elements.

$$\pi_U = (0, \ldots, 0, \pi_U^X). \tag{6.20}$$

In the next step we apply Equation 6.15 in order to obtain π from π_U. Thus

$$\pi = \pi_U \; V \; LP = (0, \ldots, 0, \pi_U^X) \; V \; LP. \tag{6.21}$$

The effect of multiplying LP by V is similar to the effect on the last M columns of LP WB in the factorization process; the last $M - q^h$ rows of each $\widehat{LP}^h$ are permuted to form the last M rows of the resulting matrix. In Chapter 5 we defined the remaining q^h rows of $\widehat{LP}^h$ as LP^h. The last $M - q^h$ rows that were permuted to the bottom are now in separate matrices called X^h.

$$V \; LP = \begin{bmatrix} LP^1 & & & \\ & LP^2 & & \\ & & \ddots & \\ & & & LP^H \\ X^1 & X^2 & \cdots & X^H \end{bmatrix}. \tag{6.22}$$

The matrices X^h are composed of q^h rows of zeroes and of $M - q^h$ rows from $\widehat{LP}^h$ which appear in the same order as in X^T. Considering the structure of $V \; LP$ and π_U, the resulting row vector $\pi = \pi_U \; V \; LP$ depends only on π_U^X and the matrices X^h. Thus

$$\pi = (\pi_U^X X^1, \pi_U^X X^2, \ldots, \pi_U^X X^H). \tag{6.23}$$

In practice, we would like to avoid the creation of the H matrices X^h of size $M \times M$. This is possible since all the information contained in the matrices X^h is available in X^T and since the non-zero rows of X^h are in the the same positions as in X^T. Consequently, it is much more efficient to use the matrix X^T and to create H M-vectors π_U^h defined as follows

$$\pi_U^h = \begin{cases} (\pi_U^X)m, & \text{if row } m \text{ of } X^T \in -\widehat{LP}^h, \\ 0, & \text{otherwise,} \end{cases} \tag{6.24}$$

for $m = 1 \ldots M$. Finally, the row vector π can be computed using the equation

$$\pi = (\pi_U^1 X^T, \pi_U^2 X^T, \ldots, \pi_U^H X^T). \tag{6.25}$$

This procedure is illustrated in Figure 6.1

The dual variables μ associated with the demand constraints are related to π according to Equation 6.13 by $\mu = -\pi \mathrm{K}$. The matrix K is a block diagonal matrix composed of the key matrices K^h on the diagonal. Considering the result in Equation 6.25, the product $-\pi \mathrm{K}$ is given by

$$\mu = -((\pi_U^1 X^T)\mathrm{K}^1, (\pi_U^2 X^T)\mathrm{K}^2, \ldots, (\pi_U^H X^T)\mathrm{K}^H). \tag{6.26}$$

Complexity analysis

When solving the system $(\pi, \mu)B = \$^B$ it is not necessary to carry out all the steps presented in this section.The important steps can be summarized as follows:

1) Compute the column vector $LP^X \$^{X^T}$.

2) Solve the system $U^X \pi_U^{X\,T} = LP^X \$^{X^T}$.

3) Establish the vectors π_U^h, $h = 1 \ldots H$.

4) Compute $\pi = (\pi_U^1 X^T, \pi_U^2 X^T \ldots \pi_U^H X^T)$.

5) Compute $\mu = -\pi \mathrm{K}$.

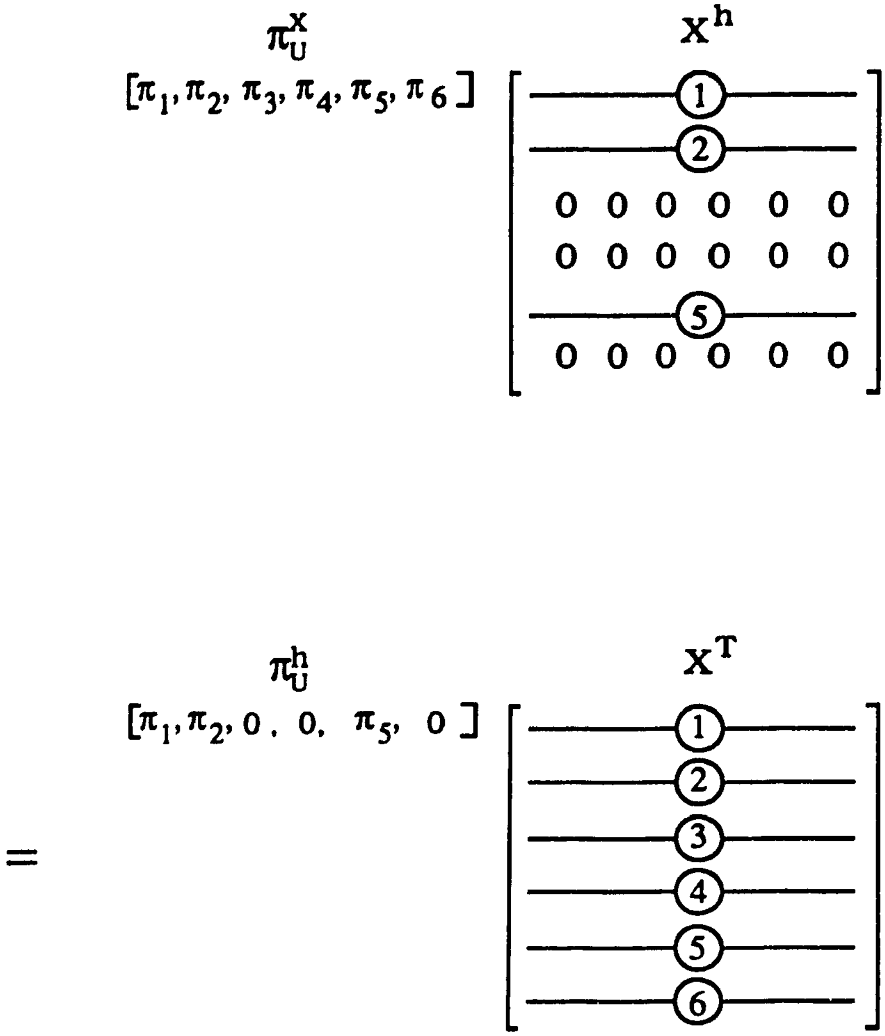

π_U^x
$[\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6]$
X^h
1
2
0 0 0 0 0 0
0 0 0 0 0 0
5
0 0 0 0 0 0
=
π_U^h
$[\pi_1, \pi_2, 0 . 0, \pi_5, 0]$
X^T
1
2
3
4
5
6

Fig. 6.1 Replacement of π_U^x by π_U^h

The multiplication of an $M \times M$ matrix by a vector of length M in 1) requires a number of multiplications proportional to M^2. Part 2), the solution of a linear system based on an upper triangular matrix is of complexity $O(M^2)$. The distribution of the M elements of π_U^X to the row vectors π_U^h in 3) is $O(M)$. Parts 4) and 5) are the dominant calculations; they each require H multiplications of a row vector of length M by an $M \times M$ and an $M \times K$ matrix respectively, for complexities of order $O(HM^2)$ and $O(HMK)$. But since in a rearrangeable network it is required that the ratio of the number of links M to its maximal number $K = \frac{N(N-1)}{2}$ be moderately high to allow for a sufficient number of rearrangement patterns, we have that $M = O(N^2)$ and consequently $M = O(K)$ for a complexity of $O(HM^2)$ in part 5). Thus, the overall complexity of Step 1 of the RSM using the decomposed factorization is $O(HM^2)$.

Example

We present in this section an example in which we solve the system $\pi\mathrm{WB} = \$^{\mathrm{WB}}$. The working basis considered is taken from a problem having $H = 4$ periods, $M = 6$ arcs, and $K = 10$ origin-destinations. The matrices Q^h composing WB, the matrix X, the factorization matrices LP^h and LP^X, and the resulting upper triangular matrices U^h and U^X are given in Figure 6.2.

The first step is the multiplication of LP^X by $\$^{X^T}$. If $\$^X = (1,1,1,1,1,1)$ then

$$LP^X \$^{X^T} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 2 \\ -1 \\ 1 \end{bmatrix}.$$

$$Q^1 = \begin{bmatrix} 0 & 80 & 0 & 80 \\ 0 & 80 & 0 & 80 \\ 0 & -80 & -10 & 0 \\ 0 & 80 & 10 & 0 \\ 0 & 0 & -10 & -80 \\ 1 & 80 & 10 & 0 \end{bmatrix}$$

$$\widehat{LP}^1 \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$\hat{U}^1 \left[\begin{array}{cccc} 1 & 80 & 10 & 0 \\ 0 & 80 & 0 & 80 \\ 0 & 0 & -10 & 80 \\ 0 & 0 & 0 & 160 \\ \hdashline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$Q^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 40 \\ 0 & 0 & 0 & 0 & -40 \\ 0 & 1 & 0 & 90 & 0 \\ 0 & 0 & 0 & 90 & 0 \\ 0 & 0 & 1 & 90 & 40 \\ 0 & 0 & 0 & -90 & 0 \end{bmatrix}$$

$$\widehat{LP}^2 \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$\hat{U}^2 \left[\begin{array}{ccccc} 1 & 0 & 0 & 0 & 40 \\ 0 & 1 & 0 & 90 & 0 \\ 0 & 0 & 1 & 90 & 40 \\ 0 & 0 & 0 & 90 & 0 \\ 0 & 0 & 0 & 0 & -40 \\ \hdashline 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$\begin{array}{rcc:c:ccc} \text{period} & \multicolumn{2}{c}{1} & 2 & \multicolumn{3}{c}{4} \\ & 0 & 1 & 0 & -1 & 0 & 0 \\ & 0 & -1 & 0 & 0 & -1 & 0 \\ & -1 & 0 & 0 & 0 & 0 & 0 \\ X & -1 & 0 & -1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & -1 & 1 & 0 \\ & 0 & 0 & -1 & 0 & 0 & -1 \end{array}$$

$$Q^3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 50 \\ 0 & 0 & 0 & 0 & 0 & -50 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 50 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\widehat{LP}^3 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\hat{U}^3 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 50 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 50 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -50 \end{bmatrix}$$

$$LP^X \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}$$

$$U^X \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$Q^4 = \begin{bmatrix} 0 & 0 & -60 \\ 0 & 0 & 60 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 60 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\widehat{LP}^4 \left[\begin{array}{cccccc} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\hat{U}^4 \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 60 \\ \hdashline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

Fig. 6.2 Factorized working basis

We then solve the system

$$U^X \pi_U^X = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \pi_U^X = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 2 \\ -1 \\ 1 \end{bmatrix},$$

for which $\pi_U^X = (-1, -0.5, 0, -1.5, -0.5, -1)$. From π_U^X we can establish the π_U^h vectors.

$$\pi_U^1 = \begin{bmatrix} 1 \\ -0.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \pi_U^2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \pi_U^3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \pi_U^4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1.5 \\ -0.5 \\ -1 \end{bmatrix}.$$

Then, the row vector π is computed using Equation 6.25,

$$\begin{aligned} \pi &= (\pi_U^1 X^T, \pi_U^2 X^T, \pi_U^3 X^T, \pi_U^4 X^T), \\ &= (0.5, -0.5, -1, -1, 0, 0,\ 0, 0, 0, 0, 0, 0,\ 0, 0, 0, 0, 0, 0,\ -1.5, -0.5, 0, 0, -1, -1). \end{aligned}$$

6.3 Solution of $B\sigma = \widehat{a}^{ent}$ Using the Decomposed Factorization

The column $\widehat{a}^{ent}$ is either a route or a slack variable from period PER($\widehat{a}^{ent}$). The top HM elements are associated with the capacity constraints and the last HK elements with the demand constraints If the entering column $\widehat{a}^{ent}$ is a route then M of the first HM elements of $\widehat{a}^{ent}$ represent a route as described in the *arc-chain* formulation. The last HK elements are all zeroes, except for the row corresponding to OD($\widehat{a}^{ent}$) and PER($\widehat{a}^{ent}$) which contains 1. In the case where $\widehat{a}^{ent}$ is a slack variable there is one non-zero element in the first HM rows, a 1, associated with the arc and period of the slack variable. Since the slack variables are not associated with any demand constraint, their last HK elements are zeroes.

The main objective of the third step of the simplex is to find the representation of the entering column in terms of the columns forming the basis which is done by solving the linear system

$$B\sigma = \widehat{a}^{ent}. \tag{6.27}$$

In order to reduce the problem to the solution of a linear system using only the working basis WB, GUB prescribes a transformation of variables such that

$$\sigma = T\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix} = \begin{bmatrix}\mathrm{I} & -C\\ 0 & \mathrm{I}\end{bmatrix}\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix}. \tag{6.28}$$

By inserting 6.28 in Equation 6.27 we obtain

$$\begin{aligned} BT\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix} &= \widehat{a}^{ent},\\ \begin{bmatrix}\mathrm{K} & \mathrm{WB}\\ \mathrm{I} & 0\end{bmatrix}\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix} &= \widehat{a}^{ent} \end{aligned}. \tag{6.29}$$

From the last HK equations we can see that σ_1 is equal to the last HK elements of $\widehat{a}^{ent}$. These elements are all zeroes except in the case where $\widehat{a}^{ent}$ is a route. In the case of a route there is a single 1 indicating with which period and with which OD-pair the column $\widehat{a}^{ent}$ is associated.

$$\sigma_1 = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \text{ } or \text{ } 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{6.30}$$

The first HM constraints contain the system $\mathrm{K}\sigma_1 + \mathrm{WB}\sigma_2 = (\widehat{a}_1^{ent} \ldots \widehat{a}_{HM}^{ent})^T$, where $(\widehat{a}_1^{ent} \ldots \widehat{a}_{HM}^{ent})^T$ is a column vector composed of the first HM elements of $\widehat{a}^{ent}$. The column vector σ_1 is defined in Equation 6.30 and the resulting linear system is given by

$$\mathrm{WB}\sigma_2 = a^{ent}, \tag{6.31}$$

where $a^{ent} = (\widehat{a}_1^{ent} \ldots \widehat{a}_{HM}^{ent})^T - \mathrm{K}\sigma_1$. The column vector a^{ent} of length HM is defined as in Chapter 2. If $\widehat{a}^{ent}$ is a route then $a^{ent} = (\widehat{a}_1^{ent} \ldots \widehat{a}_{HM}^{ent})^T - \mathrm{KEY}(\widehat{a}^{ent})$ and if $\widehat{a}^{ent}$ is a slack variable then $a^{ent} = (\widehat{a}_1^{ent} \ldots \widehat{a}_{HM}^{ent})^T$. Notice that $\mathrm{KEY}(\widehat{a}^{ent})$ is the key column for period $PER(\widehat{a}^{ent})$ and OD-pair $\mathrm{OD}(\widehat{a}^{ent})$.

6.3.1 Solution of $\mathrm{WB}\sigma_2 = a^{ent}$

We have reduced the problem of solving $B\sigma = \widehat{a}^{ent}$ in which B has size $H(M + K) \times H(M + K)$, to a smaller problem $\mathrm{WB}\sigma_2 = a^{ent}$, where WB is $HM \times HM$. We shall now use the factorization method presented in Chapter 5 to decompose the problem into $H + 1$ subproblems of size $M \times M$ or smaller. First, we rewrite the column vector σ_2 as

$$\sigma_2 = \begin{bmatrix} \sigma^1 \\ \vdots \\ \sigma^H \\ \sigma^X \end{bmatrix}. \tag{6.32}$$

where σ^h, $h = 1 \ldots H$, are column vectors of length q^h and σ^X is a column vector of length M. Since a^{ent} represents a route or a slack variable, only the elements of the M-vector a associated with the period of the entering column are non-zero.

$$a^{ent} = \begin{bmatrix} 0_{M\times 1} \\ \vdots \\ a \\ \vdots \\ 0_{M\times 1} \end{bmatrix}. \tag{6.33}$$

Equation 6.31 now becomes

$$\text{WB} \begin{bmatrix} \sigma^1 \\ \vdots \\ \sigma^H \\ \sigma^X \end{bmatrix} = \begin{bmatrix} 0_{M\times 1} \\ \vdots \\ a \\ \vdots \\ 0_{M\times 1} \end{bmatrix}. \tag{6.34}$$

The factorization process starts by pre-multiplication of WB by the matrix LP and then by the permutation matrix V.

$$V\ LP\ \text{WB} \begin{bmatrix} \sigma^1 \\ \vdots \\ \sigma^H \\ \sigma^X \end{bmatrix} = V \begin{bmatrix} 0_{M\times 1} \\ \vdots \\ \widehat{LP}^{ent}a \\ \vdots \\ 0_{M\times 1} \end{bmatrix}. \tag{6.35}$$

where V LP WB is, in fact, the near upper triangular matrix U. When the matrix V multiplies the column vector on the right-hand side of Equation 6.35; the last $M - q^h$ elements of each group are permuted to the last M rows in the same order as the rows of the $-\widehat{LP}^h$ matrices are permuied to form the matrix X^T. We call the first q^h elements of $\widehat{LP}^{ent}a$ as a' and the last M elements, formed from the last $M - q^h$ elements of each period, are called a''. Thus, Equation 6.35 can be written as

$$\begin{bmatrix} U^1 & & & & & -LP^1 \\ & \ddots & & & & \vdots \\ & & U^{ent} & & & -LP^{ent} \\ & & & \ddots & & \vdots \\ & & & & U^H & -LP^H \\ & & & & & X^T \end{bmatrix} \begin{bmatrix} \sigma^1 \\ \vdots \\ \sigma^{ent} \\ \vdots \\ \sigma^H \\ \sigma^X \end{bmatrix} = \begin{bmatrix} 0_{q^1 \times 1} \\ \vdots \\ a' \\ \vdots \\ 0_{q^H \times 1} \\ a'' \end{bmatrix}. \tag{6.36}$$

The first system we solve is $X^T \sigma^X = a''$, for which the transpose is given by

$$\sigma^{X^T} X = a''^T. \tag{6.37}$$

LP^X being invertible, we can make the change of variables

$$\sigma^{X^T} = zLP^X. \tag{6.38}$$

When Equation 6.38 is substituted in Equation 6.37 the linear system becomes

$$zLP^X X = zU^X = a''^T \tag{6.39}$$

which can be easily solved since U^X is upper triangular. Its solution is then substituted back into Equation 6.38 to obtain σ^X. The remainder of the system is solved as follows.

$$\begin{aligned} U^h \sigma^h &= LP^h \sigma^X \qquad h = 1 \ldots H, h \neq \mathrm{PER}(a^{ent}), \\ U^h \sigma^h &= a' + LP^h \sigma^X \qquad h = \mathrm{PER}(a^{ent}). \end{aligned} \tag{6.40}$$

In each of these linear systems the matrix is upper triangular. We now have that σ_2 is composed of the column vectors σ^X and $\sigma^h, h = 1 \ldots H$.

Finally, σ is obtained from Equation 6.28.

$$\sigma = \begin{bmatrix} I & -C \\ 0 & I \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix}. \tag{6.41}$$

Like the columns of Q^h in WB, the columns of C are grouped by period in submatrices C'^h of size $K \times q^h$. Since the last M columns of WB are associated with the capacity variables and since they are independent of the demand constraints, the last M columns of C' contain only zero values.

$$C = \begin{bmatrix} C'^1 & & & & 0_{K \times M} \\ & C'^2 & & & 0_{K \times M} \\ & & \ddots & & \vdots \\ & & & C'^H & 0_{K \times M} \end{bmatrix}. \tag{6.42}$$

Thus σ is given by

$$\begin{aligned} (\sigma_{(h-1)K+1} \dots \sigma_{hK}) &= (\sigma_{1_{(h-1)K+1}} \dots \sigma_{1_{hK}}) - C'^h \sigma^h, \quad h = 1 \dots H, \\ (\sigma_{HK+1} \dots \sigma_{HK+HM}) &= \sigma_2. \end{aligned} \tag{6.43}$$

Complexity analysis

We first summarize the important steps of the solution of the linear system $B\sigma = \hat{a}^{ent}$.

1) Establish a' and a'' from the product $LP^{ent}a$.

2) Solve the linear system $zU'^X = a''^T$ in order to obtain $\sigma^X = (zLP^X)^T$.

3) Solve the H linear systems

$$\begin{aligned} U^h \sigma^h &= LP^h \sigma^X, \qquad h = 1 \dots H, h \neq PER(a^{ent}), \\ U^h \sigma^h &= a' + LP^h \sigma^X, \qquad h = PER(a^{ent}). \end{aligned}$$

4) Establish σ_1 from the vector a^{ent}, and σ_2 from σ^h, and σ^X.

5) Compute σ from σ_1 and σ_2 as in Equation 6.43.

1) requires a number of operations proportional to M^2 to find the product $LP^{ent}a$. In 2), the solution of the linear system and the multiplication of the solution by the matrix LP^X are both of order $O(M^2)$. In 3), we solve H linear systems for a complexity of $O(HM^2)$. In 5), the last HM elements of σ are obtained directly from

σ_2 whereas the computation of the first HK elements necessitates H multiplications of a $K \times q^h$, $(q^h \leq M)$ matrix C^h by a vector of length q^h resulting in a complexity of $O(HKM)$. Since both K and M are of order $O(N^2)$, we have that K is of order $O(M)$; thus the complexity of 5) is $O(HM^2)$. The overall complexity of finding the representation of $\hat{a}^{ent}$ in terms of the columns in B is dictated by 3) and 5), and is of order $O(HM^2)$.

Example of the Solution of $\mathrm{WB}\sigma_2 = a^{ent}$

The working basis used in this example is the same as in the last example, the matrices Q^h, LP^H... being given in Figure 6.2. The entering column is from period 1 and is defined by

$$a^{ent} = \begin{bmatrix} a \\ 0_{M\times 1} \\ 0_{M\times 1} \\ 0_{M\times 1} \end{bmatrix}, \quad \text{where} \quad a = \begin{bmatrix} 0 \\ 0 \\ -30 \\ -30 \\ 30 \\ 30 \end{bmatrix}.$$

The first step is the multiplication of the column vector a by the factorization matrix LP^1 of the entering period.

$$LP^1 a = \begin{bmatrix} 30 \\ 0 \\ -30 \\ 60 \\ -60 \\ 0 \end{bmatrix}.$$

The column vector a' is constructed with the first q^1 ($= 4$) elements of LP^1a. The last $M - q^1$ ($= 2$) elements of LP^1a appear in a'' in the same position as the last $M - q^1$ rows of LP^1 appear in X^T; rows 5 and 6 of LP^1 are the first two rows of X^T (see Figure 6.2) and therefore the last two elements of LP^1a must appear in the first two rows of a''.

$$a'' = \begin{bmatrix} -60 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Then we solve the system $zU^X = a''^T$,

$$z \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} = [-60 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0],$$

whose solution is z =[60,0,0,0,0,0]. The column vector σ^X of length M is obtained as follows

$$\sigma^X = (zLP^X)^T = \begin{bmatrix} 0 \\ 0 \\ 60 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Now, we have to solve the $H(=4)$ systems as described in 3) considering that the entering period is period 1.

For period 1,

$$U^1\sigma^1 = a' + LP^1\sigma^X,$$

$$\begin{bmatrix} 1 & 80 & 10 & 0 \\ 0 & 80 & 0 & 80 \\ 0 & 0 & -10 & 80 \\ 0 & 0 & 0 & -160 \end{bmatrix} \sigma^1 = [30 \quad 0 \quad 30 \quad 0],$$

and the solution is $\sigma^1 = \begin{bmatrix} 60 \\ 0 \\ -3 \\ 0 \end{bmatrix}.$

For period 2,

$$U^2\sigma^2 = LP^2\sigma^X,$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 40 \\ 0 & 1 & 0 & 90 & 0 \\ 0 & 0 & 1 & 90 & 40 \\ 0 & 0 & 0 & 90 & 0 \\ 0 & 0 & 0 & 0 & -40 \end{bmatrix} \sigma^2 = [0 \quad 60 \quad 0 \quad 0 \quad 0],$$

$$\text{and the solution is} \qquad \sigma^2 = \begin{bmatrix} 0 \\ 60 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

For period 3,

$$U^3\sigma^3 = LP^3\sigma^X,$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 50 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 50 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -50 \end{bmatrix} \sigma^3 = [0 \quad 60 \quad 0 \quad 0 \quad 0 \quad 0],$$

$$\text{and the solution is} \qquad \sigma^3 = \begin{bmatrix} 0 \\ 60 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

For period 4,

$$U^4\sigma^4 = LP^4\sigma^X,$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 60 \end{bmatrix} \sigma^4 = [60 \quad 0 \quad 0],$$

$$\text{and the solution is} \qquad \sigma^4 = \begin{bmatrix} 60 \\ 0 \\ 0 \end{bmatrix}.$$

The resulting column vector σ_2 is given by

$$\sigma_2 = \begin{bmatrix} \sigma^1 \\ \vdots \\ \sigma^H \\ \sigma^X \end{bmatrix}.$$

Chapter 7 Column Generation

The arc-chain representation of the network leads to a formulation having relatively few constraints compared to the formulation based on the node-arc representation described in Section 2.4. As we have seen, our formulation reveals a structure which can be exploited in two ways. First, the GUB technique is used to take advantage of the multi-commodity nature of the problem. Then, the multi-period character of the problem is further exploited by a decomposed triangular factorization. Therefore, the formulation based on the arc-chain matrix presents many advantages.

The only characteristic of the arc-chain formulation that could be a major problem is the number of columns. Since each column is associated with a route, and the number of routes grows proportionally to $N!$, the arc-chain matrix rapidly becomes excessively large. It is almost impossible to determine the number of routes in a general network simply from the number of nooes and arcs, N and M respectively. In fact, even when the architecture of the network is known. the problem of determining the number of routes is difficult to solve. The addition of an arc can only increase the number of routes in a network. Therefore, by considering a complete graph, one can compute an upper bound on the number of routes for all networks having N nodes. This upper bound is given by

$$\frac{N!}{2} \sum_{n=0}^{N-2} \frac{1}{n!} \simeq 1.35N! \qquad \text{for } N \geq 5 \ . \qquad (7.1)$$

Figure 7.1 shows how rapidly the number of routes increases in a complete network. In a network having 10 nodes the upper bound on the number of routes is close to 5 million. In such conditions, it is inconceivable to consider solving any problem with all columns of the constraint matrix represented explicitly. Fortunately, methods based on the RSM do not necessarily require all columns; only the columns in the basis and the entering column affect the current iteration are required.

The entering column is determined in the second step of the RSM. In Section 3.2.2.2, we defined the entering column $\widehat{a}^{ent}$ as being the non-basic column with the smallest reduced cost. Thus

$$-(\pi,\mu)\widehat{a}^{ent} = \min_{\widehat{a}^j \notin B} \left\{ -(\pi,\mu)\widehat{a}^j \right\}. \tag{7.2}$$

If $-(\pi,\mu)\widehat{a}^{ent} \geq 0$ then the current solution is optimal. Until now we have always considered the choice of the column having the smallest negative reduced cost as the entering column, but any column associated with a negative reduced cost is eligible to become $\widehat{a}^{ent}$. It is perfectly acceptable to choose the first column such that $-(\pi,\mu)\widehat{a}^j < 0$ as entering column. The best compromise between the most promising candidate and the first acceptable candidate is to find the best possible candidate that can be obtained in a "reasonable" amount of time. This excludes any enumeration method because of the number of columns.

In some cases, the underlying structure of the problem enables us to use methods which generate a good candidate by solving sub-problems such as shortest path or knapsack problems. The main advantage of these methods, called column generation, is their efficiency in finding an eligible column with a relatively good reduced cost in a fast and intelligent fashion.

Number of Routes in a Complete Graph			
N	$\frac{N!}{2}\sum_{n=0}^{N-2}\frac{1}{n!}$	N	$\frac{N!}{2}\sum_{n=0}^{N-2}\frac{1}{n!}$
1	0	8	6×10^4
2	1	9	5×10^5
3	6	10	5×10^6
4	30	15	2×10^{12}
5	160	20	3×10^{18}
6	975	25	2×10^{25}
7	6846	50	4×10^{64}

Fig. 7.1 Relation between N and the number of routes

The ability to recognize optimality only when it exists is the most important characteristic of a column generation method. To find an eligible column having a value $-(\pi,\mu)\hat{a}^j$ as small as possible can be considered as the prime objective of the column generation method, but finding a column if there is one eligible is the *sine qua non* condition. In fact, it should not indicate optimality (which occurs when all columns have non-negative reduce costs) if there is at least one route with a negative $-(\pi,\mu)\hat{a}^j$. The next section presents a column generation method for the minimum cost sizing problem based on the arc-chain representation of rearrangeable networks. Then, in the following section we demonstrate that the method will not recognize optimality when it does not exist.

7.1 A Column Generation Method

The first step in the design of a column generation method is to find an interpretation of the reduce costs, $-(\pi,\mu)\hat{a}^j$, which are used to determine the entering column. Network optimization problems usually lead to a column generation method based on the structure of the underlying network. When looking for an interpretation of the columns of our formulation, there are two cases to consider: a column is associated with a route or with a slack variable. Note that the columns corresponding to capacity variables are not considered since they do not leave the basis (see Chapter 4), and therefore are not eligibible to enter it.

In order to simplify the presentation of the column generation method we use a notation where the row vectors π and μ are divided by period as follows:

$$\begin{aligned} \pi &= (\pi^1, \ldots, \pi^h, \ldots, \pi^H), \\ \mu &= (\mu^1, \ldots, \mu^h, \ldots, \mu^H), \end{aligned} \tag{7.3}$$

where each π^h and μ^h are row vectors of length M and K, respectively. The variable π^h_m is associated with the capacity constraint representing arc m in period h. Before we introduce slack variables in the formulation, the capacity constraints were of the form

$$A^h f^h \leq C \qquad \text{for } h = 1 \ldots H. \tag{7.4}$$

The relation between primal and dual problems insures that all π^h_m are ≤ 0 at optimality. In primal simplex methods, such as the RSM and GUB, feasibility of the dual problem is not preserved during the execution of the algorithm. In fact, optimality is attained at the same time as feasibility of the dual problem. This implies that some of the π^h_m will likely be positive before optimality is attained.

The product $-(\pi, \mu)\hat{a}^j$ can be divided into two parts: the first is the multiplication of π by the first HM elements of $\hat{a}^j$ and the second is the multiplication of μ by the last HK elements of $\hat{a}^j$. If $\hat{a}^j$ represents a slack variable, then there is only one non-zero value, a 1, among the first HM elements and all the last HK elements are zeroes. Thus, the reduced cost associated with a slack variable is given by the element of $-\pi$ corresponding to the non-zero element of $\hat{a}^j$. An example is presented in Figure 7.2. A column $\hat{a}^j$ associated with a route has at most M non-zero entries among the first HM elements, corresponding to the arcs that compose route $\hat{a}^j$. In the last HK elements there is just one non-zero value corresponding to the OD-pair and period of route $\hat{a}^j$ (see Figure 2.7c). Let the column vector of length M associated with a route $\hat{a}^j$ in the arc-chain matrix A^h be denoted by a^{h^j}.

For a column $\hat{a}^j$ from period $h = \mathrm{PER}(\hat{a}^j)$ and origin-destination $k = \mathrm{OD}(\hat{a}^j)$, the product $-(\pi,\mu)\hat{a}^j$ is given by $-\pi^h a^{h^j} - \mu_k^h$ as shown in Figure 7.2. In the network having the $-\pi_m^h$ as costs on the arcs, the total cost of a route $\hat{a}^j$ multiplied by the demand d_k^h for OD-pair h and k is equal to $-\pi^h a^{h^j}$. Thus, the reduced cost associated with a column $\hat{a}^j$ can be obtained by a pre-multiplication of the total cost of route $\hat{a}^j$ by the demand d_k^h, and then subtracting μ_k^h.

The column generation method presented in Figure 7.3 can be summarized as follows:

1) Find the cost SP_k^h of a shortest path between each OD-pair k at every period h. For a period h, the cost on arc m of the network is determined as follows:

$$\text{cost on arc } m = \begin{cases} -\pi_m^h & \text{if } \pi_m^h \leq 0, \\ \infty & \text{if } \pi_m^h > 0\,. \end{cases} \tag{7.5}$$

2) The best reduced cost of the routes associated with shortest paths is obtained by the following equation,

$$l^* = \min_{\substack{h=1..H \\ k=1..K}} \left\{ \mathrm{SP}_k^h d_k^h - \mu_k^h \right\}. \tag{7.6}$$

3) The entering column $\hat{a}^{ent}$ is defined as the column for which

$$-(\pi,\mu)\hat{a}^{ent} = \min_{\substack{h=1..H \\ m=1..M}} \{l^*, -\pi_m^h\}. \tag{7.7}$$

If $-(\pi,\mu)\hat{a}^{ent} \geq 0$, then the current solution is optimal.

In part 1), we start by defining H networks, one for each period, on which we apply a shortest path procedure in order to find the best route between each OD-pair. The networks are composed only of the arcs having positive costs $-\pi_m^h$, the arcs associated with a negative cost being removed to prevent possible negative cycles. Since there are no negative cycles, the algorithm due to Floyd [50] can be

$$-\left[\pi^1 \ \ldots, \overbrace{(\pi_1^h, \pi_2^h \ldots \pi_M^h)}^{\pi^h}, \ldots, \pi^H, \mu^1 \ \ldots, \mu^h \ \ldots, \mu^H\right] \begin{bmatrix} 0_{M\times 1} \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0_{M\times 1} \\ \hdashline 0_{HK\times 1} \end{bmatrix}$$

$$\uparrow$$

a^J : slack variable associated with arc m in period h

$$= -\pi_m^h$$

Fig. 7.2a) Reduced cost of a slack variable

$$-\left[\pi^1 \ \ldots, \pi^h \ \ldots, \pi^H; \mu^1 \ \ldots, \overbrace{(\mu_1^h, \mu_2^h \ldots \mu_M^h)}^{\mu^h}, \ldots, \mu^H\right] \begin{bmatrix} 0_{M\times 1} \\ \vdots \\ a^{h^J} \\ \vdots \\ 0_{M\times 1} \\ \hdashline 0_{K\times 1} \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0_{K\times 1} \end{bmatrix}$$

$$\uparrow$$

a^J : route from period h connecting OD-pa'r k

$$= -\pi^h a^{h^J} - \mu_k^h$$

Fig. 7.2b) Reduced cost of a route

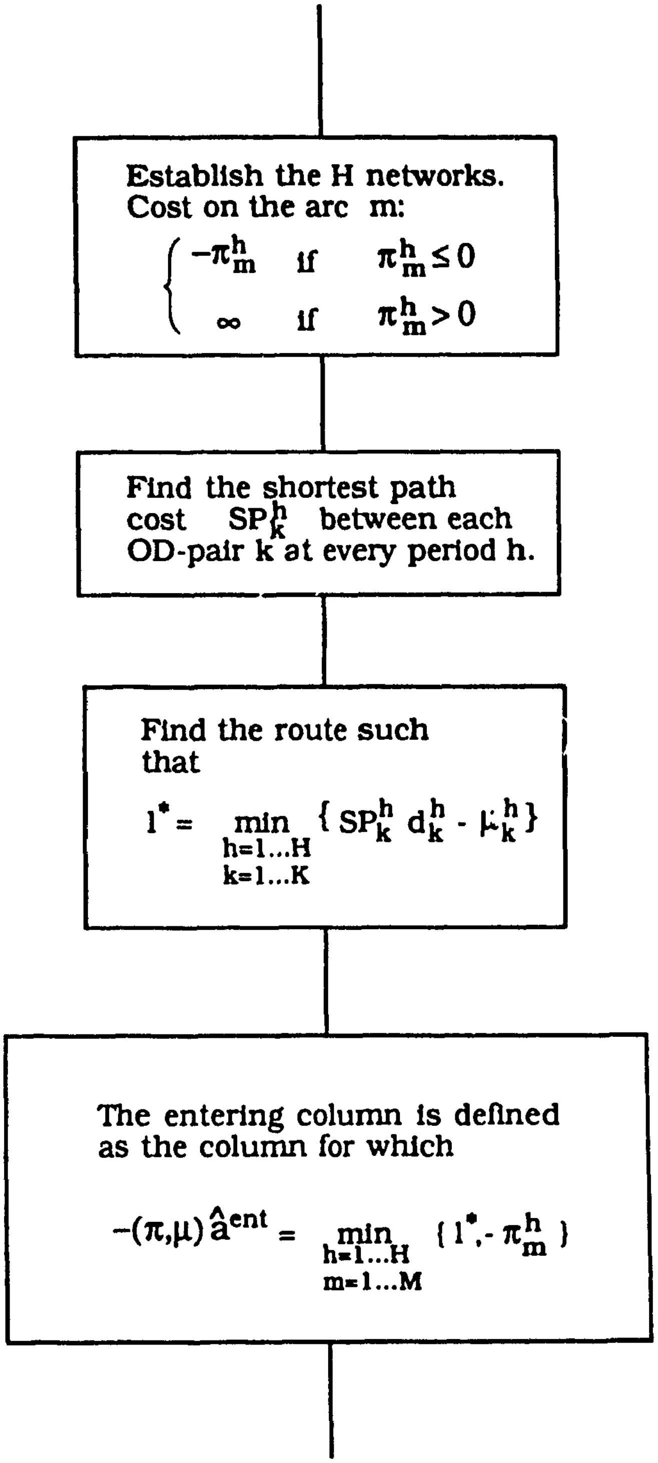

Establish the H networks.
Cost on the arc m:
$\begin{cases} -\pi_m^h & \text{if} \quad \pi_m^h \le 0 \\ \infty & \text{if} \quad \pi_m^h > 0 \end{cases}$
Find the shortest path cost SP_k^h between each OD-pair k at every period h.
Find the route such that
$l^* = \min_{\substack{h=1\ldots H \\ k=1\ldots K}} \{ SP_k^h \, d_k^h - \mu_k^h \}$
The entering column is defined as the column for which
$-(\pi,\mu)\hat{a}^{ent} = \min_{\substack{h=1\ldots H \\ m=1\ldots M}} \{ l^*, - \pi_m^h \}$

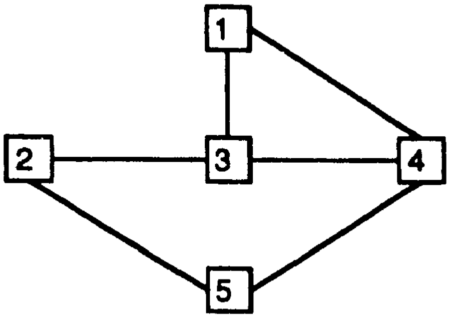
Fig. 7.3 Column generation procedure

used to find the shortest path between all OD-pairs of a given period in a time proportionnal to $O(N^3)$. Therefore, part 1) has a complexity of $O(HN^3)$ for the solution of the shortest paths for the H periods. The shortest paths obtained would be different if the arcs associated with negative costs had been considered. Since we do not have to select at each iteration the route having the smallest reduced cost, it is more efficient to use a simpler algorithm and find a good one.

The identification of the best candidate in the networks considered is done in part 2). The number of multiplications and additions is of order $O(HN^2)$ since $K = \frac{N(N-1)}{2}$. Part 3) makes the choice of the entering column, $\widehat{a}^{ent}$, by comparing the best routes obtained in step 2) with the slack variable having the smallest reduced cost. This is done in $O(HM)$ operations. The overall complexity of column generation is thus $O(HN^3)$.

Example

We present in Fig:.ɛ 7.4 a 5 node network with the corresponding dual variables π_m^h, μ_k^h and the demand between each OD-pair at every one of 4 periods. The results of the column generation method for period 1 are given in Figure 7.5. We notice that the arcs associated with negative costs, $-\pi_m^h$, are removed. In period 1, the algorithm finds three columns eligible to enter the basis: the shortest route connecting OD-pair $3-5$, and the two columns corresponding to the slack variables of arcs $1-3$ and $1-4$. We notice that the removal of arcs may disconect the network but that it does not affect the result. In period 2, there is no route eligible to enter the basis but the slack variables of arcs $2-5$ and $4-5$ are candidates with reduced costs of -1 (see Figure 7.6). Period 3 is interesting because all dual variables are



Dual variables

arc	13	14	23	25	34	45
π^1 =	0.5	0.5	-1	-2	-1	0
π^2 =	0	0	0	1	0	1
π^3 =	0	0	0	0	0	0
π^4 =	-1.5	-1.5	0	0	0	-2

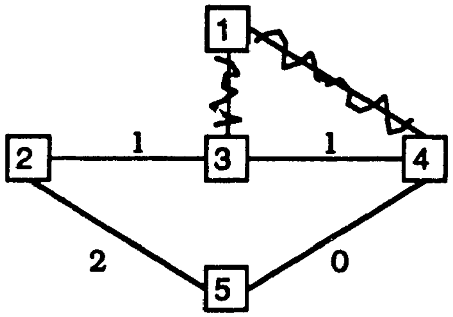
OD-pair	12	13	14	15	23	24	25	34	35	45
μ^1 =	5	-30	-5	-10	80	20	-40	80	90	0
μ^2 =	0	0	0	-10	0	0	40	0	-10	-90
μ^3 =	0	0	0	0	0	0	0	0	0	0
μ^4 =	0	90	15	70	0	0	0	0	0	120

Demands

OD-pair	12	13	14	15	23	24	25	34	35	45
d^1 =	10	60	10	20	80	10	20	80	30	10
d^2 =	30	20	40	10	10	30	40	10	10	90
d^3 =	30	20	50	10	10	30	30	10	10	60
d^4 =	40	60	10	20	20	10	10	50	30	60

Fig. 7.4 Example of π and μ values

PERIOD 1



routes

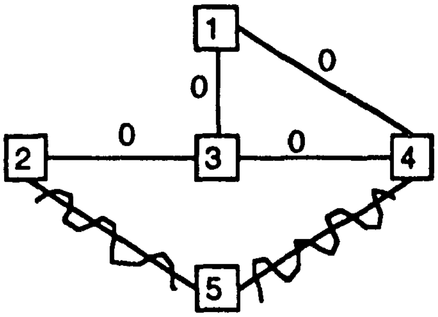
OD-pair k	shortest path	SP_k^1	$-\mu_k^1 + SP_k^1\, d_k^1$	
12	--	--	--	
13	--	--	--	
14	--	--	--	
15	--	--	--	
23	2-3	1	0	
24	2-3-4	2	0	
25	2-5	2	0	
34	3-4	1	0	
35	3-4-5	1	-60	← candidate
45	4-5	0	0	

slack variables

arc m	$-\pi_m^1$	
13	-0.5	← candidate
14	-0.5	← candidate
23	1	
25	2	
34	1	
45	0	

Fig. 7.5 Column generation for period 1

PERIOD 2



routes

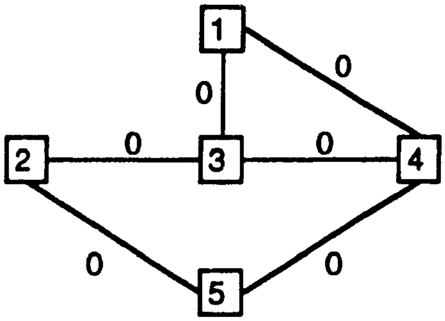
OD-pair k	shortest path	SP_k^2	$-\mu_k^2 + SP_k^2 d_k^2$
12	1-3-2	0	0
13	1-3	0	0
14	1-4	0	0
15	--	--	--
23	2-3	0	0
24	2-3-4	0	0
25	--	--	--
34	3-4	0	0
35	--	--	--
45	--	--	--

slack variables

arc m	$-\pi_m^2$	
13	0	
14	0	
23	0	
25	-1	← candidate
34	0	
45	-1	← candidate

Fig. 7.6 Column generation for period 2

PERIOD 3



routes

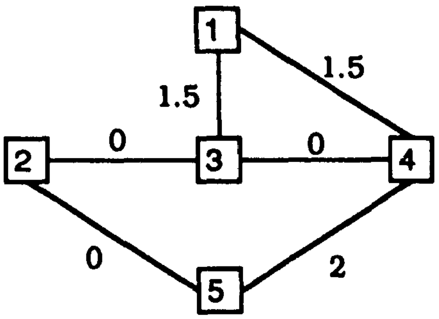
	shortest path	SP_k^3	$-\mu_k^3 + SP_k^3 d_k^3$
OD-pair k 12	--	0	0
13	--	0	0
14	--	0	0
15	--	0	0
23	--	0	0
24	--	0	0
25	--	0	0
34	--	0	0
35	--	0	0
45	--	0	0

slack variables

	$-\pi_m^3$
arc m 13	0
14	0
23	0
25	0
34	0
45	0

Fig. 7.7 Column generation for period 3

PERIOD 4



routes

OD-pair k	shortest path	SP_k^4	$-\mu_k^4 + SP_k^4 d_k^4$	
12	1-3-2	1.5	60	
13	1-3	1.5	0	
14	1-4	1.5	0	
15	1-3-2-5	1.5	-40	← candidate
23	2-3	0	0	
24	2-3-4	0	0	
25	2-5	0	0	
34	3-4	0	0	
35	3-2-5	0	0	
45	4-3-2-5	0	-120	← candidate

slack variables

arc m	$-\pi_m^4$
13	1.5
14	1.5
23	0
25	0
34	0
45	2

Fig. 7.8 column generation for period 4

zero. Consequently, all reduced costs are also zero and it is not necessary to apply the column generation method for this period. Figure 7.8 presents the results for period 4 where all the $-\pi_m^4$ are non-negative. Thus, there is no slack variable from period 4 that is eligible. It can be seen that

$$l^* = -120 < \min_{\substack{h=1 \quad 4 \\ m=1 \quad 4}} \{-\pi_m^h\} \ .$$

The column for which l^* is attained is the one that corresponds to the route 4-3-2-5 in period 4. Hence this column is chosen as the entering column.

7.2 Proof of Optimality

The validation of the column generation method is proved by showing that if there exists at least one candidate to enter the basis then the method can generate such an entering column. When there is no column eligible to enter the basis the method must be able to detect that the current solution is optimal. The proof is divided into two parts: the first is the case when there are dual variables $\pi_m^h > 0$. We show that there is always a column eligible to enter the basis and that optimality is impossible in this situation. The second case occurs when all dual variables π are non-positive. Here, we demonstrate that the algorithm finds an entering column and that if it is impossible to find one, then we are at optimality.

7.2.1 There is at Least One Positive π_m^h.

In this case optimality may not be reached because the presence of a postive π_m^h

contradicts dual feasibility and hence the optimality criterion (without degeneracy it would not be reached). The column associated with the slack variable of period h and arc m has a reduced cost equal to $-\pi_m^h$ which has a negative value. In fact, every column representing a slack variable whose defining constraint has a positive dual variable is eligible to enter the basis and may improve the objective function value. Notice that these columns are not currently in the basis because their reduced costs are necessarily non-zero.

The overall objective is the minimization of the capacity which does not favour the presence of slack variables in the basis. In most of the problems we have solved, the unused capacity at optimality is considerably small. We want to promote the choice of routes instead of slack variables as entering columns, since most of the slack variables must leave the basis in order to attain optimality. Consequently, even if we already have a slack column eligible to enter the basis, we establish the H networks, solve the shortest path problems, and find the route having the best reduced cost for the given arc costs. Then, we choose between the routes and the slack variables by comparing the reduced costs. This allows the introduction of a slack variable only if it can lead to a significant reduction of the objective function.

7.2.2 All Dual Variables π_m^h are Non-Positive.

In this case the slack variables are not eligible to enter the basis since their reduced costs, $-\pi_m^h$, are not negative. Only a column associated with a route can have negative reduced cost. If we can show that there is no such route, then we are at optimality. In the previous section, we removed the arcs having negative costs in order to avoid the presence of negative cycles. In this case. there is no arc with a negative cost, and consequently, all the arcs appear in the network of every period. Since no arc is removed, the routes found by the shortest path algorithm are in fact the best possible routes between each OD-pair. No other routes in the original formulation lead to a smaller reduced cost for a given OD-pair.

The costs of the shortest paths are multiplied by the demand d_k^h and the dual variable μ_k^h is subtracted to obtain the reduced costs. Then, the best route is defined as the route having the smallest reduced cost. If it is negative, then the corresponding route is the entering column, since there is no slack variable having a negative reduced cost to compete for the role of $\bar{a}^{ent}$. When the smallest reduced cost is not negative, then there is no route eligible to enter the basis. Thus all the routes, the slack variables, and the capacity variables have non-negative reduce costs which indicate optimality.

Chapter 8 Summary and Problems for Further Study

The recent trend towards Integrated Service Digital Networks (ISDN) has forced planners to considerably revise their approach. The principles of operation and planning cannot be easily introduced into the existing infrastructure of the telephone network. The diversity of services made available by ISDN introduces an equivalent diversity in the demand patterns.

Considerations of survivability as well as non-coincidence in the demands over several periods and the flexibility required for the integration of new concepts have led to the development of the Dynamic Network Architecture (DNA) which allows the virtual architecture of the network to evolve with the demand.

8.1 Summary

The first objective in this project has been the formulation in terms of a mathematical program of one of these concepts. The dimensioning of non-directed *rearrangeable networks* with *multi-period demands* can be formulated as a linear program. The formulation obtained, which is based on the *arc-chain* representation of the network, presents a structure that permits exploiting fully the multi-commodity and the multi-period aspects of the problem. In addition to the structure, the formulation contains a relatively small number of constraints as compared to a node-arc formulation.

The efficiency of a linear program is directly affected by the number of constraints since it determines the size of its most important component, the basis. The formulation derived in this thesis contains $HM + HK$ constraints which are divided into two groups. The first HM constraints assure that the capacity is respected on all arcs for each period. The last HK constraints contain at most one non-zero element in every column. These constraints assure that the demands are fully satisfied.

The formulation using the *node-arc* representation presents an interesting structure, but the number of constraints is proportional to HM^3. For example, in a problem having 20 periods, 10 nodes and 25 arcs, the number of constraints would be roughly 27500 whereas the arc-chain formulation chosen has only 1400 constraints for the same network. Moreover, when the structure of this formulation is fully exploited, as we have shown, the problem reduces to the equivalent of 21 problems, each of size 25.

The reduction of the problem into H subproblems is made possible by the following three characteristics of the formulation:

- Theorem 4.1 proves that the capacity variables can be kept in the basis. This structure of the basis remains unchanged during the execution of the algorithm.
- The decomposed factorization developed in this thesis replaces the inverse when solving Steps 1 and 3 of the RSM (Figure 3.3). Furthermore, it takes advantage of the structure of the working basis whereas inversion would destroy it.
- A column generation algorithm can be designed in order to avoid the explicit representation of the columns.

The application of the GUB technique to our formulation leads to the reduction of a basis of size $HM + HK$ to a working basis of size HM. Then, the use of the decomposed factorization presented in Chapter 5 as a computational device reduces considerably the number of operations necessary to carry out the steps of the RSM. The advantage of this decomposed factorization is shown by comparing its computational complexity with that of a standard factorization procedure applied to the working basis. In the following paragraphs we shall compare these two methods of factorization with respect to:

- Initial factorization.
- Effect on the steps of the RSM.
- Update of the basis.
- Refactorization.

The factorization of the initial basis requires only a permutation matrix per period. The computational complexity of the initial factorization is $O(HM)$ with both methods.

The factorization of the working basis without decomposing does not exploit its structure. The computational complexity of finding the dual variables in Step 1 of the RSM is $O(H^2M^2)$. The solution of the other linear system, where the representation of the entering column is computed, also requires $O(H^2M^2)$ operations. The update of the factorization for a matrix of order HM has complexity $O(H^2M^2)$. Finally, the number of operations required for the periodic refactorization of the working basis is of order $O(H^3M^3)$.

By contrast, the decomposed factorization presented in this thesis reduces the computational complexity of an iteration of the RSM. In Step 1, there is only one linear system of size M to solve. The row vector π is obtained by the multiplication of H matrices by the result of the linear system for a complexity of $O(HM^2)$. The product of π by the key columns gives the remaining HK dual variables, μ, requiring $O(HM^2)$ operations. The overall complexity of Step 1 is $O(HM^2)$. The solution of the linear system of Step 3 also requires $O(HM^2)$ operations due to solution of H linear systems of size at most M.

The update of the factorization is the crucial step, since it must consider the positions of the entering and the leaving columns, preserve the structure of the working basis, and avoid affecting the matrices associated with periods not involved in the update process. The procedure presented in Section 5.2 updates only the matrices of periods PER(ent), PER($leav$), and the matrix associated with the capacity variables in the last M columns. The update of the three upper-triangular matrices requires only $O(M^2)$ operations. The refactorization of the working basis can be combined with the update process. We have shown that on the average refactorization requires $O(M^3)$ operations per iteration.

The column generation method presented in Chapter 7 is necessary if we consider that the number of columns is proportional to $N!$. This eliminates any enumeration method that would not search for an entering column in an intelligent fashion. Instead, we have a method that decomposes the problem by periods and uses the underlying network to directly generate a good candidate for each OD-pair based on a shortest path procedure. The overall complexity of the column generation procedure is $O(HM^{1.5})$, which is equivalent, in terms of computational complexity, to solving H problems, one for each period.

8.2 Problems for Further Study

In the context of this project, there was no cost associated with the use of a route. It is assumed that all the costs are incurred when the capacity is installed. A possible extension to this work would be to consider a cost for each route variable. In some situations, it may also be interesting to associate a cost with the slack variables. Considering costs on the flow and slack variables would probably affect only the first two steps of the RSM. The first is affected since it solves a linear system where the right-hand side is given by the costs of the basic variables. The second step, which determines the entering column, would be the most affected as the route with the smallest reduced cost could not be identified simply using a shortest path algorithm.

Another problem to consider for further study is the capacity expansion problem in a rearrangeable undirected network. In this thesis, we have addressed the sizing problem, given the demands for H periods without having any restriction on the size of the arcs. Capacity expansion addresses a problem having a certain capacity already installed in the network. The same problem may also consider upper-bounds on the capacity of pre-defined sets of arcs. The presence of new columns in the formulation would probably change the structure of the working basis. The effect of changing the structure is difficult to forecast, but we believe that it is possible to adapt our factorization procedure to this situation.

Depending on the technology used in the network, the modularity of the equipment leads to highly non-linear costs which cannot be approximated with the linear cost function used in this thesis. The implications for the solution method can be divided into two parts. First, the cost function becomes piecewise linear. Since the problem must be separated into intervals having a minimum and maximum capacity, it is necessa.y to solve the capacity expansion problem mentioned above. The second aspect of the problem consists of developing a method based on the capacity expansion problem that can find the interval for each arc in which the optimal solution lies. For example, an algorithm of the *branch and bound* type would probably be the basis for a solution of this problem.

In Chapter 7, we presented a column generation method that finds the best route containing no arcs having a negative cost, then it is compared with the best slack variable eligible to enter the basis. If it were possible in a computational time comparable to the Floyd-Warshall algorithm to find shortest paths in a network having negative cycles, then it would be possible to find routes with smaller reduced costs. Since we want to promote the choice of routes having smaller reduced costs as entering columns, the efficiency of the algorithm would be further increased by this possibility.

REFERENCES

[1] Likoray T. , Poirier J.P. , O"Farrell D. and Huberman R., "Strategic Planning Tools Evolution ," technical letter: TL 86-0210 , November 1986.

[2] Mason L., "Network planning notes," INRS report 84-02, 1984.

[3] Hayes J.F., *Modeling and Analysis of Computer Communication Networks*. New-York: Plenum, 1984.

[4] Luetchford J.C., "CCITT Recommendations - Network Aspects of the ISDN," *IEEE Journal on Selected Areas in Communications* , vol. SAC-4, no.3, pp. 334-342, May 1986.

[5] Schwartz M., *Telecommunication Networks: Protocols, Modeling and Analysis* . Addison-Wesley, 1987.

[6] CCITT Red book III.3, "I-series recommendations," October, 1985.

[7] CCITT Red book III.5, "I-series recommendations," October, 1985.

[8] CCITT Study group XVIII, "Meeting report R1-R4," presented at the Geneva, Switzerland, June 1985.

[9] Ashkar G. , Ford G. and Pecsvaradi T., "Reshaping the network for special services," *Telephony*, vol. , pp. 72-88, January 1984.

[10] Assad A., "Multicommodity network flows - A survey," *Networks*, vol. 8, pp. 37-91, 1978.

[11] Ford L. and Fulkerson D., *Flows in networks* . Princeton university press, 1963.

[12] Hartman J. and Lasdon L., "A generalized upper bounding algorithm for multicommodity network flow problems," *International Conference on Communications*, vol. , pp. 6.2.1-6.2.7, 1986.

[13] Bazaraa M. and Jarvis J., *Linear Programming and network flows*. John Wiley and Sons, 1977.

[14] Jewell W., "A primal-dual multicommodity flow algorithm," Oper. res. center 66-24, 1966.

[15] Ferland A. , Girard A. and Lafond L., "Multicommodity flow problem with variable arcs capacities," Universite de Montreal no.200, 1977.

[16] McCallum C., "A generalized Upper Bounding approach to a communication network planning problem," *Networks* , vol. 7, pp. 1-23, 1977.

[17] Chen H and DeWald C., "A generalized chain labelling algorithm for solving multicommodity flow problems," *Comp. and oper. res.*, vol. 1, pp. 437-465, 1974.

[18] Wollmer R., "Multicommodity network flows with resource constraints: The generalized multicommodity flow problem," *Networks* , vol. 1, pp. 245-263, 1972.

[19] Aronson J. and Chen B., "A forward network simplex algorithm for solving multiperiod network flow problems," *Naval Research Logistics Quaterly*, vol. 33, pp. 345-67, August 1986.

[20] Aronson J., "The multiperiod assignment problem: A multicommodity network flow model and specialized branch and bound algorithm," *Eur. J. Operation research*, vol. 23, pp. 367-81, March 1986.

[21] Hu T., "Multicommodity network flows," *Operation research*, vol. 11, pp. 344-360, 1962.

[22] Creamens J. , Smith A. and Tyndall G., "Optimal multicommodity network flows with resource allocation," *Naval Research Logistics Quaterly*, vol. 17, pp. 269-280, March 1970.

[23] Ford L. and Fulkerson D., "A suggested computation for maximal multicommodity network flows," *Management science*, vol. 5, pp. 97-101, 1958.

[24] Dantzig G. and Wolfe P., "The decomposition algorithm for linear programming," *Econometrica*, vol. 29, pp. 767-778, 1961.

[25] Dantzig G. and Van Slyke M., "Generalized Upper Bounding techniques," *J. comp. System Science* , vol. 1, pp. 213-226, 1967.

[26] Rapp Y., "Planning of junction network with non-coincident busy hours," *Ericsson Technics* , vol. 27, pp. 3-23, May 1971.

[27] Eisenberg M., "Engineering traffic networks for more than one busy hour," *The Bell System Technical Journal*, vol. 56, no.1, pp. 1-21, January 1977.

[28] Elsner W., "A descent algorithm for the multihour sizing of traffic networks," *The Bell System Technical Journal*, vol. 56, no.8, pp. 1405-1430, October 1977.

[29] Girard A. , Lansard P. and Liau B., "A Multihour ECCS theory and applications ," *International Conference on Communications*, vol. , pp. 6.2.1-6.2.7, 1986.

[30] Horn R., "A simple approach to dimensionning a tellecommunication network for many hours of traffic demand," *International Conference on Communications*, vol. . pp. 67.2.1-67.2.5, 1981.

[31] Huberman R. , Hurtibise S. , Lenir S. and Drwiega T, "Multihour dimensioning for a dynamically routed network," presented at the ITC-11, , 1985.

[32] Zadeh N., "On building minimum cost networks over time," *Networks* , vol. 4, pp. 19-34, 1974.

[33] Yaged B., "Minimal cost routing for dynamic network models," *Networks* , vol. 3, pp. 193-224, 1973.

[34] Yaged B., "Minimum cost routing for static network models," *Networks* , vol. 1, pp. 139-172, 1971.

[35] Smith R.L., "Deferral strategies for a dynamic communication network," *Networks* , vol. 9, pp. 61-87, 1979.

[36] Akiyama M., "Variable communication network design ," presented at the ITC 9, Torremerolinos, Spain, 1979.

[37] Winnicki A. and Paczynski J., "Approach to design of three-layer controlled telephone networks." *Large Scale Systems* , vol. 1, pp. 245-256, 1980.

[38] Larocque M., "Choix d'un algortihme de rearrangement pour un reseau existant avec commutateurs lents," Rapport technique INRS 85-06 , aout 1985.

[39] Girard A., "Modeles mathematiques du dimensionnement multi-matrice," Rapport technique INRS 85-13, 1985.

[40] Kortanek K. , Lee D. and Polak G., "A linear programming model for design of communication networks with time-varying demands," *Naval Research Logistics Quaterly*, vol. 28, pp. 1-32. March 1981.

[41] Kortanek K. and Polak G., "Network design and dynamic routing under queueing demand," Unpublished paper, april 1984.

[42] Chvátal V., *Linear Programming*. Freeman, 1983.

[43] Dantzig G. , Orden A. and Wolfe P., "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, vol. 5, pp. 183-195, 1955.

[44] Murty K., *Linear and Combinatorial Programming*. Krieger, 1976.

[45] Kallio M. and Porteus E., "Triangular Factorization and Generalized Upper Bounding." *Operation research*, vol. 25, pp. 89-99, 1977.

[46] Dantzig G. and Orchard-Hays W.. "The product form for the inverse in the simplex method," *Mathematical Tables and Other Aids to Computation*, vol. 8, pp. 64-67. 1954.

[47] Gill P.E. , Murray W. and Wright M., *Practical Optimization*. Stanford: Academic Press, 1981.

[48] Saunders M., "The complexity of LU updating in the simplex method," in *The Complexity of Computational Problem Solving*. Andersen R. and Brent R. Queenlands University Press, 1976, pp. 214-230.

[49] Saunders M., "A fast, stable implementation of the simplex method using Bartels-Golub updating," in *Sparse Matrix Computations*, Bunch J. and Rose D. New-York: Academic Press, 1976, pp. 213-226.

[50] Floyd R.W., "Algorithm 97: Shortest Path," *Communication of ACM*, vol. 5, pp. 345, 1962.