Network partitioning techniques based on network natural properties for power system application

by

Ali Mani Turki Alkhelaiwi

A Thesis submitted for the Degree of Doctor of Philosophy in Electrical Engineering, at Brunel University

Department of Electronic and
Computer Engineering,
Brunel University,
Uxbridge, Middlesex.

January 2002

Abstract

In this thesis, the problem of partitioning a network into inter-connected sub-networks is addressed. The goal is to achieve a partitioning which satisfies a set of specific engineering constraints, imposed in this case, by the requirements of the decomposed state-estimation (DSE) in electrical power systems. The network-partitioning problem is classified as NP-hard problem. Although many heuristic algorithms have been proposed for its solution, these often lack directness and computational simplicity.

In this thesis, three new partitioning techniques are described which (i) satisfy the DSE constraints, and (ii) simplify the NP-hard problem by using the natural graph properties of a network.

The first technique is based on partitioning a spanning tree optimally using the natural property of the spanning tree branches. As with existing heuristic techniques, information on the partitioning is obtained only at the end of the partitioning process. The study of the DSE constraints leads to define conditions of an ideal balanced partitioning. This enables data on the balanced partitioning to be obtained, including the numbers of boundary nodes and cut-edges. The second partitioning technique is designed to obtain these data for a given network, by finding the minimum covering set of nodes with maximum nodal degree. Further simplification is then possible if additional graph-theoretical properties are used. A new natural property entitled the 'edge state phenomenon' is defined. The edge state phenomenon may be exploited to generate new network properties. In the third partitioning technique, two of these, the 'network external closed path' and the 'open internal paths', are used to identify the balanced partitioning, and hence to partition the network.

Examples of the application of all three methods to network partitioning are provided.

Acknowledgements

My continuous thanks are to Allah, the creator of the universe. Then, I would like to express my gratitude to my research supervisor Dr. John F. Marsh, who aroused my interest in network-partitioning problem, for his continuous invaluable guidance, help and interest throughout the course of this work.

I am indebted to my family for their constant encouragement and support, which made this thesis at all possible. I would like to thank my mother for her continuous praying for me.

Finally, I am very grateful to my friends, for their praying and encouragement throughout this work.

Memorandum

This thesis is based on work carried out by the author in the Department of Electronic and Computer Engineering at Brunel University between February 1997 and January 2002.

All work and ideas in this thesis are original unless otherwise acknowledged in the text or by references. The work has not been submitted for another Degree in this University, nor in any other University.

The main contributions of this thesis include:

1. Development of an optimal partitioning technique suitable for partitioning an electrical power system network.
2. Definition of conditions for ideal balanced partitioning.
3. Development of a fast algorithm to partition the network whilst satisfying the DSE restrictions.
4. The discovery of the edge state phenomenon, following an exploration of the network natural properties.
5. Development of the balance partitioning theorem based on the edge state phenomenon.

Dedication

To

Anwar

and

to my sons

List of contents

List of Tables

List of Examples

List of Figures

List of Flowcharts

List of Symbols

Φ	=	The empty set
A	=	The incidence matrix
B_i	=	The i^{th} branch
BB	=	A bottom branch in a spanning tree
BJ	=	A junction branch in a spanning tree
c	=	The number of cycles in the network
c_I	=	The number of internal cycles in the network
c_X	=	The number of external cycles in the network
c_{XI}	=	The number of mixed cycles in the network
D_G	=	The total degree of the network G
D_X	=	The degree of the external nodes
D_I	=	The degree of the internal nodes
D_B	=	The degree of the bridge nodes
D_S	=	The degree of the one degree nodes
e	=	An edge
E	=	The set of network edges
E_{XCi}	=	The set of external edges of the i^{th} cycle
E_{ICi}	=	The set of internal edges of the i^{th} cycle
E_{Ci}	=	The set of edges of the i^{th} cycle
E_P	=	The set of edges of a path P
g_C	=	The sum of the circumferences of the c cycles in the network
g_i	=	The circumference of the i^{th} cycle
g_{XC}	=	The circumference of an external cycle in the network

g_{IC}	=	The circumference of an internal cycle in the network
g_{XI}	=	The circumference of a mixed cycle in the network
g_{max}	=	The circumference of a cycle in the network with maximum number of edges
g_{min}	=	The minimum circumference of a cycle is three
G	=	A graph
G_N	=	A graph of a network
L_i	=	The length of the i^{th} branch
m	=	The total number of network edges
m_X	=	Number of external edges in the network
m_I	=	Number of external nodes in the network
m_S	=	Number of one-degree edges in the network
m_B	=	Number of bridge edges in the network
m_{XCi}	=	The number of external edges of the i^{th} cycle
m_{ICi}	=	The number of internal edges of the i^{th} cycle
m_{Ci}	=	The number of edges of the i^{th} cycle
m_t	=	The number of edges of a spanning tree
m_P	=	The number of elements of E_P
n	=	Number of nodes in the network
n_X	=	Number of external nodes in the network
n_I	=	Number of internal nodes in the network
n_S	=	Number of one-degree nodes in the network
n_B	=	Number of bridge nodes in the network
n_P	=	Number of elements of V_P
n_{Xd}	=	Number of elements of the V_{Xd} set
n_{XD}	=	Number of elements of the V_{XD} set

n_{XdI}	=	Number of elements of the V_{XdI} set
n_{XdB}	=	Number of elements of the V_{XdB} set
n_{XDB}	=	Number of elements of the V_{XDB} set
n_{XdBI}	=	Number of elements of the V_{XdBI} set
n_{XDBI}	=	Number of elements of the V_{XDBI} set
n_{XCi}	=	Number of external nodes of the i^{th} cycle
n_{ICi}	=	Number of internal nodes of the i^{th} cycle
n_{Ci}	=	The number of nodes of the i^{th} cycle
P	=	A path
P_C	=	A closed path
P_o	=	An open path
P_{OX}	=	An open external path
P_{OXI}	=	An open mixed path
p_{GX}	=	The network external close path
T	=	A spanning tree
T_i	=	The i^{th} sub-spanning tree
T^*	=	A co-spanning tree
U	=	A spanning tree matrix
V	=	The network set of nodes
V_X	=	The set of external nodes
V_I	=	The set of internal nodes
V_B	=	The set of bridge nodes
V_S	=	The set of degree nodes
V_P	=	The set of nodes of a path P
V_{Xd}	=	The set of external nodes of degree d for $d = 2, 4, \cdots$
V_{XD}	=	The direct sum of the V_{Xd} sets

V_{XdI}	=	The set of external-internal nodes of degree d
V_{XDI}	=	The direct sum of the V_{XdI} sets for $d = 2, 4, \cdots$
V_{XdB}	=	The set external bridge nodes of degree d
V_{XDB}	=	The direct sum of the V_{XdB} sets for $d = 2, 4, \cdots$
V_{XdBI}	=	The set of external bridge nodes with internal edges
V_{XDBI}	=	The direct sum of the V_{XdBI} sets for $d = 2, 4, \cdots$
V_{XCi}	=	The set of external nodes of the i^{th} cycle
V_{ICi}	=	The set of internal nodes of the i^{th} cycle
V_{Ci}	=	The set of nodes of the i^{th} cycle

Glossary

Adjacent nodes	Two nodes are adjacent if they are connected by an edge.
Adjacent edges	Two edges are adjacent if they share one node.
Boundary nodes	The end-nodes of the cut-edges.
Circumference	The number of edges or nodes of a cycle.
Common node	A node between two dependent edges.
Connected	A graph is connected if there is a path connecting every pair of vertices. A graph that is not connected can be divided into **connected components** and disconnected.
Covering set of nodes	A set of nodes that has connection with every node in the network.
Cut-edge	An edge that its removal disconnects the network.
Cut-line	A line that partitions every edge into two nodes.
Cut-node	A cut node is a node that if removed (along with all edges incident with it) produces a graph with more connected components than the original network.
Cycle	A closed path with a minimum number of edges.
Dependent edges	Two edges that share one node
Independent edges	Two edges are independent if do not have any node between them
Maximum circumference	The maximum number circumference of a cycle in the network

Minimum circumference	The minimum number of edges that a cycle may have
The network external closed path	The number of external edges in the network
Partitioning	A conditional division operation
Path	A path is a sequence of consecutive edges in a network and the length of the path is the number of edges traversed.
Gobal boundary nodes	The set of boundary nodes in the complete network
Gobal internal nodes	The set of nodes in the complete network that are not boundary node
Spanning tree	A tree with n nodes and m-1 edges
Set of cut-edges	The minimum set of edges, the removal of which produces a disconnected network

Chapter 1

Introduction

1.1General

A network is a graphical representation of a system, which is a physical or abstract object exhibiting complexity. A network comprises a set of vertices or **nodes**, together with a set of **edges,** which define interconnections between pairs of nodes. A set of nodes and a set of interconnected node-pairs provide a complete and unique characterisation of the **configuration** for a related network.

In studies of system performance, it is often required to perform computations based on a model description of a network, together with a set of measurement data obtained from simulations and/or instrumentation. Such computations typically utilise a database indexed on each node, and covering all nodes in the network. The volume of computation clearly increases with network size. In an n-node network, computations of $O(n^p)$ may be required where the exponent p depends on the functional nature of a performance index. Often $p>1$, perhaps significantly so. If the network is sufficiently large, it may become unattractive to carry out the computational task in a single processor, for reasons of computing time or numerical accuracy.

One approach to overcoming the dimensionality problem in large networks is to partition the network into a number of interconnected sub-networks of reduced size. This provides the opportunity for distribution of the original computational task within a multi-processor configuration, where individual processors handle sub-network tasks of reduced-order, together with a further task of sub-network co-ordination which arises directly as a result of partitioning. In principle, the multi-processor solution offers an improvement in overall computational performance; the original single serial computation on the overall network may be replaced by reduced-dimension sub-network computations performed in parallel, plus a further serial co-ordination computation.

Network partitioning may be achieved by drawing cut-lines through the original network. Cut-lines intersect edges of the original network but do not pass through its nodes; a cut-line may approach another cut-line, but cut-lines do not intersect. The sub-networks so-formed are defined by the cut-lines, and each will contain a countable number of sub-networks. The interconnections between the sub-networks are identifiable as original network edges, which are intersected by the cut-lines, and termed **cut-edges**. Neighbouring sub-networks are therefore inter-connected by countable number of cut-edges.

As the number of partitions k increases, the average number of nodes in a sub-network will decrease, leading generally to a lighter computational load in each of the parallel sub-network processors. However, a greater number of smaller sub-networks will lead to an increase in sub-network interconnections, and hence an increased number of cut-edges in the partitioned network. This will increase the computational load associated with sub-network co-ordination. A trade-off is therefore necessary between the number of sub-networks to be created as a result of partitioning, and the resulting volume of computation necessary for sub-network co-ordination. Intuitively, it may seem that a useful objective in partitioning is to devise a set of cuts, which produce sub-networks which are **balanced** (i.e. of equal or near-equal size), whilst generating cut-edges with total number as small as possible, and certainly not significant greater than the average number of sub-system nodes. This statement is seen to be reasonable in the sense that, for a network with a fixed number k of partitions, as the number of cut-edges decreases, the sub-network interconnection load decreases, and the overall multi-processor computational load becomes increasingly dominated by the k-parallel sub-network computations. In the limit as the number of cut-edges tends to zero, the overall load tends to the parallel computations only; this is expected, since the overall network would then consist of k isolated subsystems.

The problem of partitioning a network according to such a specification may be relatively straightforward when the network is small and the number of partitions is low (e.g. n = 4 or n = 5 and k = 2), since the number of possibilities is not great, and the problem may be investigated by hand. The amount of work required to bring about a solution increases dramatically, however, with even a modest increase in network size. It then becomes essential to devise a systematic computer procedure for obtaining the desired result.

These observations on computations in interconnected networks have provided the motivation for investigating computer-assisted methods for network partitioning.

1.2 Applications and early developments of partitioning

The network-partitioning problem has long been recognized as being of far more than theoretical importance. Extensive application in many areas has been recorded, including: scientific computing [56, 61, 63]; VLSI design [53, 42, 73]; geographical information systems [57]; electrical power systems [44, 67]; operation research [59]; and task scheduling [16]. Other applications include circuit partitioning [53], and computer-aided design [74]. The partitioning of a network into smaller sub-networks is sometimes termed the "minimum-cut". The problem of determining the connectivity of a network arises frequently in issues of network design and reliability [10]. In a network subject to random edge failure, the network is most likely to be partitioned at the minimum cuts [40]. Many other problems, which are physically non-graphical, may be expressed as a graph- or a network-partitioning problem. For example, graph partitioning plays a fundamental role in parallel computing by identifying concurrency in a given problem when the computation process may be modelled by a graph. A partitioning of a graph into sub-graphs leads to a decomposition of data and/or computing tasks, and the sub-graphs can be mapped to individual processors of a multi-processor configuration. A useful survey of applications is given in [54].

The development of systematic algorithms for partitioning has been considerable; one of the earliest techniques, used in algorithmic problem solving, is the so-called **divide-and-conquer** approach [38], which entails dividing a given problem into a number of smaller sub-problems, finding solutions to each part, and combining these into a solution for the overall problem. If required, a further reduction in sub-problem size may be obtained by extending the technique to a lower level. Graph partitioning also has an important role to play in the design of many serial algorithms using a divide-and-conquer paradigm. Two important examples of this technique are in the solution of partial differential equations (PDEs) by domain decomposition [14], and in the computation of nested dissection ordering for solving systems of sparse linear equations [14].

In most cases, the objectives for solution of the network-partitioning problem are:

partition a given network or graph into a specific number of smaller sub-graphs having approximately equal numbers of nodes, such that the cut-edges between the sub-graphs are as few in number as possible.

In the context of parallel computation, the size of a subgraph determines the computational task that a processor in the parallel set has to perform, and the number of cut-edges is a measure of (i) the

commuincation volume in the algorithm, and (ii) the work involoved in co-ordinating the results of the parallel computations. The weaker the interactions between the sub-graphs, the lower will be the communication volume and co-ordination computation.
In the field of information retrieval, minimum cuts have been used to identify clusters of topically related documents in hypertext systems [7, 8, 15]. If the links in a hypertext collection are treated as edges in a graph, then a lower number of cuts correspond to groups of documents that have few links between them, and the clusters are thus likely to be weakly related. In this case, the computational load is dominated by parallel processing of the cluster problems, which enables significant savings in computational time to be achieved compared with a single-processor solution.

1.3 The partitioning problem

The network-partitioning problem as described generally in the previous sections is not amenable to a direct solution. To be practically useful, any technique for solution should be applicable to most, if not all, arrangements of nodes and edges. The minimum information available on the network is expected to be a set of indices for the network nodes, plus a set of indices for node-pairs representing the edges, i.e. the network configuration is known precisely. It is well-known, however, that the network-partitioning problem is one of Gary and Johnson's six basis NP-complete problems which lie at the heart of the theory of NP-completeness [13, 25]. This implies that the information of the network configuration is insufficient to facilitate a mathematical solution to the problem. At best, any attempt at devising a procedure which will result in an algorithmic solution, if one exists, must be based on heuristic principles. However, improvements in algorithmic design should be possible if the information base influencing the principles of design is enhanced. Such enhancements may come from graph-theoretic and other properties, which may be deduced for a given network. Information such as degree of each node, the number of cycles in the network, and properties of a spanning tree for the network are examples of possible properties for consideration.

1.4 Classification of partitioning algorithms

Regardless that the graph-partitioning problem is NP-hard, methods for solution have undergone considerable development since the divide-and-conquer approaches discussed in section 1.2. Some interesting heuristic algorithms have been proposed in the literature, [22, 41, 73, 74]. A good partitioning technique can significantly reduce the complexity of the problem and improve both the timing performance and the reliability of the system.

In general, the proposed partitioning algorithms can be classified in three ways [13, 26]. First, partitioning algorithms can also be classified based on the nature of the algorithms. There are two types under such criteria; **deterministic** and **probabilistic** algorithms. Deterministic algorithms produce repeatable or deterministic solutions. For example, an algorithm that makes use of deterministic functions will always generate the same solution for a given problem. On the other hand, the probabilistic algorithms are capable of producing a different solution for the same problem each time they are performed, as they depend on some random functions.

Second, partitioning algorithms can also be classified into constructive and iterative algorithms. The input to the constructive algorithms is the circuit netlist. The output is the set of partitions along with new netlist. Constructive algorithms are typically used to form some initial partitions using construction methods such as breadth first search, network flow, or eigenvector decomposition methods, which then can be improved by iterative algorithms. In that sense, constructive algorithms are used as preprocessing algorithms for partitioning. They are usually fast, but the partitions generated by these algorithms may be far from optimal. Iterative algorithms, on the other hand, accept a set of partitions and the netlist as input and generate an improved set of partitions along with the corresponding netlist. Iterative improvement algorithms are based on the greedy strategy: They start with some feasible solution and iteratively move to the best (improving) neighboring solution. The process terminates when the algorithm reaches a local minimum, i.e., a solution for which all neighbors have greater cost. Greedy improvement methods apply simple pair-swap or single-move neighborhood operators, and tend quickly to reach local minimal corresponding to poor solutions. Thus, many approaches rely on extended neighbourhood structures, which effectively allow hill-climbing out of local minimal.

Third, partitioning algorithms can also be classified based on the process used for partitioning. We have the following categories under such criteria; group migration, stochastic hill-climbing, clustering, and multilevel algorithms. The group migration algorithms start with some partitions, usually generated randomly, and then move cells among partitions to improve the partitioning. In practice, group migration algorithms have been used extensively due to its flexibility in handling various constraints and controlling runtime vs solution quality trade-off. The stochastic hill-climbing algorithms such as simulated annealing, tabu search, and genetic algorithms can move to higher-cost neighboring solutions in order to escape local minimal during the search based on local perturbation of the solution.

The clustering algorithms are commonly used to deal with increasing problem sizes. The netlist modules are divided into many small clusters, and these clusters form the new nodes of a smaller, coarser netlist. Then, the subsequent partitioning performs on top of the coarser netlist. The multilevel algorithms apply clustering repeatedly to build multilevel clustering hierarchy. Partitioning can then be performed on each level of the hierarchy from top to bottom while projecting partitioning information.

1.5 Conditional partitioning

Partitioning may be restricted and it might be without restrictions. If no restrictions have been put on partitioning, then partitioning is a simple division operation and is termed **unconditional or unrestricted partitioning**. If restrictions are put on partitioning, then the restricted partitioning is a conditional division operation and is termed a **conditional or restricted partitioning**.

Restrictions may be achievable and they might be unachievable. The achievable restrictions are termed **possible restrictions,** and the unachievable restrictions are termed **impossible restrictions**. For example, partitioning a set of five elements into two subsets such that the two subsets have the same number of elements is an impossible restriction.

In Section 1.3, partitioning has been discussed subject to the basic restrictions that sub-networks will be balanced and inter-sub-network connections (i.e. cut-edges) are minimized. In practice, other conditions may arise which affect the outcome of partitioning. In many practical applications, where a graph is used as a descriptor of a physical system, a graphical object such as a node may possess attributes other than an identifying index. An example of this is in the case of an electrical network, where power or current flows around the network may be described (using Kirchhoff's laws) in terms of the voltage level, which is present at each node of the network. Nodal voltage is therefore such an attribute in this case. The set of nodal voltages are present in mathematical expressions, which form the basis for computations, which in turn are the subject of partitioning. This has the effect of placing constraints on the balance sought between groups of sub-network nodes. This situation is examined in more detail for a particular application, in chapter two.

1.6 The thesis organization

This thesis contains a description of some new procedures for the simplification of computational solutions to a conditional network-partitioning (NP-hard) problem. Throughout the investigation, the conditions on partitioning are drawn from an application in the monitoring and control of electric power systems, known as Power Systems State Estimation (PSSE).

PSSE enables a validated database of electrical network information to be constructed from a model of the network and a set of measurements taken from it. The work described in this thesis has not been concerned with the development of PSSE algorithms. One existing PSSE technique, however, has features which are particularly attractive for partitioning. This is achieved through a particular decomposition of the set of algebraic equations defining PSSE for a global network. In order that the conditions which then apply to the partitioning procedure may be clearly understood, this particular method for PSSE is reviewed in Chapter 2. Following the establishment of conditions, Chapter 2 concludes with some basic network definitions and properties.

Given the constraints applicable to the partitioning problem to be investigated, it is useful to examine the characteristics of existing heuristic procedures for partitioning. Three recent contributions are examined and reviewed in Chapter 3.

Following definition of the conditions for partitioning and a review of some partitioning approaches, it is sensible to seek properties of a candidate network which may be utilised to bring about some improvement in a computational partitioning procedure. The field of graph theory was selected as a starting point. The result of searching graph theory for exploitable equations or properties is presented in Chapter 4. The relationship between a network and an associated spanning tree is of particular interest; the properties of a spanning tree are introduced followed by introduction of the cut concept. A procedure for partitioning based on a spanning tree and its cut-set is developed, and the effect on the solution of selection of an initial spanning tree is examined.

The restrictions imposed by the special conditions of PSSE, and their effect on the partitioning procedure are examined in more detail in Chapter 5. A theoretical foundation to find the number of internal nodes of the k subsystems and the number of boundary nodes in the network in the balanced case without partitioning the network is described.

The limitations of the partitioning technique of Chapter 4 may be reduced with the use of a network property termed the covering set of nodes, in order to partition every spanning tree of the network, and then to partition the network. The network has many different covering sets. To minimise the search, the technique is designed to use a special covering set termed the set with higher degree. The covering set concept and a new faster technique is described in Chapter 6. To test the speed and validity of the covering set approach, the technique has been applied over all spanning trees of the standard IEEE 14-bus network. Some simulation results are presented.

Chapter 7 opens with a description of a new property known as the edge phenomenon in the plane. The edge phenomenon is then used to explore with proofs many new different network properties. If n and m are given, then edges can be connected in many different ways. Identifying all possible connections presents severe practical difficulties, particularly for large networks. Under the paradigm of the edge phenomenon, the difficulties are reduced significantly. The parameters of the network properties are used to define uniquely the network entity. Each connection can be defined uniquely.

Even the thesis target was not only to find a partitioning technique to satisfy the DSE restrictions, but also to simply the network partitioning problem

Finally, some of the properties introduced in Chapter 7 are applied to a further heuristic partitioning technique in Chapter 8. The technique defines the cut line concept and then presents a special cut line termed the I-I cut-line. The I-I cut-line is used to partition the network by partitioning the external closed path into equal parts.

Concluding remarks and recommendations for extensions of this research are given in chapter 9.

The network basic definitions and notations are introduced in Appendix A.

Chapter 2

Review of PSSE algorithms

2.1 General

In modern computer-based operation of an electric power system (PS), it is essential to provide a validated database, updated at regular intervals, which describes with acceptable accuracy the current state of the PS network. This is the function of the PSSE. The output of the algorithm is a set of estimates of the state-variables for the system, together with a measure of the accuracy of these estimates. The computations are based on input data which typically comprise a model of the network in suitable form, together with a set of measurements received from the PS network via a telemetry system. These measurements usually consist of a mix of nodal voltage magnitudes, line power flow and/or power injections.

Since the PS model is non-linear algebraic, the solution is iterative. The PSSE algorithm must have the qualities of good numerical accuracy in computation with respect to the data supplied and reliable and rapid convergence. Also, since regular updates are required, all computations plus any data communication operations must be completed within a defined sampling time-period.

There are numerous approaches to the design of state estimators, the most commonly encountered of which is the weighted least squares (WLS) technique [3]. When applied to a given PS network and implemented in a single processor at a control centre, the WLS algorithm is termed 'integrated state-estimator' (ISE).

The ISE requires all measured data to be communicated to a single location, which can lead to heavy information transfer from many sites to the control centre. The performance of the ISE has been tested against the system size [3]; it is shown that the computing requirements and numerical errors of ISE increase super-linearly with the system size. It follows that, for power systems of large size, it may not be possible to meet the required execution time for ISE due to the high amount of computation involved.

This disadvantage of a single-processor solution to PSSE has led to the development of alternative techniques, [3, 47, 49] which utilize partitioning of the PS network model followed by decomposition of the PSSE algorithm and implementation in a two-level hierarchical processor configuration.

These different techniques are based on splitting the state estimation problem into a number of smaller sub-problems. A topologically partitioned solution is proposed in [75], which used the output of an observability algorithm to rearrange the measurement vector into non-critical and critical sub-vectors. This method needs high proportion of critical measurements, and involves heavy exchanges of information during its iterative procedure.

The original system was divided into a number of subsystems which overlapped at boundary nodes [44a], or tie-lines. [44b]. In these approaches, the overall model for the whole system is generally related to subsystems using a diakoptical representation. The estimation is obtained by using two-level structure of hierarchical solution [44a], or by an alternating sequential parallel computer system [44b]. These approaches have common disadvantages of heavy data transfers at each iteration and the relatively high numbers of iterations could lead to time delays which may be unacceptable in a real-time environment.

It was clear from the outset that the design of a practically useful partitioning method would be influenced by the decomposed SE scheme with which it is to be used. A decision was therefore taken to select, somewhat arbitrarily, one method, which would provide a set of constraints for the study.

The approach described in [49] has features which lead to a simple and particularly well-defined set of constraints, and was therefore chosen as the application on which design of a partitioning procedure would be.

The approach described in [49] has particular practical advantages. Termed the 'decomposed state-estimator (DSE), the algorithm is iterative between two computational levels. The lower level task is shared by k processors operating in parallel, each one estimating the states of a single subsystem. The upper-level task is to co-ordinate the lower-level results, and is performed in a single processor. Per iteration, computation therefore consists of the parallel lower-order task in series with the single upper level task; this combination is iterated to convergence. The advantage in overall computational performance is derived from the reduced order of each of the parallel computations at lower level, together with an upper-level computation which is of low order if the subsystems can be created by partitioning so that the number of inter-subsystem connections is low.

This leads to a broad objective for partitioning: the network is to be divided into k non-overlapping, balanced subsystems, so that the number of inter-connections between these subsystems is minimum. The minimum number of inter-connections is identical with the minimum number of cutting edges.

The ISE and DSE methods are considered in further detail in the following two sections.

2.2 The integrated state estimation algorithm

The objective of ISE is to determine the best estimate, in the WLS sense, of an overall state vector $\underline{x}$ from an available measurement vector $\underline{z}$ which is subjected to uncertainty $\underline{e}$. The measurement vector $\underline{z}$ is considered to be related to the state variable $\underline{x}$ by

$$\underline{z} = h(\underline{x}) + \underline{e} \qquad (2.1)$$

where:

$\underline{\mathbf{x}}$ is a vector of n state variables, defined as the voltage magnitude at each node, together with the phase angle at each node except the slack node;

$\underline{\mathbf{z}}$ is a vector of m measured quantities comprising active and reactive nodal power injections, and nodal voltage magnitudes;

$\underline{\mathbf{e}}$ is a random m vector representing measurement uncertainty, in the form of random bias or noise;

and

$\underline{\mathbf{h}}$ is an m vector of the non-linear observation functions based on the application of Ohm's and Kirchoff's laws to the power system network.

$\underline{\mathbf{e}}$ is considered to be a random process with statistics:

mean: $E[\underline{\mathbf{e}}] = 0$;

and covariance: $E[\underline{\mathbf{e}}\ \underline{\mathbf{e}}^T] = R > 0$

A performance index for the overall system is given by:

$$J = \frac{1}{2}\ \|\underline{z} - \underline{h}(\underline{x})\|^2_{R^{-1}}\ . \qquad (2.2)$$

The weighted least square (WLS) estimator $\hat{\underline{x}}$ for the system states is then given by a convergence of:

$$\Omega(i).\Delta\underline{\mathbf{x}}(i) = H^T(i)\, R^{-1}.\Delta\underline{\mathbf{z}}(i); \qquad (2.3)$$

in which:

$\Delta\underline{x}(i) = \underline{x}(i+1) - \underline{x}(i)$;

$\Omega(i) = H^T(i)\, R^{-1}\, H(i)$ where Ω is the gain matrix ;

$\Delta\underline{z}(i) = \underline{z} - \underline{h}(\underline{x})$ with $\underline{h}(\underline{x})$ evaluated at $\underline{x} = \underline{x}(i)$;

H= •h/•x, a Jacobian matrix;

and i is the iteration index.

Let $\hat{H}$ and $\hat{\Omega}$ denote H and Ω respectively at convergence. With $\underline{\mathbf{e}}$ so defined as a random process, $\hat{\underline{x}}$ may be interpreted as a linear-unbiased minimum-variance (LUMV) estimator for $\underline{\mathbf{x}}$ with covariance of estimation error given by

$$E[\tilde{\underline{x}}\,\tilde{\underline{x}}^T] = \hat{\Omega}^{-1}, \text{ where } \tilde{\underline{x}} = \underline{x} - \hat{\underline{x}}.$$

The condition: Rank {H(i)} = n; at each iteration of equation (2.3) is sufficient to ensure convergence of x(i) to give a local minimum for J; the iteration then has quadratic convergence properties.

2.3 The decomposed state estimation algorithm

An overall network of n nodes is decomposed into k non-overlapping subsystems interconnected by ties, which are physically either lines or transformers. The k subsystems are defined uniquely by cuts through the ties. Ties are terminated within adjacent subsystems at nodes termed **"boundary nodes"**. Subsystem i contains n_i nodes, n_{ib} of which are **boundary nodes**. The remaining $n_{ir} = n_i - n_{ib}$ nodes are termed **"internal nodes"**. One slack node, at which the voltage phase angle is assigned to be zero reference, is selected for the entire network. By convention only, the slack node is assigned to an internal node in subsystem 1.

A mathematical model for subsystem i may then be based upon:

internal states $\underline{\mathbf{x}}_{ir} \in R^{n_{ir}}$;

boundary states $\underline{\mathbf{x}}_{ib} \in R^{n_{ib}}$;

and the boundary states of adjacent subsystems.

A key features of this decomposition is that the internal states $\underline{x}_{jr}$ for other subsystems ($j \neq i$) will not appear in the measurement model for subsystem i. The overall set of measurements $\underline{\mathbf{z}}$ is distributed exclusively and exhaustively amongst the k subsystems; a measurement taken on an inter-subsystem tie may be assigned to either of the adjacent subsystems. The measurement model for subsystem i will then be expressible in general form as:

$$\underline{\mathbf{z}}_i = \underline{\mathbf{h}}_i\,(\underline{\mathbf{x}}_{ir}, \underline{\mathbf{x}}_b) + \underline{\mathbf{e}}_i\,;$$

where

$$\underline{\mathbf{z}}^T = [\underline{\mathbf{z}}_1^T\ \underline{\mathbf{z}}_2^T\ \text{- - -}\ \underline{\mathbf{z}}_k^T] \text{ with } \underline{\mathbf{z}}_i \in R^{m_i}\,;$$

and

$$\underline{\mathbf{x}}_b^T = [\underline{\mathbf{x}}_{1b}^T\ \underline{\mathbf{x}}_{2b}^T\ \text{- - -}\ \underline{\mathbf{x}}_{kb}^T] \text{ with } \underline{x}_{ib} \in R^{n_{ib}}\ .$$

At lower level, each of the k parallel processors is assigned to a subsystem. All computation at this level is carried out simultaneously for all k subsystems but independently of each other.

A model of DSE as reported in [49] can be represented by the master and slave configuration shown in Figure 2.1.

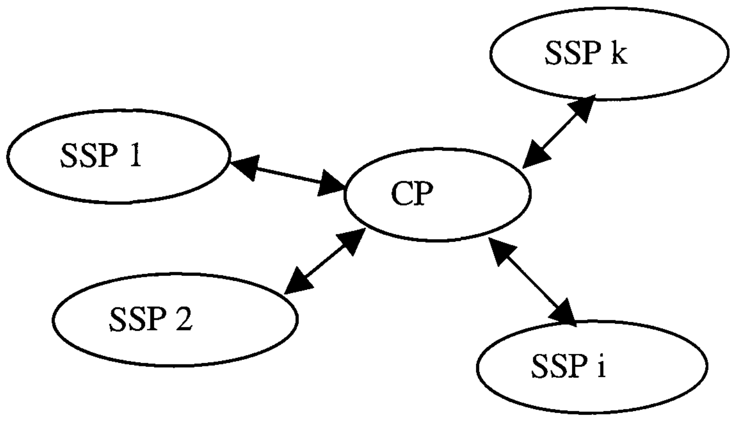


Figure 2.1 Two level state estimator
CP: Co-ordination Processor
SSP i: Subsystem Processor i

The DSE algorithm is iterative between the two computation levels. The scheme does not require communication between processors at lower-level.

The two-level iterative procedure for the DSE is summarised in Table 2.1.

At lower level (k-parallel processors),	At the upper level (single processor),
1. Initialise with $\underline{z}_i$, $\underline{x}_{ir}(0)$, $\underline{x}_b(0)$. set j=0.	
2. Compute: $\underline{h}_i(\underline{x}_{ir}(j), \underline{x}_b(j))$; $\Delta\underline{z}_i(j)$; $\underline{H}_{ir}(j)$; $\underline{H}_{ib}(j)$; $H^T_{ir}(j)R_i^{-1}$; $\Omega_{ir}(j)$; $\Omega_{ir}^{-1}(j)$; $W_{ir}(j)$; $W_{ir}(j)H_{ib}(j)$; and $\lfloor G_i(j) \;\vdots\; \underline{g}_i(j) \rfloor$. 2= Send $\lfloor G_i(j) \;\vdots\; \underline{g}_i(j) \rfloor$ to upper level; $n_b(n_b+1)$ elements in k-parallel.	3. Compute $\left[G \;\vdots\; \underline{g}\right] = \sum_{i=1}^{k} \left[G_i(j) \;\vdots\; \underline{g}_i\right]$; and $\Delta\underline{x}_b(j)$ from $G(j).\Delta\underline{x}_b(j) = g(j)$. 3= Send $\Delta\underline{x}_b(j)$ in k-parallel to lower level; n_b elements.
4.Compute: $\Delta\underline{r}_{ib}(j) = \Delta\underline{z}_i - H_{ib}.\Delta\underline{x}_b(j)$; and $\Delta\underline{x}_{ir}(j)$ from $\Omega_{ir}(j).\Delta\underline{x}_{ir}(j) = H^T_{ir}(j)\,R_i^{-1}.\Delta\underline{r}_{ib}(j)$	
5.Set j=j+1. Compute: $\underline{x}_{ir}(j)$; $\underline{x}_b(j)$. If convergence not reached, go to 2; else $\underline{x}_{ir}(j) = \hat{\underline{x}}_{ir}$; and $\underline{x}_b(j) = \hat{\underline{x}}_b$. End of algorithm.	

Table 2.1 The two-level iterative procedure for the SE

2.4 The DSE constraints

The hierarchical decomposition of the integrated state estimation problem into two levels introduces new definitions and constraints which have a direct influence on the specification of the network-partitioning problem. Without these constraints, the problem would be to partition the network such that (i) the k subsystems are equal or balanced and (ii) to minimize the number of connections between the k subsystems.

Under the new hierarchical decomposition of the state estimation paradigm, however, the cuts which bring about partitioning define: each subsystem; the internal nodes of each subsystem; and the boundary nodes of each subsystem which together form the set of global boundary nodes. The amount of computation at each level and the volume of communication between the two levels have been determined by the DSE. Thus partitioning has a direct consequence on DSE performance. To meet the computational performance requirements of the DSE, partitioning must consider the size of the global interconnected area and the size of internal areas of each subsystem produced by partitioning (measured by the number of nodes) as well as its impact on performance (measured by the amount of computation and communication). Reduction in subsystem size clearly produces a reduction in the computation at lower level. At the same time, reducing subsystem size by partitioning generally leads to an increase in the number of interconnections between subsystems, which in turn leads to an increase in volume of communication and a corresponding increase in the amount of computation at upper level. In addition, balancing the sizes of the k subsystems has the effect of approximately equalizing the amounts of parallel computation at lower level, so that there is no time delay or waiting in the parallel operation. The computational tasks at both lower- and upper-level are basically the solution of respective sets of linear algebraic equations. The size of computation at the upper level of the DSE is therefore $O(n_b^3)$. The amount of computation of each independent process at the lower level could similarly be considered, in terms of dimension only, as $O(n_{ir}^3)$. It should be stressed, however, that the lower-level problems are relatively sparse, and hence sparsity programming [60] techniques would in practice be employed, for computational advantage. In this case, the amount of computation at lower level would be more realistically represented as $O(n_{ir}^2)$.

2.4.1 The computation constraints in the lower level

In the lower level, k parallel processors are use to estimate the internal nodes of k subsystems; each processor estimates the internal nodes of one subsystem. The i^{th} processor has n_{ir} internal nodes. The DSE algorithm determines the amount of computation at each process at the lower level to be $O(n_{ir}^3)$ or $O(n_{ir}^2)$ if sparsity programming is employed. If the values of n_{ir} are different for $i = 1, 2, ..., k$, this will cause a delay in computation at lower level. The processor with maximum n_{ir} will require more time to complete its computation than the other subsystem processors which are dealing with lower order problems. Communication from lower to upper level cannot proceed until the lower level task has been completed. The upper level processor does not start computing until it receives the data from all k processors. Communication delays arise from the time taken to transfer data between levels; this clearly increases as the volume of data increases. Since the volume of data in either direction is a function of the number of boundary nodes, this means that the number of interconnections between subsystems needs to be kept low if the communications delay is to be kept low.

It is therefore clear that the constraints imposed by the DSE problem produce the requirement that the balance to be sought at lower level is not between numbers of subsystem nodes, but between subsystem **internal** nodes.

2.4.2 The computation constraints at the upper level

After computation of the lower level tasks, the data from the k processors at the lower level are sent to the single processor at the upper level, in order to compute updates of the n_b boundary nodes. The computation size is $O(n_b^3)$. If $n_b > n_{ir}$, then $n_b^3 >> n_{ir}^3$, i.e. overall computational performance will be dominated by that of upper level processing, which is undesirable. For example, if $n_b = 5 > n_{ir} = 3$, then $n_b^3 = 125 >> n_{ir}^3 = 27$. The computation time at the upper level processor is about 5 times that of the i^{th} processor at the lower level.

To avoid such a situation, the size of n_b needs to be not greater than the largest value of n_{ir}; overall performance is improved if n_b can be made as small as possible. In probability, a decrease in n_b will be obtained if k, the number of subsystems, is decreased.

Many proposed partitioning techniques do not consider delays which result from computation or communication, while others do not classify the network nodes into internal and boundary nodes nor do they consider subsystem size. As a result, there is a strong need for method which gives a balanced partitioning, considering both the number of boundary nodes, fast computation and communication delay, in providing an equal or balanced partitioning.

Thus, the partitioning problem can be defined as partitioning the DSE network into k sub-networks such that the number of the k sub-networks internal nodes are equal or balanced with each other, and the number of the total boundary nodes are less than, equal or balanced with the number of internal nodes of the i^{th} subsystem. These new definitions and specifications are of great significance for any partitioning techniques for the DSE to perform efficiently.

Chapter 3

Review of recent partitioning techniques

3.1 General

In this chapter, three recent partitioning techniques are presented. Each technique introduces a different approach to solve the network-partitioning problem. The first approach has been introduced by [33]. It presents a development of some heuristic algorithms to partition a PS network into two or more sub-networks. The proposed heuristic algorithm partitions a spanning tree of a PS network. These partitioning algorithms are based on using an integer linear programming (ILP) eigenvector based approach to have a good initial partition, and then on using an interchange method to obtain the optimal partition.

The second approach has been introduced by [39]. The new approach is based on the observation that the edges of a graph's minimum cut form a very small fraction of the graph's edges so that a randomly selected edge is unlikely to be in the minimum cut. Therefore, if an edge is chosen at random and its end-points are contracted into a single vertex, the probability is high that the minimum cut will be unaffected. Therefore, the minimum cut are found by repeatedly choosing and contracting random edges until the minimum cut is apparent.

The third approach has been developed by [32], to partition a PS network for the purpose of the decomposed state estimator. The technique is based on markov chains process. It partitions a spanning tree into k sub-spanning trees and then finds from the network the minimum cuts.

The three approaches do not classify the network nodes into internal and boundary nodes, but they can be modified to satisfy the DSE restrictions.

3.2 The Integer-linear-programming approach

3.2.1 General

This approach has been introduced by [33]. It presents a development of some heuristic algorithms to partition an observable PSSE network into two or more observable sub-networks. The proposed heuristic algorithm partitions a spanning tree of an observable PSSE network. These partitioning algorithms are based on using an integer linear programming (ILP) eigenvector based approach to have a good initial partition, and then on using an interchange method to obtain the optimal partition.

3.2.2 Interchange methods for partitioning

Iterative improvement algorithms start with a random partition and try to optimise it by making small local changes such as successively shifting (or moving) of modules from one block to another. Kernighan and Lin [41] described a heuristic procedure for netlets (hypergraphs) partitioning, which became the basis for most of the iterative improvement partitioning algorithms.

The Kernighan and Lin algorithm, which starts with a given random partition, consists of a series of passes. In each pass, two modules are interchanged in turn until all nodes are moved. Each pass consists of a series of interactions. At each iteration the modules to be moved are chosen from among the ones that have not yet been moved during the pass. The modules to be interchanged or moved are chosen so that the maximum decrease in cut-set size (i.e. minimum number of nets cut by a partition) may be obtained (or minimum increase if no decrease is feasible). At the end of each pass, since all modules must have been interchanged, the cut-set size of the partitioned blocks should be exactly the same as it was at the beginning of the pass. The partitions produced during a pass are examined and the one with the smallest cut-set size is chosen as the starting (initial) partition for the next pass. Passes are performed until no improvement in cut-set size can be obtained.

Fiduccia and Mattheyeses [22] introduced modifications to the Kernighan and Lin algorithm. One of the significant modifications suggested by them is to move one module at a time instead of switching pairs. This modification allows for more flexibility in the size of the partitioned blocks.

Partitioning simple netlist example:

This example gives a general idea of how the interchange method works in partitioning a netlist, using Fiduccia and Mattheyses approach. Consider the partitioning of the following five-modules, three-net netlist example shown in figure 3.2.1 into two blocks. Let the upper bound of any block be four (i.e. the maximum number of modules inside any block at any iteration and during any pass is four).

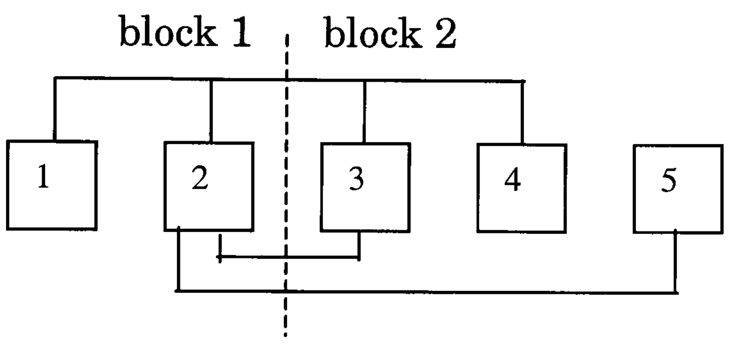


Figure 3.2.1 Five-module, three-net netlist example

Define Δ_{ij} as the gain in cut-set size when module i moves to block j. Assume the starting random partition given in figure 3.2.1 that places modules 1 and 2 in block 1, and modules 3, 4 and 5 in block 2. This initial partition, as shown in Figure 3.2.1, cuts three nets. Now start with the first pass, as explained.

Pass 1. Iteration 1

$\Delta_{12} = 0 \quad \Delta_{22} = +2 \quad \Delta_{31} = +1 \quad \Delta_{41} = 0 \quad \Delta_{51} = +1$

It is obvious from iteration 1 that the maximum decrease in cut-set size that can be obtained in this iteration is by moving module 2 to block 2; this partition cuts 1 net only, as shown in Figure 3.2.2a Module 2 is locked in block 2 and cannot be moved any more in pass 1. Moreover, the number of modules in block 2 does not exceed the upper bound limit (i.e. four modules).

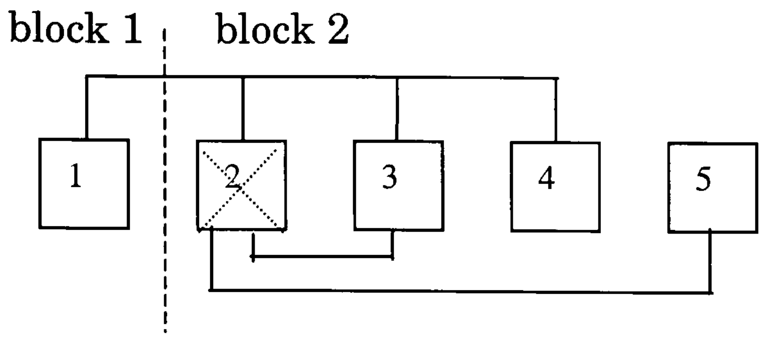


Figure 3.2.2a Shifting of module 2 to block 2

Pass2.Iteration 2

$\Delta_{12} =$ violates upper bound of block 2.

$\Delta_{21} =$ locked in block 2

$\Delta_{31} = -1 \qquad \Delta_{41} = 0 \quad \Delta_{51} = -1$.

It is obvious from iteration 2 that the cut-set size cannot be further reduced. Since moving module 4 to block 1 does not affect the cutest size, this is the best move in this iteration, as shown in figure 3.2.2B. Module 4 is locked in block 1 and cannot be moved any more in this pass.

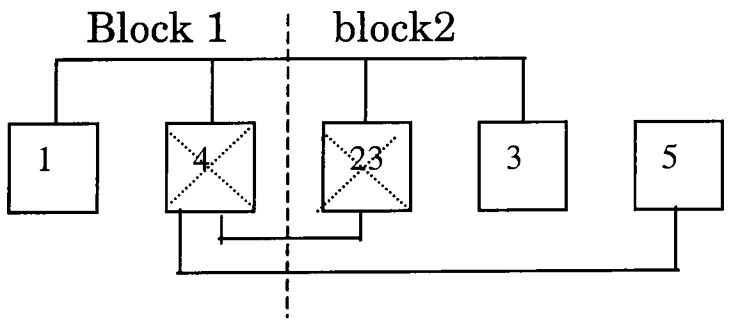


Figure 3.2.2b Shifting of module 4 to block 1

Proceeding until the end of this pass (i.e. at iteration 5) the final partition will look like the one shown in figure 3.2.2c.

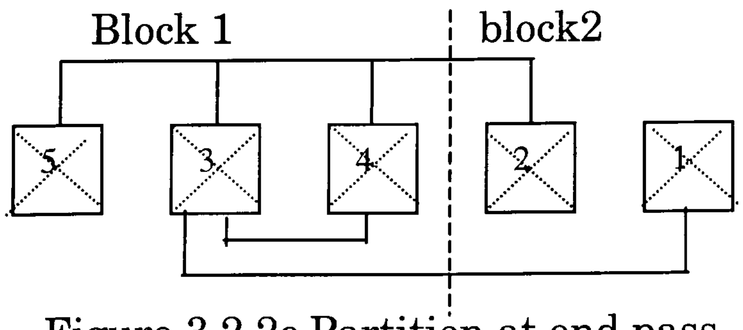


Figure 3.2.2c Partition at end pass

The number of cuts is exactly same as the starting (initial) partition and this is due to the fact that all modules have been moved and got locked; this is the end of pass 1.

Investigation of the smallest cut-set size throughout this pass shows that partition of the first three iterations has the smallest number of cuts. Therefore, any one of these three partitions could be chosen as the starting partition for the next pass. The procedure continues until a pre-specified number of passes is reached. In this example, since the optimal cut-set size is one and has been obtained from the first pass, one can terminate at this pass.

3.2.3 ILP eigenvector-based approach

In this section an integer-linear-programming eigenvector (ILP) based approach is presented. The ILP is to find a good initial partition between the busses of an undirected graph. This approach was, first presented by Barnes [4]. A brief summary of Barnes approach follows:

Assume that an undirected graph G of n busses needed to be partitioned into k disjoints blocks of size $m_1, m_2, \cdots, m_k$. Define the following 0-1 integer variable:

$$x_{ij} = \begin{cases} 1 \text{ if bus i is in block j} \\ \quad i = 1, \cdots, n; j = 1, \cdots, k \\ 0 \text{ otherwise} \end{cases}.$$

Let a_{st} be the number of edges connecting busses s and t, $s \neq t$, and let $a_{ss} = 0$, $s, t = 1, \cdots, n$. Let A denote the n x n matrix that corresponds to the adjacency matrix (or connectivity matrix) of the undirected graph G. Let v_{ij} be defined as the ith component of the eigenvector corresponding to the largest eigenvalue jth largest eigenvalue of the adjacency matrix A of the undirected graph. Barnes shows that the solution of the following ILP transportation problem gives an approximate solution to the undirected graph G partitioning problem:

$$\max\left\{\sum_{i=1}^{n}\sum_{j=1}^{k}\frac{v_{ij}}{\sqrt{m_j}}x_{ij}\right\}$$

subject to

$$\sum_{i=1}^{n}x_{ij}=m_j,\quad j=1,\cdots,k$$

$$\sum_{j=1}^{k}x_{ij}=1,\quad i=1,\cdots,n$$

$$x_{ij}\geq 0,\quad i=1,\cdots,n;\quad j=1,\cdots,k$$

The partitioning given by the solution of the transportation problem (expr.1) usually places most of the busses of G in the correct blocks.

In the two-block case (i.e. k=2), the transportation problem can be further simplified by replacing x_{i2} by $1-x_{i1}$ [11]. Let $x_i=x_{i1}$, then the transportation problem could be replaced to the following ILP {0,1}-Knapsack problem:

$$\max\left\{\sum_{i=1}^{n}\left[\frac{v_{i1}}{\sqrt{m_1}}-\frac{v_{i2}}{\sqrt{m_2}}\right]x_i\right\} \tag{3.1}$$

Subject to

$$\sum_{i=1}^{n}x_i=m_1 \quad ; \tag{3.2}$$

$$0\leq x_i\leq 1,\quad i=1,\cdots,n$$

The solution to the knapsack problem (expr.2) can be obtained (without solving the knapsack problem [63]) by sorting the objective coefficients in non-increasing order and setting $x_i=1$ for the first m_1 variables in the sorted list (all other variables are set to zero). An interchange technique, whose initial partitioning is one obtained from the solution of the problem expr.2, can be applied afterwards to obtain a good final partition.

The ILP eigenvector-based approach is used to obtain a good initial partition for a spanning tree of an observable PSSE network. An interchange method can be applied afterwards to obtain the optimal partition of the spanning tree (i.e. every spanning tree is full of rank).

3.2.3.1 Algorithm for equi-partition of a spanning tree

A heuristic algorithm is proposed to partition a spanning tree of an observable PSSE network into two blocks of buses of sizes m_1 and m_2. Optimality can be achieved when the number of cuts between the two partitioned blocks is equal to one (i.e. the existent of two sub-spanning trees of full rank). The proposed algorithm involves the following steps:

Algorithm 3.2.1:

a- Obtain a spanning tree of an observable PSSE network.

b- Use the knapsack integer-linear-problem of equation (3.2) to partition the spanning tree into two blocks of busses with block size m_1 and m_2. If the number of cuts is one, stop, optimality has been reached: otherwise, proceed to the next step.

c- Interchange those busses that are connected between the two partitioned blocks (while maintaining the block sizes m_1 and m_2 fixed) until optimality is reached. If so, stop, otherwise continue to the next step.

d- Allow one or more of those buses that are connected between the two partitioned blocks to move from one block to the other (i.e. change the block size m_1 and m_2) such that the size of the new blocks does not violate a pre-specified limit. If optimality is reached, stop, otherwise continue to the next step.

e- The existing spanning tree cannot be partitioned. Optimality (i.e. number of cuts between the partitioned blocks cannot be one). Find another possible spanning tree and go back to step 2. If there is no more possible spanning trees that can be partitioned optimally, stop; the system cannot be partitioned into two observable sub-networks.

Once the spanning tree is optimally partitioned into sub-spanning trees of full rank (i.e. the number of cuts is one), the actual interconnected lines between the two partitioned sub-networks can be obtained directly from the original network graph.

3.2.3.2 Equi-partition of IEEE-14 network

Consider partitioning of the IEEE-14-bus observable PSSE network, shown in Figure 3.2.3, into two observable sub-networks each with block size of seven (i.e. $m_1 = 7$ and $m_2 = 7$).

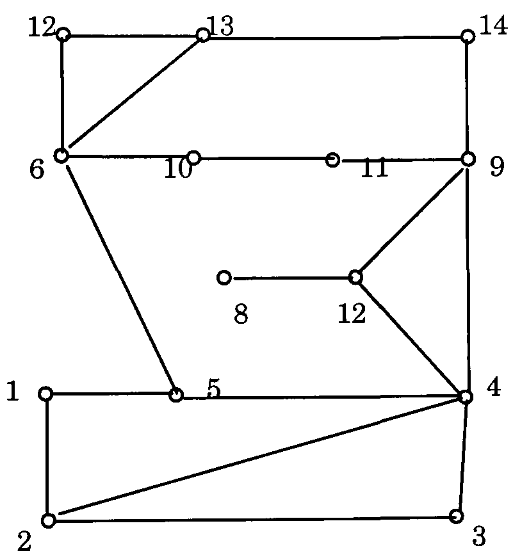


Figure 3.2.3 The IEEE-14 network

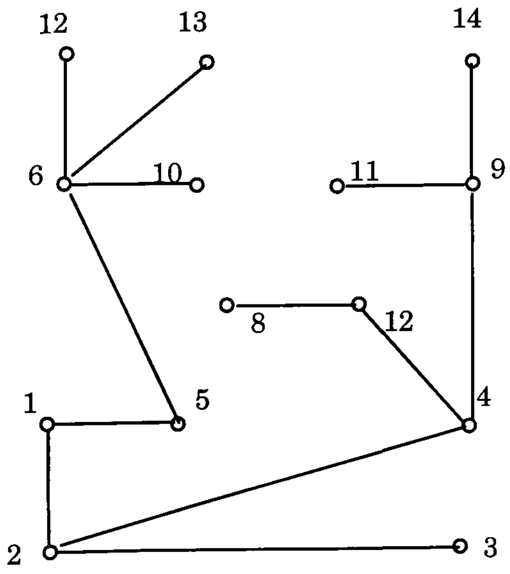


Figure 3.2.4 A spanning tree

Allow changes in the size of any of the two blocks such that the maximum allowable difference between the two blocks be zero (i.e. retaining the size of each block to be seven). A possible spanning of this network is shown in Figure 3.2.4.

The largest, the second largest eigenvalues and the corresponding eigenvectors have been calculated.
The solution of the knapsack problem (expr. 2) after sorting its coefficients leads to

$$x_4 = x_9 = x_3 = x_7 = x_{11} = x_{14} = x_2 = 1,$$

and

$$x_8 = x_1 = x_5 = x_{12} = x_{13} = x_{10} = x_6 = 0.$$

This solution cuts two of the spanning tree branches. According to step 3 of algorithm 1, if node 8 is interchanged with node 2, the optimal partition can be obtained (number of cuts is one). The optimal partitioned buses are shown in Figure 3.2.5.

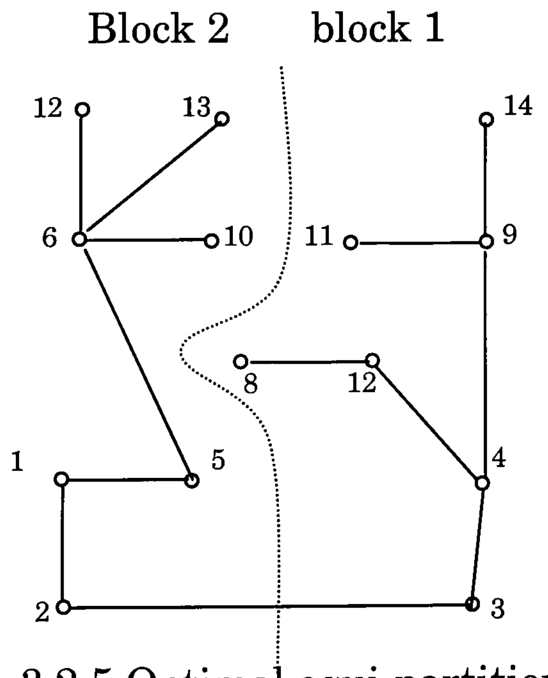


Figure 3.2.5 Optimal equi-partitioning of spanning tree

This optimal partition cuts 5 lines of the IEEE 14-bus network graph, as shown in Figure 3.2.6, and the two partitioned sub-networks are still observable (i.e. every sub-network has a sub-spanning tree of full rank).

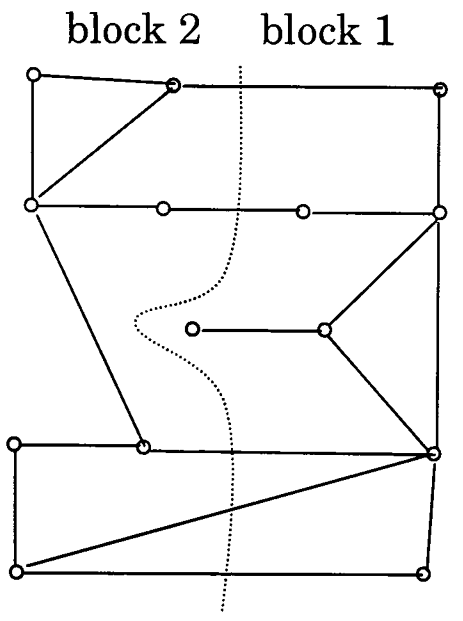


Figure 3.2.6. Partitioning of IEEE 14-bus into two observable sub-networks.

3.2.3.3 Algorithm for multi-partitioning of spanning tree

Multi-partitioning of a spanning tree using the transportation problem (expr. 1) directly is not an easy task; it requires the solution of kn variables with k+n constrains. This subsection presents a heuristic multi-partitioning algorithm of a spanning tree that avoids the complexity associated with using the transportation problem (expr. 1) directly. Optimality can be achieved when the number of cuts between the k partitioned sub-spanning trees is equal to k-1. The following heuristic steps of the proposed algorithm leads to multi-partitioning.

Algorithm 3.2.2:

a- Obtain a spanning tree of an observable PSSE network.

b- Use the knapsack interchange-linear-program problem of expr.2 to partition the spanning tree into two blocks of busses with block sizes of m_i and $\sum_{j=i+1}^{k} m_j$ for $i = 1, 2, \cdots, k-1$.

c- For every i in step 2 of this algorithm, follow step 2-5 of algorithm 1 to obtain the optimal partition between every two blocks. Overall optimality can be obtained when the total number of cuts between the k-partitioned blocks of buses is equal to k-1.

Once the spanning tree is optimally partitioned into k sub-spanning trees (i.e. the number of cuts is k-1) the actual interconnected lines between the k sub-networks can be obtained directly from the original network graph.

3.2.3.4 Multi-partitioning of IEEE 14-bus network

Assume the case k=3, i.e. partitioning the IEEE 14-bus network into three sub-networks, and the original size of each sub-network (block) be $m_1 = 4$, and $m_2 = m_3 = 5$. Also allow changes in the size of a block such that the maximum allowable difference between any two blocks be two. The largest and second largest eigenvalues and corresponding eigenvectors have been calculated. According to Algorithm 3.2.2, this multi-partitioning has to be done in two steps.

In the first step, the entire spanning tree is to be partitioned into two blocks with sizes of $m_1 = 4$ and $m_2 + m_3 = 10$, and the second step partitioning the ten-bus into two other blocks of busses with block sizes of $m_2 = 5$ and $m_3 = 5$. The partitioned buses are shown in Figure 3.2.7.

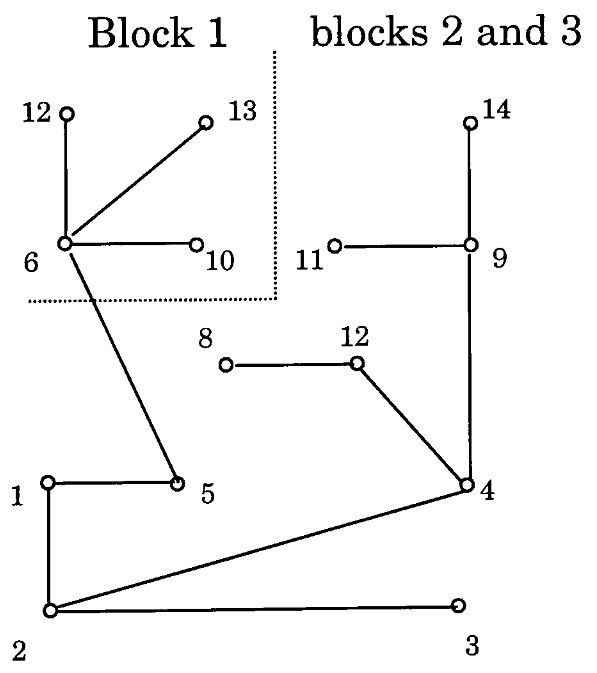


Figure 3.2.7 First step of multi-partitioning the spanning tree.

The next step, according to Algorithm 3.2.2, is to partition the ten buses sub-spanning tree of Figure 3.2.7 into two further blocks, with five buses in each block.

Whatever a pair of buses are interchanged between the two blocks 2 and 3, the minimum number of cuts that can be obtained would still be two. Therefore, this is the only way to partition the ten-bus sub-spanning tree of Figure 3.2.8 without changing the block sizes m_2 and m_3.

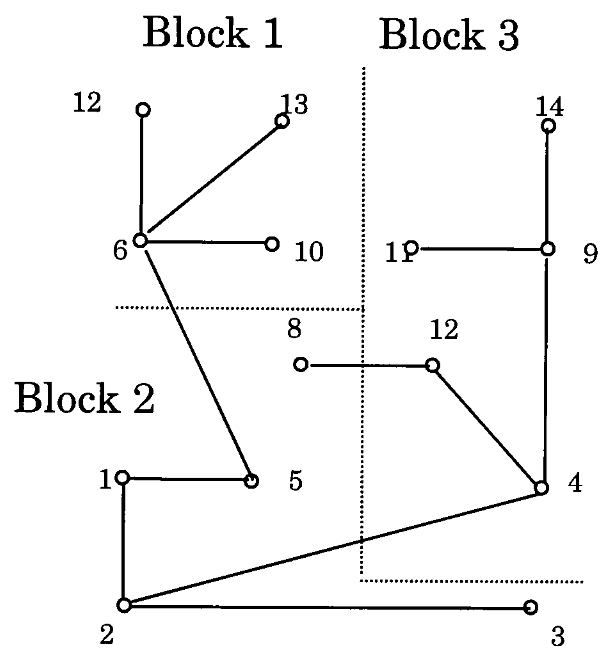


Figure 3.2.8 Second step of multi-partitioning spanning tree

According to step 4 of algorithm 1, if bus 8 from block 2 is allowed to move to block 3 (this is what the interchange method will do within the allowable limit), optimality can be achieved. In this case, only one branch of the ten-bus sub-spanning tree will be cut, as shown in Figure 3.2.9. Therefore, the overall optimal partition of the spanning tree of the IEEE 14-bus network can be obtained.

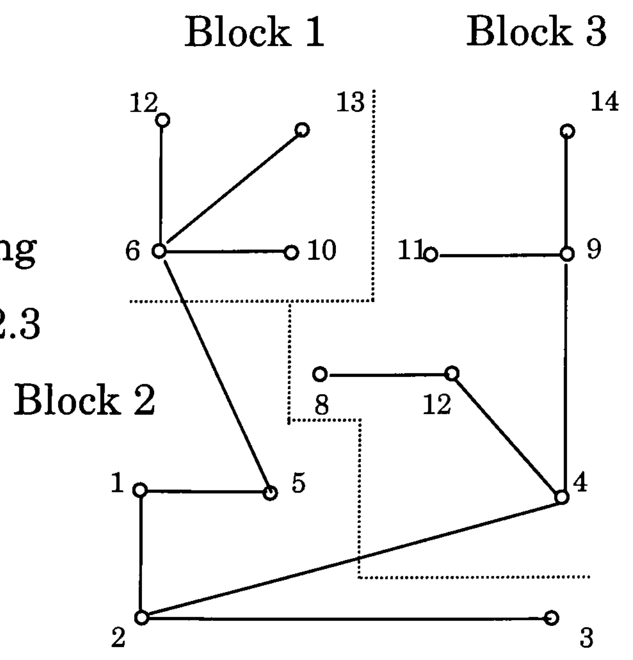


Figure 3.2.9
Optimal multi-partitioning of spanning tree of Fig.3.2.3

3.3 Karger approach to the minimum cut problem

The minimum cut approach has been introduced by [39].

3.3.1 The minimum cut strategy

Given a network which can be represented by an undirected graph with n vertices and m (possibly weighted) edges, it is required to partition the vertices into k nonempty sets so as to minimize the number (or weight) of edges crossing between them. More formally, a cut (A, B) of a graph G is a partition of the vertices of G into two nonempty sets A and B. An edge (v, w) crosses cut (A, B) if one of v and w is in A and the other in B. The value of the cut is the number of edges that cross the cut or, in a weighted graph, the sum of the weights of the edges that cross the cut. The minimum cut problem is to find a cut of minimum value.

Throughout this section, the graph is assumed to be connected, and all edge weights is assumed to be nonnegative.

3.3.2 An abstract formulation of the contraction algorithm

The algorithm is based on the observation that the edges of a graph's minimum cut form a very small fraction of the graph's edges so that a randomly selected edge is unlikely to be in the minimum cut. Therefore, if an edge is chosen at random and its endpoints are contracted into a single vertex, the probability is high that the minimum cut will be unaffected. Therefore, the minimum cut is found by repeatedly choosing and contracting random edges until the minimum cut is apparent.

The algorithm is divided into two stages. In the first stage, an efficient way to implement the repeated selection and contraction of edges, which form a single trial of the contraction algorithm, is introduced. The second stage deals with the need for multiple trials of the contraction algorithm.

3.3.3 The contraction algorithm

The Contraction algorithm uses one fundamental operation, contraction of vertices. To contract two vertices v_1 and v_2 replace them by a vertex v and let the set of edges incident on v be the union of the sets of edges incident on v_1 and v_2.

Edges from v_1 and v_2 that have the same other ends are not merge; instead, multiple instances of those edges are created. However, self loops, formed by edges originally connecting v_1 and v_2, are removed.

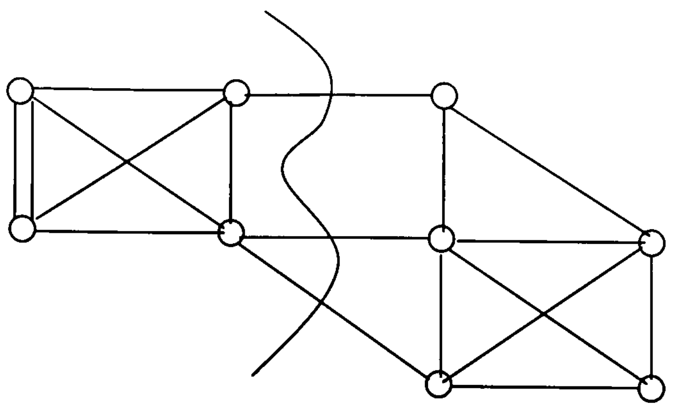


Figure 3.3.1a A graph and a cut-line

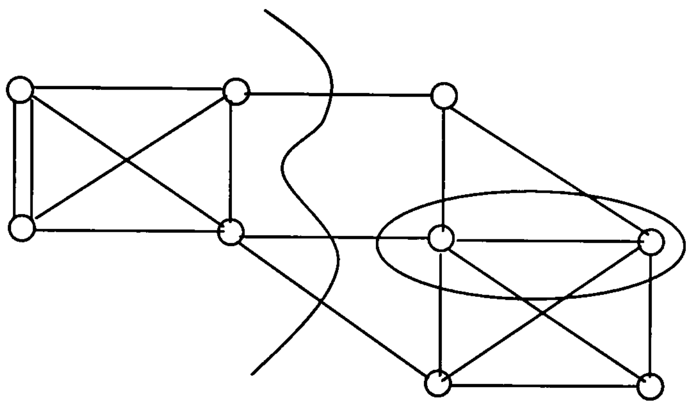


Figure 3.3.1b selecting an edge in the graph

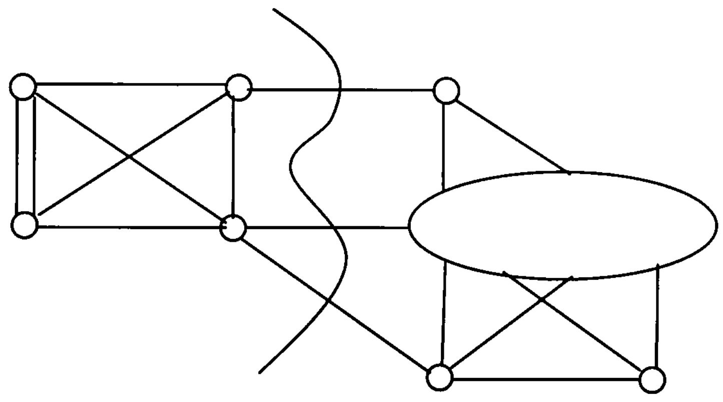


Figure 3.3.1c contracting the edge

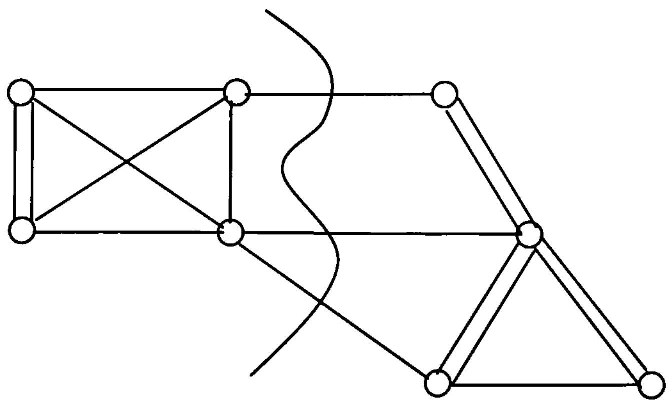


Figure 3.3.1d The graph after contracting one edge

Formally, An edge $e(v_1,v_2)$ is deleted, and each $e(v_1,w)$ or $e(v_2,w)$ is replaced with $e(v,w)$. The rest of the graph remains unchanged. The contracted graph G with e(v,w) contracted is denoted by G/(v,w), (contracting an edge, means contracting the two endpoints of the edge). Extending this definition, for an edge set F let G/f denote the graph produced by contracting all edges in F. An example of an edge contraction is given in figures 3.2.1a, b c and d.

Assume initially that given a graph G (V, E) with n vertices and m edges . The contraction algorithm is based on the idea that since the minimum cut is small, a randomly chosen edge is unlikely to be in the minimum cut. The contraction algorithm, which is described in figures 3.2.1, repeatedly chooses an edge at random and contract it.

When the contraction algorithm terminates, each original vertex has been contracted into one of the two remaining "meta-vertices". This defines a cut of the original graph: each side correspond to the vertices contained in one of the meta-vertices.

More formally, at any point in the algorithm, s(a) can be defined as the set of original graph vertices contracted to a current meta-vertex a. Initially, s(v)=v for each v in V, and whenever the algorithm contracts (v, w) to create vertex x it sets s(a)=s(v) U s(w). A cut (A,B) in the contracted graph corresponds to a cut (A',B') in G, where A' =Us(a) and B'=Us(b).

Note that a cut and its corresponding cut will have the same value, where the value of a cut is defined to be the sum of the weights of the edges crossing the cut.

Procedure 3.1.1: Contract (G)

```
repeat until G has 2 vertices
    choose an edge (v, w) uniformly at random from G
    let G $\leftarrow$ G /(v, w)
return G
```

When the contraction algorithm terminates, yielding a graph with two meta-vertices a and b, the corresponding cut (A, B) in the original graph are A = s(a) and B = s(b). Lemma (3.3.1) has been given its and proof in [39].

Lemma 3.3.1:

A cut (A, B) is output by the Contraction Algorithm if and only if no edge crossing (A,B) is contracted by the algorithm.

Proof:

The only direction is obvious. For the other direction, consider two vertices on opposite sides of the cut (A, B). If they end up in the same meta-vertex, then there must be a path between them consisting of edges that were not contracted. However, any bath between them crosses (A, B), so an edge crossing cut (A, B) would have had to be contracted. This contradicts the hypothesis. Theorem (3.3.1) is given and its proof in [39].

Theorem (3.3.1)

A particular minimum cut in G is returned by the Contraction Algorithm with probability at least

$$\binom{n}{2}^{-1} = \Omega\left(n^{-2}\right)$$

Proof: [39].

3.3.4 Implementation of the contraction algorithm

To implement the contraction algorithm an n x n weighted matrix, W is used. The entry W(u, v) contains the weight of edge (u, v), which can equivalently be viewed as the number of edges connecting u and v. If there is no edge connecting u and v then W(u, v)=0. The optimal (weighted) degree D(u) of each vertex u is also maintained; thus

$$D(u) = \sum_v W(u, v) \ ;$$

Next, is the implementation of the two steps: randomly selecting an edge and performing a contraction.

3.3.4.1 Choosing an edge

A fundamental operation, that is needed to be implemented, is the selection of an edge with probability proportional to its weight. A natural method is the following. First, from edges $e_1,\cdots,e_m$ with weights $w_1,\cdots,w_m$, construct cumulative weights $W_1 = \sum_{i=1}^{k} w_i$ W. Then choose an integer r uniformly at random from $0,\cdots,w_m$ and use binary search to identify the edge e_i such that $W_{i-1} \le r < W_m$. This can be done in O (log W) time.

Assume that given a subroutine called RANDOM-SELECT. The input to Random-Select is a cumulative weight array of length m. Random select returns an integer between 1 and m, with the probability that i is returned being proportional to w_i.

Now, the Random_Select will be used to find an edge to contract it. The goal is to choose an edge (u, v) with probability proportional to W (u, v). To do so, choose a first endpoint u with probability proportional to $D(u)$, and then once u is fixed choose a second endpoint v with probability proportional to $W(u, v)$ W(u, v). Each of these two choices requires $O(n)$ time to construct a cumulative weight array plus one $O(\log n)$ time call to Random-Select, for a total time bound of $O(n)$. The following lemma proves the correctness of this procedure. The lemma and its proof is given in [39].

Lemma 3.3.2

If an edge is chosen as described above, then Pr[(u,v)is chosen] is proportional to W(u,v).

Proof: [39].

3.3.4.2 Contracting an edge

Having shown how to choose an edge, next is the implementation of contraction. Given W and D, which represent a graph G, to update W and D to reflect the contraction of a particular edge (u, v). Call the new graph G' and compute its representation via the algorithm in Procedure 3.3.2.

Procedure (3.3.2) Contract an edge

```
        let D(u) <-- D(u) + D(v) - 2w(u, v)
        let D(v) <-- 0
        let W(u, v) <-- W(u, v) <--  0
For each vertex w except u v
        let W(u, v) <-- W(u, v)+W(u, w)
        let W(w, u) <-- W(w, u)+W(w, v)
        let W(v, w) <-- W(w, u) <-- 0
```

This algorithm moves all edges incident on v to u. The algorithm replaces row u with the sum of row u and row v, and replaces column u with the sum of column v. It then clears row v and column v. W and D now represent G', since any edge that was incident to u or v is now incident to u and any two edges of the form (u, w) and (v, w) for some w have had there weights added.

Furthermore, the only vertices whose total weighted degrees have changed are u and v, and D(u) and D(v) are updated accordingly. This procedure can be implemented in O(n) time. Summarizing, an edge can be chosen and contracted in O(n) time. This yields the corollary (3.3.1) [39].

Corollary: (3.3.1)

The Contraction Algorism can be implemented to run in $O(n^2)$ time .

Proof: [39].

For the rest of this section we will use the contraction algorithm as a subroutine, that accepts a weighted graph G and parameter k and in $O(n)$ time , returns a contraction of G to k vertices. With probability at least $\binom{k}{2}/\binom{n}{2}$ (Corollary3.3.1), a particular cut of the original graph will be preserved in the contracted graph.

3.3.5 The recursive contraction algorithm

The Contraction Algorithm can be used by itself as an algorithm for finding minimum cuts. The contraction Algorithm has an $\Omega(m)$ probability of success. Therefore repeating the contraction algorithm $(cn^2 \ln n)$ times, and out put the smallest cut produced by any runs of the contraction algorithm. The only way this procedure can fail to find the minimum cut is if all $(cn^2 \ln n)$ runs of the contraction algorithm fail to find the minimum cut, but we can upper bond the probability that this occurs by:

$$\left(1-\frac{1}{n}\right)^{cn^2 \ln n} \leq \exp(c \ln n) \leq n^c$$

Thus, the minimum cut will be found with high probability. However, the resulting sequential running time of $\overline{O}(n^4)$ is excessive. Lemma (3.3.3) and its proof is given in [39].

Lemma (3.3.3)

Algorithm Recursive Contract runs in $O(n^2 \log n)$ time.

Proof: [39].

3.4 The Markov chains partitioning approach

3.4.1 General

The partitioning technique, in this section, has been developed by [32], for the purpuse of the decomposed state estimator. The technique is based on markov chains as explained in Section (3.4.2). It partitions a spanning tree into k sub-spanning trees and finds all optimal minimum cuts. Partitioning a spanning tree guarantees the observability of all the sub-networks.

3.4.2 Markov chain model

In decision-making process, decisions are taken based upon past experiences. Not all decisions however prove to be correct, often because of certain inconsistencies or uncertainty associated with the phenomena upon which the decision was based.

Markov Chain theory has been used in hypertext, by associating a certain probability value to each hypertext link. Such an application helps the designer to decide how nodes are to be placed, and to make informed hardware configuration decisions based on the projected performance of the resulting system.

A Markov Chain can be described as a set of states, and transitions between those states that occur with a given probability. It is possible that in a stochastic process, the future (or next state) of the system depends only on the present state, and is independent of past events. This is called the Markovian property. A stochastic process is claimed to have the Markovian property if it satisfies the following condition:

$$\forall i, j.\ p_{ij} \geq 0; \qquad (3.4.1)$$

$$\forall i, \sum_{j=0}^{N} p_{ij} = 1; \qquad (3.4.2)$$

A typical N-states Markov chain and its N-step transition probabilities are represented in a matrix. The matrix is called the transition matrix.

state	0	1	2	3	…	…	…	N
0	p_{00}	p_{01}	p_{02}	p_{03}	…	…	…	p_{0N}
1	p_{10}	…	…	…	…	…	…	…
2	p_{20}	…	…	…	…	…	…	…
3	p_{30}	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
N	p_{N0}	…	…	…	…	…	…	p_{NN}

Table 3.4.1 The n -step transition matrix

3.4.3 Partitioning algorithm

The following summarizes the steps that are required to obtain all optimal possibilities of spanning tree by using Markov chain model. The technique guarantees minimum number of cuts to obtain the required of observable sub-networks.

1. Obtain a spanning tree of an observable PSSE network.
2. For each node in the spanning tree, count the number of branches that are connected directly to this node.
3. Assign to each branch a probability of one over the number of branches found in step 2 (i.e. assign equal branch probabilities such that their summation is equal to one).
4. Establish the transition matrix T, whose dimension is equal to the number of nodes in the system, in which each entry p_{ij} represents the probability value that has been assigned in step 3.

5. For each row i of the matrix T, search for a probability p_{ij} equal to one. The probability equal to one implies that node i in the ith row is connected only to node j in the jth column. Store node i as a child of node j, and delete row i and column i from the matrix T.

6. Redistribute the probabilities related to node j (which is the parent of the deleted child i).

7. Repeat steps 5 and 6 until the dimension of the matrix T reduces to one. A family tree (parent-child relationship) is established with actual connection between the nodes.

Once the family tree is established, it is easier to check all possible optimal cuts amongst the family tree. All optimal cuts take place between a parent and one of its children. In this case the selected child with its offspring are assigned to one cluster (or clusters).

3.4.4 Comparative performance

The performance of this technique has been evaluated by using different IEEE standard networks:24-bus, 30-bus, 118-bus and larger networks of 707-bus and 1084-bus. The 14-bus was used as an example.

The transition matrix T of the IEEE-14 network, according to the proposed algorithm will then be as in Table 3.4.2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.0	0.5	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.5	0.0	.05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.33	0.0	0.0	0.0	0.33	0.0	0.33	0.0	0.0	0.0	0.0	0.0
5	0.5	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.25	0.0	0.0	0.0	0.0	0.25	0.0	0.25	0.25	0.0
7	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.33	0.0	0.0	0.33
10	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

Table 3.4.2 The transition matrix of the IEEE-14 network

The non-zero entries represent the connection between the nodes and the summation of the assigned probabilities of any row equals one.

In the above example, node 3 is connected to node 2 and 4, and in the assigned probabilities to each branch is equal to one half. The next step is to search the transition matrix T for those entries that have a probabilities value of one, i.e. looking for the youngest child in the family of this system; or according to Markov Chain terminology, we are looking for transition probabilities equal to one. The out come of the first search iteration is the following children and their respective parents in Table 3.4.3.

Parent	Attached Children
7	8
6	10, 12, 13
9	11, 14

Table 3.4.3 First iteration Parent-Child Relationship

The new transition matrix after removing the corresponding rows and columns of the attached children (after the first iteration) and updating the related rows, we have a matrix as shown in Table 3.4.4.

	1	2	3	4	5	6	7	9
1	0.0	0.5	0.0	0.0	0.5	0.0	0.0	0.0
2	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0
3	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
4	0.0	0.0	0.3	0.0	0.0	0.0	0.3	0.3
5	0.5	0.0	0.0	0.0	0.5	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
7	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

Table 3.4.4 Updated transition matrix

The summation of the new probabilities in any row in the modified transition matrix remains one, and confirms that the modified matrix T still has the Markovian property explained in Section 3.4.3. The final result will converge after the fifth iteration. The family tree is shown in Figure 3.4.1.

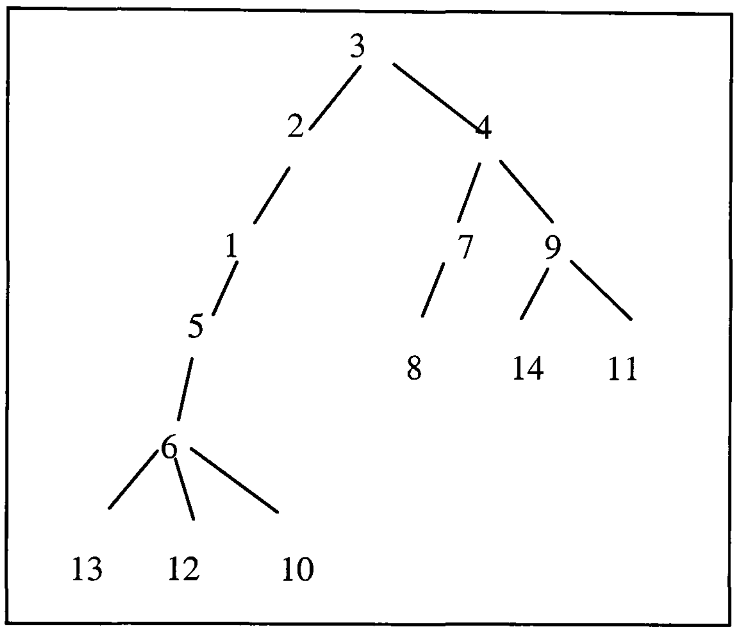


Figure 3.4.1 Family tree of spanning tree of IEEE-14 network

As represented in this family tree, the youngest members are calculated first, and then proceed upwards towards the eldest member of the tree.

In terms of partitioning, this family tree provides all optimal partitioning possibilities of the spanning trees. In case of large networks, the computer solution will automatically present all possible solutions.

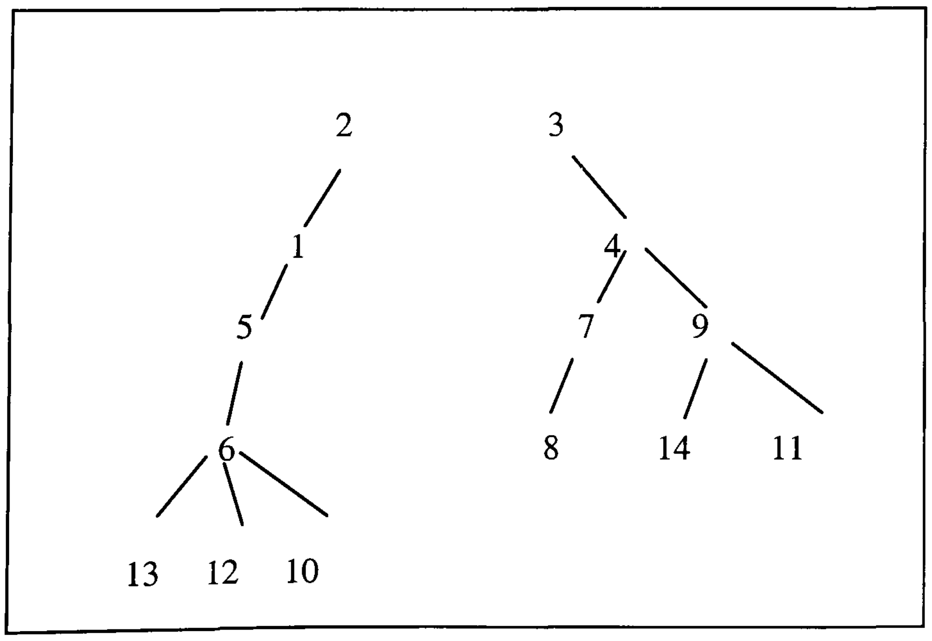


Figure 3.4.2. Two equal clusters of spanning tree of IEEE 14-bus network

If the spanning tree needs to be partitioned into two equivalent clusters, the only solution would cut child 2 (and its offspring) from its parent 3. Child 2 with its offspring form a collection of 7 family members, and the rest with parent 3 form another cluster. The two partitioned clusters, therefore, are shown in Figure 3.4.2.

If the spanning tree is to be partitioned into two clusters of unequal sizes with added constraint that the minimum number of nodes in any of the two clusters must be at least six. The family would provide three possible solutions. The optimal has already been shown above. The others are:

- By cutting the branch between child 1 and its parent 2, the clusters are shown in figure 3.4.3.
- Or, by cutting the branch between child 4 and its parent 3, the clusters are shown in Figure 3.4.4.

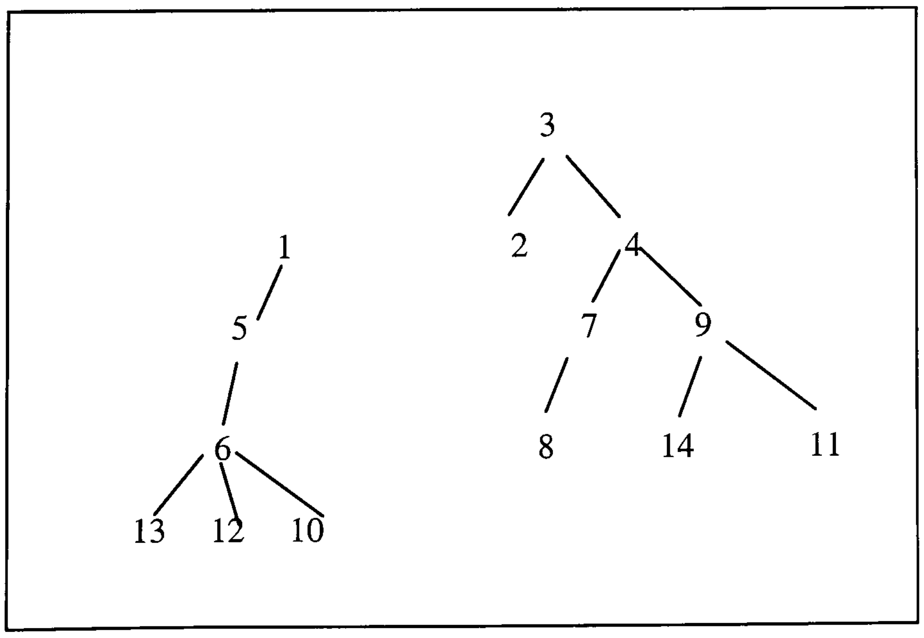


Figure 3.4.3 One option of two unequal clusters of spanning tree

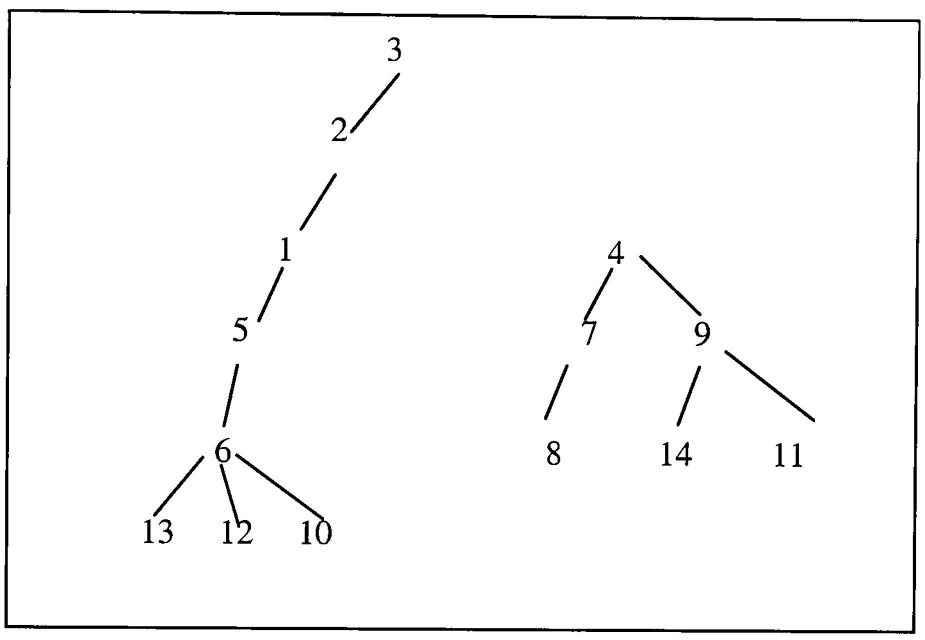


Figure 3.4.4 Another option of two unequal clusters of the spanning tree of the IEEE 14 –bus.

3.5 Advantages and disadvantages of these approaches

Knowing the DSE objective and constraints as introduced in Section 2.4, the three approaches can be developed to satisfy the DSE requirements. In the present case they share several points against the DSE constraints, such as:

1. None of the three approaches take into account the DSE constraints and they do not classify the network nodes into internal nodes and boundary nodes.
2. Depending on which spanning tree is used, the partitioning which results from the first and the third partitioning approaches may give good balanced results, but it may not.
3. Obtaining the optimal cut edges from a spanning tree does not guarantee obtaining the minimum number of cut-edges from the network.

4. The three methods are heuristic in their approach. They do not discuss the network-partitioning problem itself or its NP-hard nature, nor do they provide any mathematical model to simplify the problem.

Chapter 4

Spanning tree partitioning technique

4.1 General

Graph theory is a branch of mathematics that has wide practical applications. Many problems arising in such diverse fields as psychology, chemistry, electrical network, transportation planning, management, marketing, and education can be posed as problems in graph theory [5]. A graph of a power system network has n nodes connected by m edges representing the actual network.

Partitioning a network has objectives and constraints. These objectives and the constraints are directly related to the addressing problem. Optimal partitioning of a network is relative to the partitioning objectives and constraints, i.e. if these objectives are met, and these constraints are satisfied, then the partitioning solution is classified as optimal, and if these objectives are met and the constraints are almost satisfied, then the partitioning solution is classified as near optimal. If the solution does not met the objectives the partitioning solution is rejected.

The objective of partitioning a power system network for the purposes of the decomposed state estimation problem has been discussed in Section (2.4). The idea is to partition the network into k non-overlapping subsystems subject to constraints. The constraints are such that (i) the internal nodes of the k subsystems are balanced and (ii) the number of total boundary nodes is balanced with the number of internal nodes of the i^{th} subsystem.

The problem associated with partitioning networks is a compound problem. Networks have different sizes together with different types of connections they can take. In addition, the nodes in a very large network may have different type of distribution. They may be concentrated in one or more clusters, or they may be distributed equally over the very large network. The partitioning problem has proven [13, 25] to be NP-Hard problem; thus most existing partitioning algorithms are heuristically based.

Searching graph theory [18, 21, 68] for relationships that describe a network of any size with any type of connection, very few such relationships appear to exist. Examples are: the network degree relationship, i.e. $D = 2m$, which relates the sum of the nodes degrees to twice the number of edges; and the cycle relationship i.e. $c = m - n + 1$. Neither of these direct relationships are particularly helpful in providing information for partitioning the network. They do not describe the network connection in an appropriate way.

Alternatively, it may be useful to examine indirect relationships that can describe the network connection. An example of this is the spanning tree relationship. A spanning tree of a network is a connected sub-graph of the network, having n nodes (as has the network) and n-1 edges. A spanning tree can be obtained by eliminating exactly one edge from each cycle in the network [64]. Thus, a spanning tree of a given network may be used to partition the network.

Partitioning a spanning tree into k partitions is obtained by cutting exactly (k-1) edges from the spanning tree. The partitioning technique introduced in this chapter is based on partitioning the spanning-tree optimally into k sub-spanning trees. The technique first defines the number of edges of each sub-spanning tree, and then finds the branches of each sub-spanning tree such that the number of edges in those branches equals the number of edges of the sub-spanning tree. The DSE constraints are to have balance between the internal nodes and the boundary nodes. These constraints are achieved by balancing the edges of the k sub-spanning trees.

Balanced k-subsystems that satisfy the requirements of the DSE may be obtained by partitioning the given network for different values of k. Thus the partitioning technique starts from k=2, and then for k=3, and so on. It terminates when k does not give balanced results.

The chapter is organized as follows: A power system network can be represented by a direct graph or undirected graph. The direct representation of a network is presented in Section 4.2. The definition and the properties of the spanning trees are presented in Section 4.3. A method for obtaining a spanning tree, using row reduction, is described in Section 4.3.1. The resultant matrix, described in Section 4.3.2, is a descendent star matrix representing the spanning tree. The partitioning technique utilizes the descendent property as the starting point for partitioning. Cutting one edge from the spanning tree partitions the spanning tree into two separate parts, but it may not partition the network into two separate subsystems. Thus, the cut concept is introduced in Section 4.4. Obtaining the cut-edges between the k subsystems is described in Section 4.4.5 as a matrix sum operation. Balancing the edges of the k sub-spanning trees is derived in Section 4.6. An overview of the spanning tree partitioning technique is given in Section 4.7. The technique is based on finding the branches of each sub-spanning tree. Thus, the spanning tree nodes are classified in Section 4.7.1 into three types. The definition and the property of a branch are discussed in Section 4.7.2. The steps of finding the sub-spanning tree are described in Section 4.7.3. An example and the simulation results are given is Sections 4.9 and 4.10. The performance of the technique and its disadvantages are described in Sections 4.11 and 4.12 respectively. Some conclusions are drawn in Section 4.13. Definitions of terms associated with graph theory are provided in Appendix A.

4.2 Graph representation of network

Directed graphs are adequate for representing many situations [70], such as traffic flow networks, where an edge may represent a street and the direction to indicate the permissible direction of traffic flow. Also, a power system network, consisting of power stations and transmissions lines, requires a directed graph, where each node is represented by a vertex in the graph, and each line is represented by an edge. The current flow direction is represented by an edge direction.

In this chapter, a directed graph is used to represent a power system. An example of a directed network is shown in Figure 4.1:

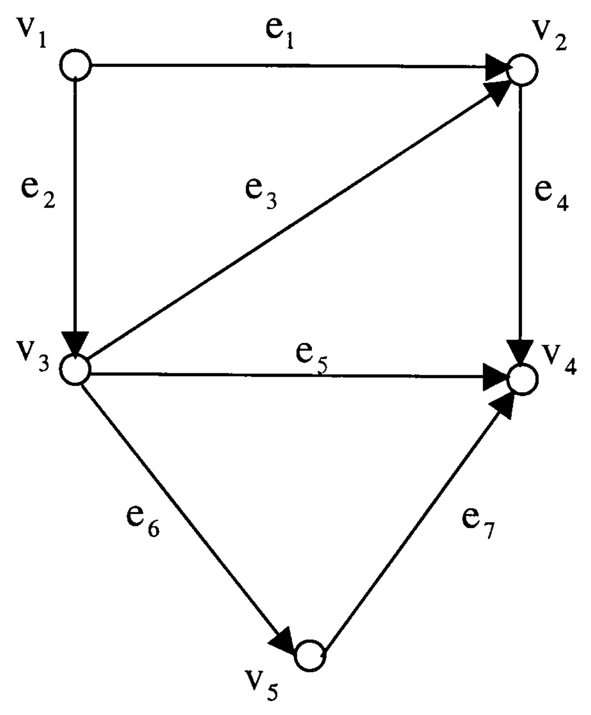


Figure 4.1 A directed graph of a network G_N

4.2.1 The incidence matrix

Consider a network represented by a direct graph G_N with n nodes and m edges and having no self-loops. The all-nodes incidence matrix $A = \lfloor a_{ij} \rfloor$ of G_N has m rows, one for each edge, and n columns, one for each node. The element a_{ij} of A is defined as follows:

$$a_{ij} = \begin{cases} 1, & \text{if the } j^{th} \text{ edge is from the } i^{th} \text{ node} \\ -1, & \text{if the } j^{th} \text{ edge is to the } i^{th} \text{ node} \\ 0, & \text{if there is no edge between the } i^{th} \text{ node and the } j^{th} \text{ node} \end{cases} \quad (4.1)$$

For example the incidence matrix of the network in Figure 4.2 is:

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Figure 4.2 The incidence matrix of the network of Figure 4.1

It is noticed from the incidence matrix definition that it has common properties [21, 64, 70], namely,

Properties of the incidence matrix

- Each column of A contains exactly the degree of the j^{th} node.
- Each row contains exactly two non-zero entries 1 and -1.
- Any row of A can be obtained from the remaining n-1 rows. Thus any n-1 rows of A contain all information about A. In other words, the rows of A are linearly dependent, and the rank of A $\leq n-1$.
- For any connected graph, the rank of A is n-1.
- The determinant of any incidence matrix of a tree is equal to +1 or -1.
- If a network consists of k disconnected components, then the rank of A is n-k.

4.3 Spanning trees

A spanning tree, T, of a network G_N is, by definition, connected and it includes all the network nodes [64, 70]. Figure 4.3a shows a spanning tree of figure 4.1 network. A co-spanning tree T^* of a spanning tree T of a graph G_N is a sub-graph of G_N having exactly those edges of G_N that are not in the spanning tree T.

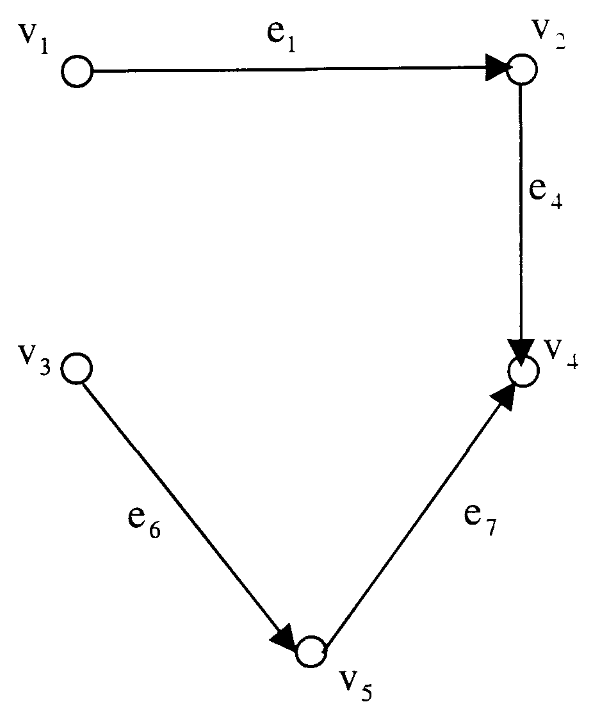


Figure 4.3a A spanning tree of Figure 4.1

Figure 4.3b shows a co-spanning tree T^* of figure 4.1. Each spanning tree T uniquely determines its co-spanning tree T^*.

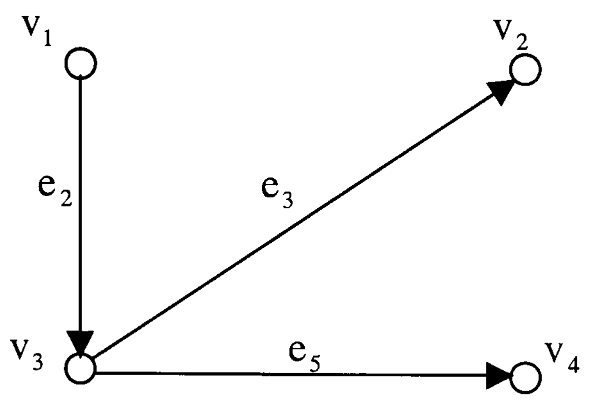


Figure 4.3b A co-spanning tree of Figure 4.1

The following gives full characterization of any tree [70]:

Trees Properties

The following statements are equivalent for a tree, T, with n nodes and m_t edges:

1. T is a tree.
2. There exists exactly one path between any two nodes of T.
3. T is connected and $m_t = n - 1$. (4.2)
4. T is acyclic and $m_t = n - 1$.
5. T is acyclic, and if any two nonadjacent nodes of G are connected by an edge, then the resulting graph has exactly one circuit.

A consequence of the tree properties is the following [70] spanning tree properties.

Spanning trees property (4.1)

Consider a sub-graph T of n nodes of graph G. Let T have n nodes and m_t edges, then the following statements are equivalent:

1. T is a spanning tree of G.

2. There exists exactly one path between any two nodes of T.
3. T is connected and $m_t = n - 1$.
4. T is acyclic and $m_t = n - 1$.
5. T is acyclic, and , if any two nonadjacent nodes of T are connected by an edge, the resulting graph has exactly one circuit.

4.3.1 The row reduction method

A network may be represented by A, the associated incidence matrix. The incidence matrix has m rows of edges and n columns of nodes. It is interesting to consider the operation of row reduction on matrix A. Row reduction eliminates a certain type of edge from A, specifically one edge from each cycle in the network [64]. When row reduction is performed on A, each operation has a meaning in the network, i.e. each operation moves or eliminates one or more edges from the network. Since there are m-n+1 cycles in the network, then there will be m-n+1 edges eliminated from the network in O (m-n+1) operations. Once row reduction is complete, the row reduced form of A is termed the spanning tree matrix, represented by U. The spanning tree matrix has exactly n-1 rows of edges, and it has n columns, each column representing a node.

Example (4.1) Example 4.1 illustrates the row reduction method.
The network in Figure 4.4 has:
n = 4 nodes and m = 5 edges.

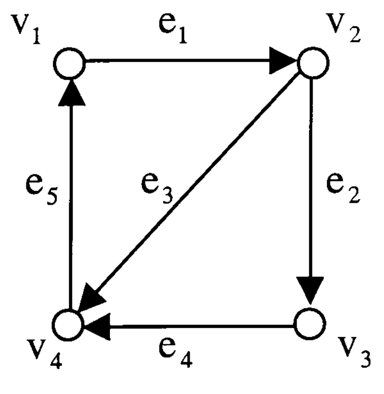


Figure 4.4 A network used to

illustrate on the row reduction

The incidence matrix of Figure 4.4 is shown in Figure 4.5.

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.5 The incidence matrix of Figure 4.4

Figures 4.6a and 4.6b show the first row reduction operation, in which, when row 5 is reduced by row 1, e_5 will move from e (v_4 , v_1) to e (v_4 , v_2).

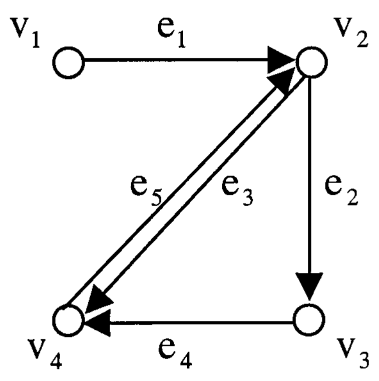


Figure 4.6a The first row reduction operation

$$A_1 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & -1 & 0 & 1 \end{bmatrix};$$

Figure 4.6b The resultant matrix of the first row reduction operation

Figure 4.7a and 4.7b show the second row reduction operation, in which, column 2 (or v_2) has 3 edges, e_2, e_3 and e_5. The second edge, e_2, has the leading one, so it will be the reference for the other two edges. The ones of e_5

and e_3 will be eliminated by the one of e_2. This results in moving e_3 to be from v_3 to node v_4, and moving e_5 to be from node v_4 to node v_3.

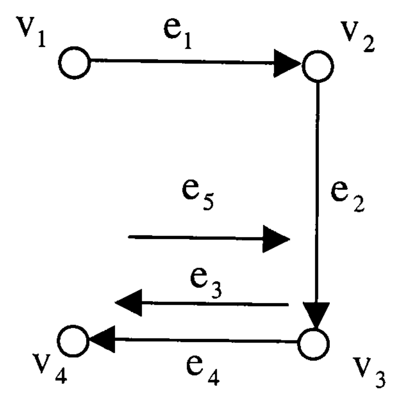


Figure 4.7a The second row reduction operation

$$A_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix};$$

Figure 4.7b The resultant matrix
of the second row reduction operation

Figure 4.8a and Figure 4.8b show the result of the third row reduction operation. Between node 3 and node 4, there are three edges, e_3, e_4 and e_5, and those edges are in opposite directions. From those edges e_3 has the leading one at v_3. Then the ones at e_4 and at e_5 will be eliminated. Therefore, eliminating the edges e_4 and e_5 from the matrix and from the graph. The result is a spanning tree of the network as shown in Figure 4.8b. The row reduction method is described in Flowchart 4.1.

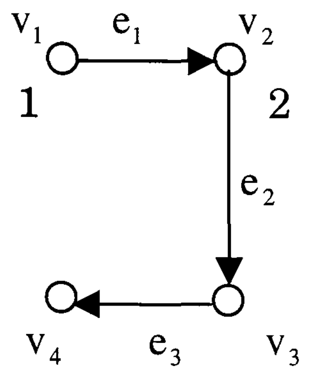
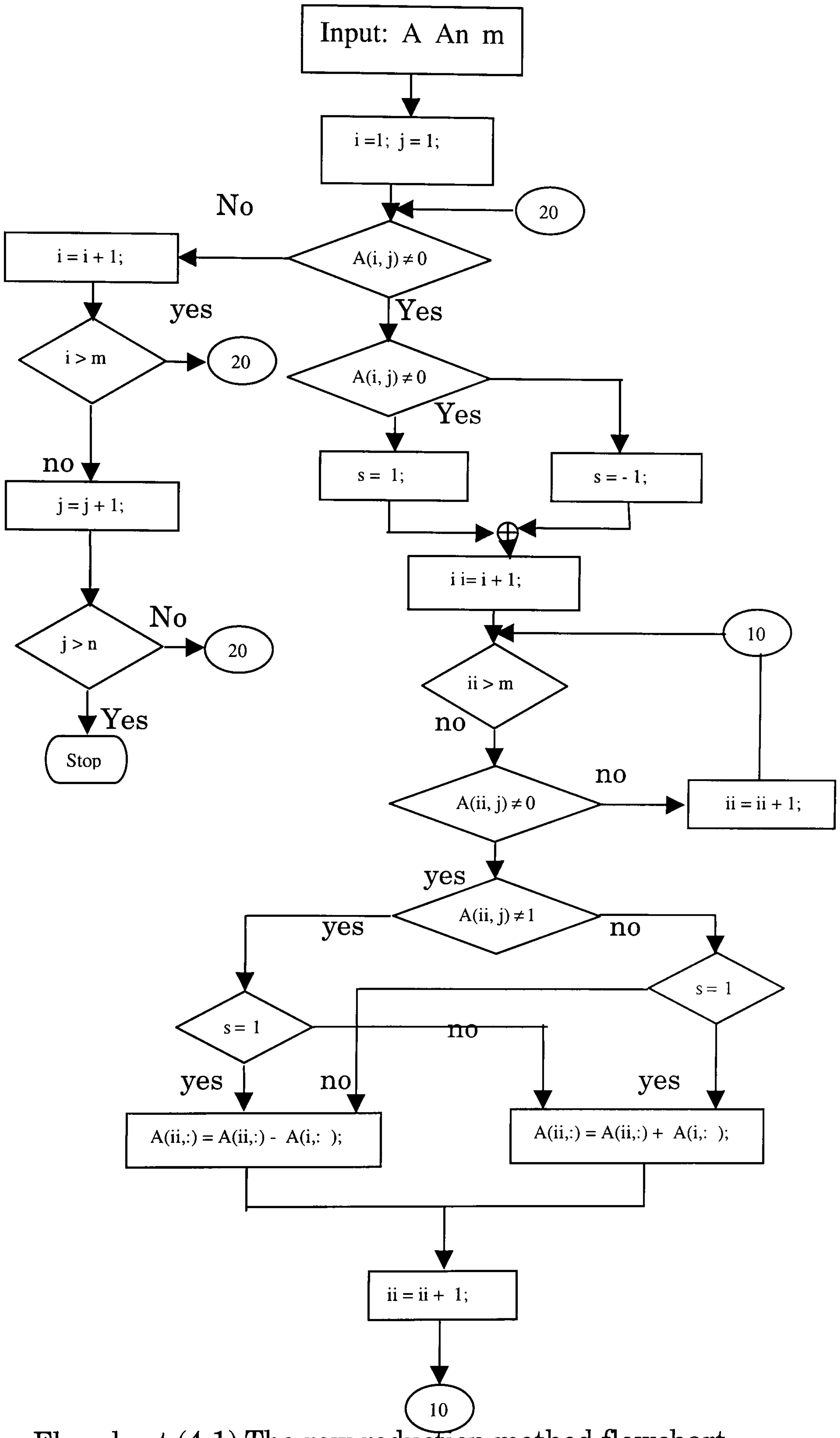


Figure 4.8a A spanning tree of Figure 4.4

$$U = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix};$$

Figure 4.8b A spanning tree matrix of Figure 4.4



Flowchart (4.1) The row reduction method flowchart

4.3.2 The spanning tree matrix

Provided that the edge numbered one leaves the node numbered one (to ensure that the element in the first row and the first column is always non-zero), the resultant spanning tree matrix U has a descending staircase shape. The spanning tree matrix U has exactly n columns of nodes and n –1 rows of edges. Each row has two non-zero elements representing an edge in the spanning tree. Each column has one or more non-zero elements representing a node in the spanning tree. The columns with one nonzero element represent a single node, i.e. a node with only one edge. The number of the non-zero elements in each column gives the degree of that node in the spanning tree. The descending staircase shape of matrix U has been utilized to partition the spanning tree, as explained in Sections 4.6 to 4.8.

4.3.3 Testing connectivity

The spanning tree is anther way of testing the connectivity of the network [70]. Property (4.2) proves that a network is connected if it has a spanning tree.

Spanning trees property (4.2)

A graph G is connected if and only if it has a spanning tree.

4.4 The cut concept

When a network is partitioned into k disjoint subsystems, one or more edges from the network must be cut. Those edges are termed cut-edges. In this section, the cut concept is introduced.

4.4.1 Cut-sets

A cut-set E_b of a connected graph is a minimal set of edges of G such that its removal from G disconnects G [70]; that is the graph $G - E_b$ is disconnected. For example, consider the subset $E_{b\text{-}1} = \{e_2, e_6, e_9\}$ of edges of the graph G in Figure 4.9a. The removal of $E_{b\text{-}1}$ from G results in the graph

$G_1 = G - E_{b-1}$ of Figure 4.9b, whereupon G_1 is disconnected. Furthermore, the removal of any proper subset of E_{b-1} cannot disconnect G. Thus E_{b-1} is a cut-set of G.

Consider next the set $E_{b-2} = \{e_1, e_4, e_6, e_8\}$. The graph $G_2 = G - E_{b-2}$ shown in Figure 4.9c is disconnected. However, the set $E'_{b-2} = \{e_1, e_4, e_8\}$, which is a proper subset of E_{b-2}, also disconnects G. The graph $G_3 = G - E'_{b-2}$ is shown in Figure 4.9d. Thus E_{b-2} is not a cut-set of G.

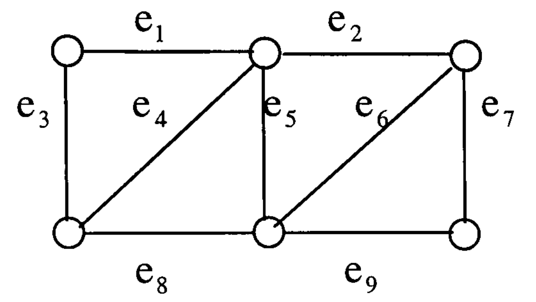


Figure 4.9a A graph G, to illustrate the cut-set concept

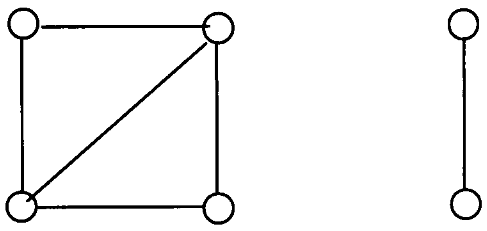


Figure 4.9b $G_1 = G - E_{b-1}$ $E_{b-1} = \{e_2, e_6, e_9\}$

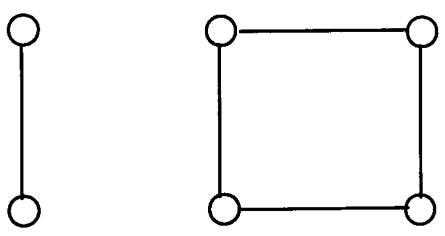


Figure 4.9c $G_2 = G - E_{b-2}$ $E_{b-2} = \{e_1, e_4, e_6, e_8\}$

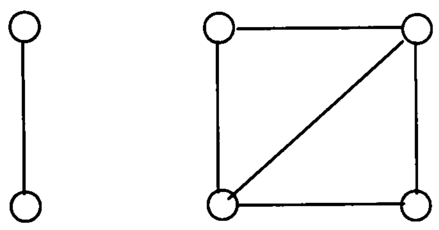


Figure 4.9d $G_3 = G - E'_{b-2}$ $E'_{b-2} = \{e_1, e_4, e_8\}$

4.4.2 Cuts

Consider a network $G_N = (V, E)$ with node set V. Let V_1 and V_2 be two mutually disjoint subsets of V such that $V = V_1 \oplus V_2$; that is, V_1 and V_2 have no common nodes and together contain all the nodes of V. Then **the set E_b of all those edges of** G_N having one end node in V_1, and the other end in V_2, **is called a cut of** G_N [70]. This is usually denoted by $<V_1, V_2>$ [70]. For example, for the network shown in Figure 4.10a, if $V_1 = \{v_1, v_2, v_3, v_4\}$ and $V_2 = \{v_5, v_6, v_7\}$, then the cut $<V_1, V_2>$ of G_N is equal to the set $\{e_6, e_7, e_8\}$ of edges, as shown in Figure 4.10b.

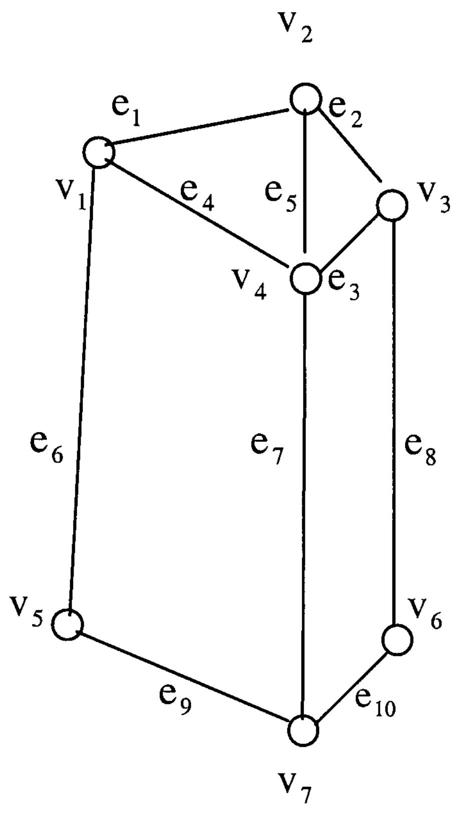


Figure 4.10a A network to illustrate the cut concept

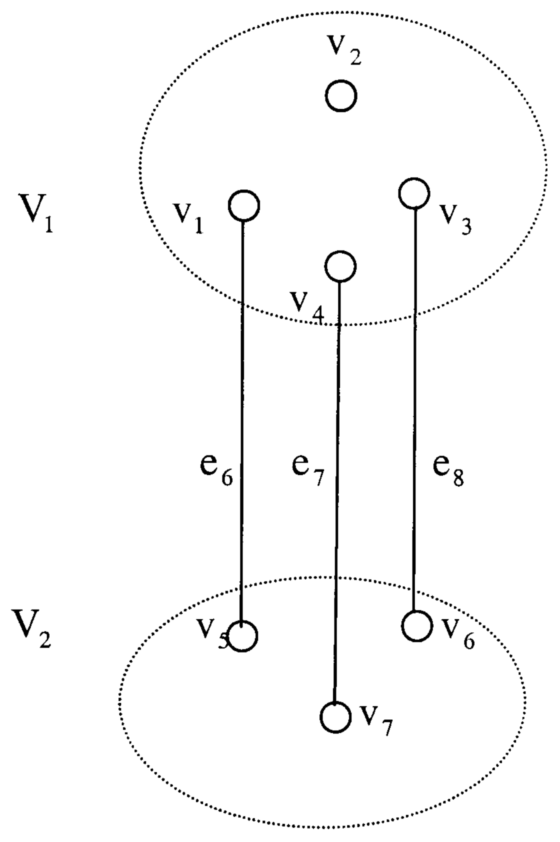


Figure 4.10b The cut

The cut $<V_1, V_2>$ of G_N is the minimal set of edges of G_N whose removal disconnects G_N into two (induced) sub-graphs G_1 and G_2, on the node sets V_1 and V_2. Thus, if the number of edges between the two (induced) sub-graphs, i.e. the cut $<V_1, V_2>$, is minimum, then the cut $<V_1, V_2>$ is E_b, by definition, the cut-set of G_N.

Partitioning a spanning tree

Since every spanning tree T of G is acyclic, then every sub-graph of T is also acyclic [70]. A sub-graph of a spanning tree is termed **a sub-spanning tree**.

Spanning trees property (4.3)

A sub-graph T of a connected graph G is a sub-graph of some spanning tree of G if and only if T is acyclic .

If k-1 edges are cut from a spanning tree of a network then the spanning tree is partitioned into k sub-spanning trees. The nodes of each sub-spanning tree are the nodes of one subsystem. The k-1 cut edges are used to find the remaining cut-edges between the k subsystems.

The partitioning technique partitions the spanning tree into k sub-spanning trees by cutting k-1 edges. Each sub-spanning tree, T, has m_{it} edges and $(m_{it} + 1)$ nodes. Determining the edges of each sub-spanning tree is given in Section 4.4.

Spanning trees property (4.4)

1. A cut $<V_1, V_2>$ of a connected graph G is a cut-set of G if the two induced sub-graphs of G on node set V_1 and V_2 are disconnected.
2. If E_b is a cut-set of a connected graph G, and V_1 and V_2 are the node sets of two induced sub-graphs of $G - E_b$, then $E_b = <V_1, V_2>$.

Any cut $<V_1, V_2>$ in a connected graph G contains a cut-set of G, since the removal of $<V_1, V_2>$ from G disconnects G. In fact, a cut in a graph G is the union of some edge-disjoint cut-sets of G. This is stated in the following property.

Spanning trees property (4.5)

A cut in a connected graph G is a cut-set or union of edge-disjoint cut-sets of G.

4.4.3 Fundamental cut-sets

This section shows how a spanning tree can be used to define a set of fundamental cut-sets. Consider a spanning tree T of a connected graph G. Let b be an edge of T. Removal of the edge b from T disconnects T into exactly two components T_1 and T_2. Note that T_1 and T_2 are trees of G. Let V_1 and V_2 be the nodes of T_1 and T_2 respectively. Then V_1 and V_2 contain all nodes of G.

Let G_1 and G_2 be, respectively, the induced sub-graphs of G on the nodes set V_1 and V_2. It can be seen that T_1 and T_2 are, respectively, spanning trees of G_1 and G_2. Hence, by property (4.2) G_1 and G_2 are connected. This in turn, proves property (4.4) that the cut $< V_1, V_2 >$ is a cut-set of G [70]. This cut-set is known as the fundamental cut-set of G with respect to the edge b of the spanning tree T of G. The set of all the $n - 1$ fundamental cut-sets with respect to the $n - 1$ edges of a spanning tree T of a connected graph G is known as the fundamental set of cut-sets of G with respect to the spanning tree T.

Note that the cut-set $< V_1, V_2 >$ contains exactly one edge, namely, the edge b of T. All other edges of $< V_1, V_2 >$ are belong to G_T. This follows from the fact that $< V_1, V_2 >$ does not contain any edge of T_1 and T_2.

Further, the edge b is not present in any other fundamental cutest with respect to T. Because of these properties, the edge set of the fundamental cut-set can be expressed as the **ring sum** of the edge sets of some or all of the remaining fundamental cut-sets. A graph G and a set of fundamental cut-sets of G are shown in Figures 4.11a to 4.11d.

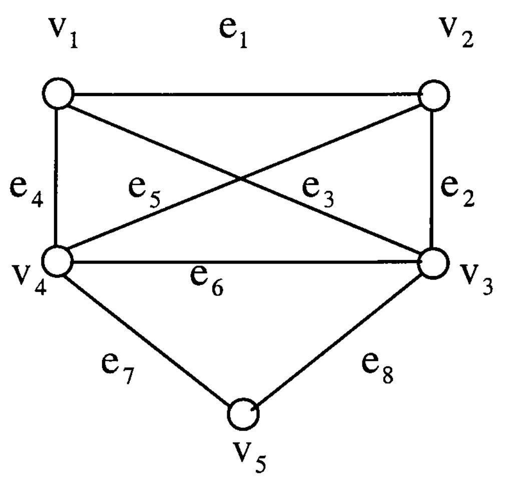


Figure 4.11a A network to illustrate the fundamental cut-set

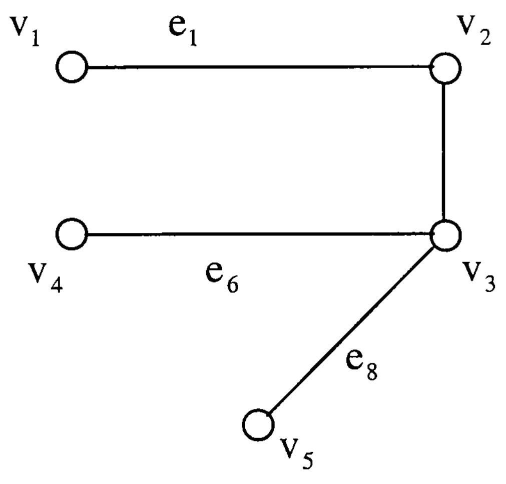


Figure 4.11b A spanning tree

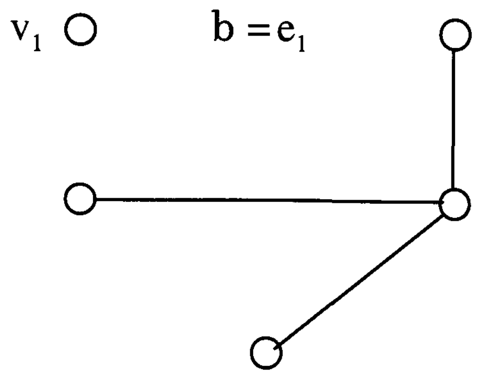


Figure 4.11c The edge b is removed from the spanning tree

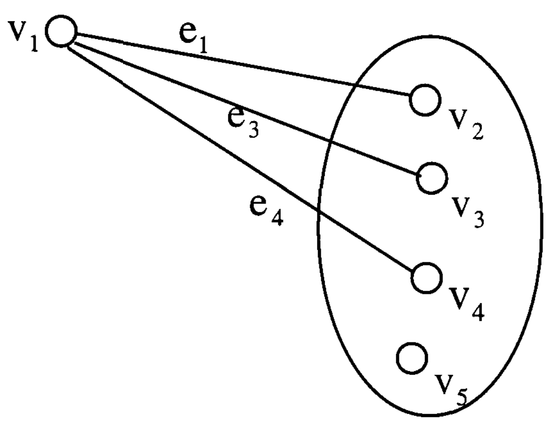


Figure 4.11d The cut-edges between the two cuts

4.4.4 The cut matrix

Define a cut matrix Q of m columns and c rows, where m is number of edges of a graph G and c is the number of cutting edges in G. For example, let A^T given in Figure 4.12a be the transpose of the incidence matrix of the graph given in Figure 4.12, and let the graph has three different cuts as shown in Figures 4.12b, 4.12c and 4.12d.

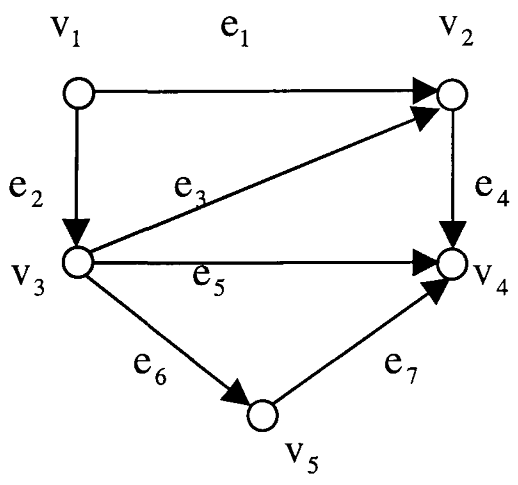


Figure 4.12 A graph G=(5, 7)

$$A^T = \left[\begin{array}{cccccccc} \text{nodes/edges} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ v_1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ v_2 & -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & -1 & 1 & 0 & 1 & 1 & 0 \\ v_4 & 0 & 0 & 0 & -1 & -1 & 0 & -1 \\ v_5 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right];$$

Figure 4.12a A transpose of the incidence matrix of figure 4.12

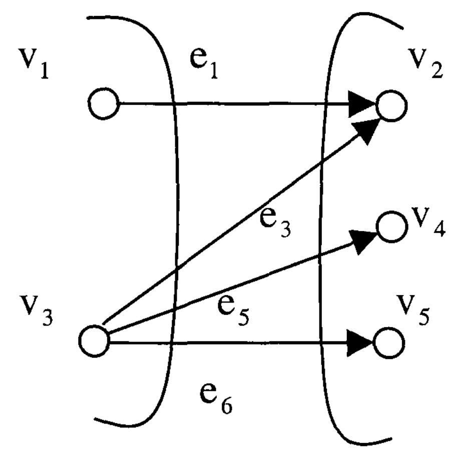


Figure 4.12b Cut-1 = { e_1 , e_7, e_5, e_6}

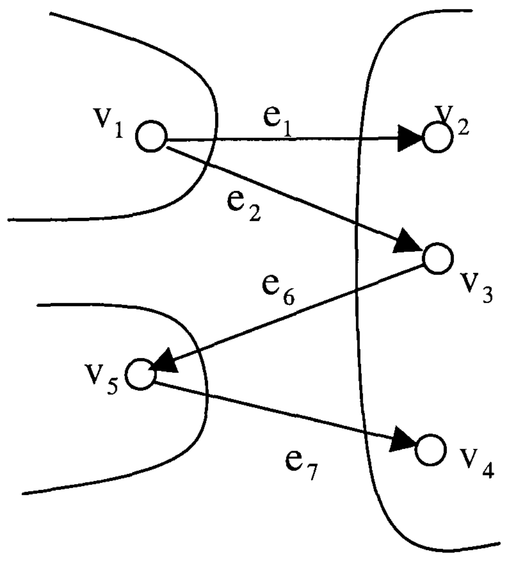


Figure 4.12c Cut-2 = { e_1, e_2 , e_6, e_7}

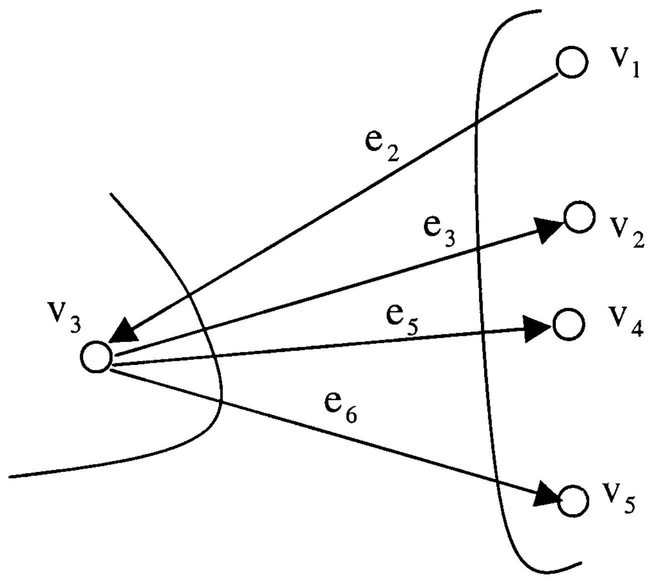


Figure 4.12d Cut-3 = { e_2, e_3, e_5, e_6}

Then, the cut matrix of the three cuts is

$$Q = \begin{bmatrix} \text{cut/edges} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ \text{cut1} & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ \text{cut2} & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ \text{cut3} & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 4.12f The cut matrix

Each cut (or row) in Q can be expressed as a linear combination of the rows (i.e. the nodes) of A^T which represent the nodes of the induced sub-graph, i.e. sum of the rows of nodes (in A^T) of one sub-graph = (-) sum of the rows of nodes (in A^T) of the other sub-graph [69].

Property of cut matrix

Each cut (or row) in the cut matrix Q can be expressed in **two ways**, as a linear combination of the nodes (i.e. the rows in A^T) of one sub-graph or as a linear combination of nodes of the other sub-graph with (-1) sign.

Example (4.2) Finding the cut-edges

Considering cut-1, this cut partitions G into two sub-graphs. The nodes of the sub-graphs are

$$V_1 = \{v_1, v_3\};$$

$$V_2 = \{v_2, v_4, v_5\}.$$

To find the cut-edges between the two sub-graphs, add the rows corresponding to the nodes of each sub-graph, i.e.

$$v_1 = \text{row1} = [1\ 1\ 0\ 0\ 0\ 0\ 0];$$

$$v_3 = \text{row3} = [0\ \text{-}1\ 1\ 0\ 1\ 1\ 0].$$

Adding the two rows gives cut1

$$\text{cut1} = v_1 + v_3 = [1\ 0\ 1\ 0\ 1\ 1\ 0].$$

Thus the cut edges between V_1 and V_2 are $\{e_1, e_3, e_5, v_6\}$ as shown in Figure 4.12b. Considering the nodes of the second sub-graph, i.e. $S_2 = \{v_2, v_4, v_5\}$, we get the same result.

$$v_2 = \text{row2} = [\text{-}1\ 0\ \text{-}1\ 1\ 0\ 0\ 0];$$

$$v_4 = \text{row4} = [0\ 0\ 0\ \text{-}1\ \text{-}1\ 0\ \text{-}1];$$

$$v_5 = \text{row5} = [0\ 0\ 0\ 0\ 0\ \text{-}1\ 1];$$

$$\text{cut1} = -(v_2 + v_4 + v_5) = [1\ 0\ 1\ 0\ 1\ 1\ 0].$$

4.4.5 The fundamental cut-sets matrix

Another important sub-matrix of Q is defined, namely Q_f. It is known that a spanning tree T defines a set of n-1 fundamental cut-sets, one fundamental cut-set for each branch of T. The sub-matrix of Q corresponding to these n-1 fundamental cut-sets is known as the fundamental cut-set matrix Q_f of G with respect to the spanning tree T. Q_f has dimensions (n-1) x m; the i^{th} row of Q_f is a linear combination of the nodes of the partitioned sub-network. In the matrix Q_f, an element +1 (-1) indicates branch direction from (to) this sub-network.

The i^{th} row of Q_f carries the information:

1- the number of cut-edges in the network if the i^{th} branch is cut from T.

2- the identification of all cutting edges in the network, from which the boundary nodes can be obtained.

Example (4.3):

The fundamental cut-set matrix Q_f of the graph of Figure 4.10 with respect to the spanning tree $T = \{e_1, e_2, e_6, e_7\}$ is given in Figure 4.13

$$Q_f = \left[\begin{array}{cccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ e_1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 \\ e_2 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ e_6 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ e_7 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]$$

Figure 4.13 The fundamental cut-matrix

It is clear that the rank of Q_f is n-1, the rank of Q [70]. Thus every cut-set (or cut vector) can be expressed as linear combination of the fundamental cut-set.

The spanning tree partitioning technique partitions the spanning tree into k sub-spanning trees by cutting (k-1) edges from T. It uses the sub-spanning tree edges to find the nodes of the subsystem. Each cut-edge determines one fundamental cut-set. This fundamental cut-set determines exactly all connections, which are minimum, between the i^{th} subsystem to all other subsystems. The boundary nodes are the end-nodes of the cut-edges.

4.5 Balancing the edges of the k sub-spanning trees

The goal of partitioning a spanning tree is to cut the minimum number of edges such that the total number of boundary nodes is minimum and to have balanced internal nodes for the k sub-spanning trees. Since T, the spanning tree has n nodes and exactly n − 1 edges, if T is partitioned into k sub-spanning trees, the minimum number of cutting edges is exactly k-1 edges. Let $m_{\text{sbtree-i}}$ and $n_{\text{sbtree-i}}$ be the number of edges and nodes, respectively, in the i^{th} sub-spanning tree. Then, the total number of nodes in the k sub-spanning trees is:

$$n = n_{\text{sbtree-1}} + n_{\text{sbtree-2}} + \cdots + n_{\text{sbtree-k}}; \qquad (4.3)$$

the number of edges of each sub-spanning tree is:

$$m_{\text{sbtree-i}} = n_{\text{sbtree-i}} - 1; \qquad (4.4)$$

and the total number of edges of the k sub-spanning trees is:

$$m_{\text{ktrees}} = m_{\text{sbtree-1}} + m_{\text{sbtree-2}} + \cdots + m_{\text{sbtree-k}} . \qquad (4.5)$$

Substituting equation (4.4) into equation (4.5)

$$m_{\text{ktrees}} = (n_{\text{sbtree-1}} - 1) + (n_{\text{sbtree-2}} - 1) + \cdots + (n_{\text{sbtree-k}} - 1);$$

It follows that

$$m_{\text{ktrees}} = n - k . \qquad (4.6)$$

Each sub-spanning tree belongs to a subsystem. It is required by the DSE to have balanced subsystems. Therefore, balancing the edges of the k sub-spanning trees will balance the nodes of the k subsystems.

In general, when k-1 edges are cut from a spanning tree, the resultant k sub-spanning trees might be balanced, and they might be unbalanced. Since a tree consists of nodes connected by edges with no loops, then the smallest possible tree has one node without an edge, i.e. when a spanning tree is partitioned into k sub-spanning trees, some of the sub-spanning trees might have only one node. This case is not desirable.

It is required by the DSE to balance the internal nodes in the k sub-spanning trees and find the minimum boundary nodes that is with respect to equation (4.6). Equation (4.6) can be rewritten as:

$$(n - k) = m_{\text{sbtree-1}} + m_{\text{sbtree-2}} + \cdots + m_{\text{sbtree-k}} \qquad (4.7)$$

Balancing the subsystems nodes is achieved by balancing the number of edges of the k sub-spanning trees. The algorithm of balancing the edges of the k sub-spanning trees is described below:

Since m_{ktrees} are the edges of the k parts, then, define m_p to be the edges of the one part.

$$m_p = \left\lfloor \frac{m_{\text{ktrees}}}{k} \right\rfloor = \left\lfloor \frac{n - k}{k} \right\rfloor; \qquad (4.8)$$

where $\lfloor \ \rfloor$ indicates rounding down to the nearest integer.

Let R be the remainder. Then

$$R = (n - k) - m_p; \tag{4.9}$$

and the range of R is:

$$0 \leq R \leq k - 1 \tag{4.10}$$

This range of R determines exactly the number of possible balanced solutions, by distributing R equally and increasingly, in turns.

If R=0, then the k sub-spanning trees are equal, i.e. there will be one solution.

$$m_{sbtree\text{-}i} = m_p \quad \text{for } i = 1, \ldots, k; \tag{4.11a}$$

and if R > 0 then there will be R possible balanced solutions, as follows:

$$m_{sbtree\text{-}i} = m_p \qquad \text{for } i = 1,\ldots, k\text{-}R$$

$$m_{sbtree\text{-}i} = m_p + 1 \qquad \text{for } i = k\text{-}R\text{+}1, \ldots, k. \tag{4.11b}$$

After determining the edges of the balanced k sub-spanning trees it is easy to determine the nodes of each sub-spanning tree.

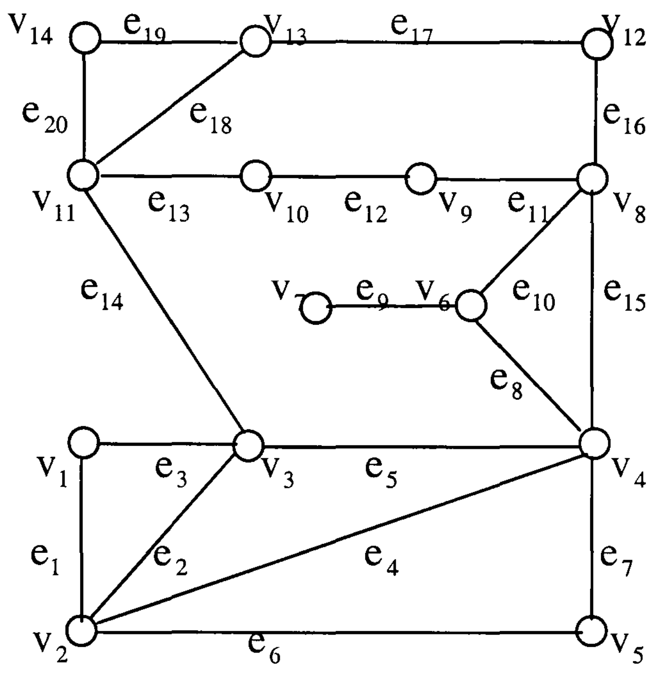
4.6 Classifying the spanning tree nodes

The partitioning technique classifies the nodes of the spanning tree, according to their degrees, into three types, as follows:

If the degree of the i^{th} node is one then the i^{th} node is termed **a bottom-node**. If the degree of the i^{th} node is two then the i^{th} node is termed **a branch-node.** If the degree of the i^{th} node is more than two then the i^{th} node is termed **a junction-node**. For example, Figure 4.14a is the IEEE 14-bus network, with a spanning tree shown in Figure 4.14b.

Figure 4.14a

The IEEE-14 network



In the spanning tree of Figure 4.14b, the nodes $\{v_2, v_5, v_7, v_{13}, v_{14}\}$ are bottom nodes, the nodes $\{v_1, v_6, v_8, v_9, v_{10}, v_{12}\}$ are branch-nodes and the nodes $\{v_3, v_4, v_{11}\}$ are junction-nodes.

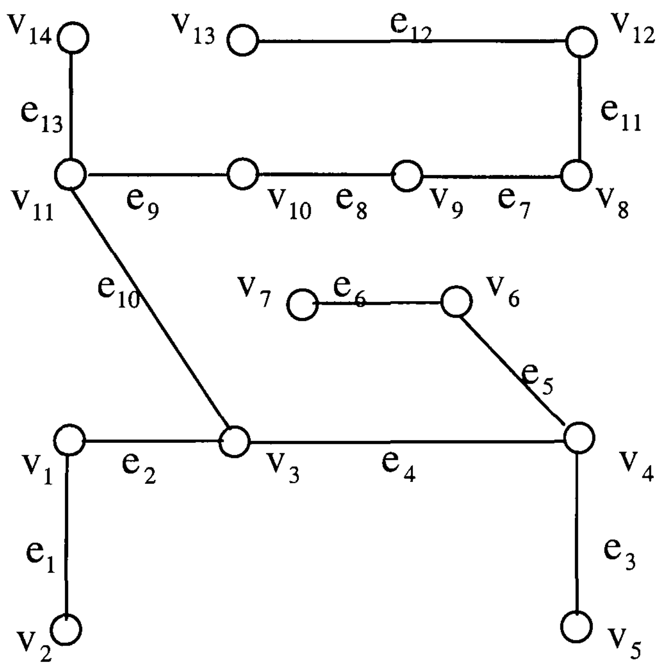


Figure 4.14b A spanning tree of the IEEE-14 network

4.7 The branches of a spanning tree

A branch B in a spanning tree is an open path, consisting of one or more edges. Each branch in the spanning tree has two **end-nodes.** The two end-nodes are either junction-nodes or one of them is a junction-node and the other one is a bottom-node. The first end-node is termed v_{First}, **the first-node** and the other end-node is termed the v_{Last}, **the last-node**.

The branch end-nodes are used to classify the branch. If the two end-nodes of a branch are junction-nodes then the branch type is classified as BJ. If the branch has different end-nodes, i.e. one junction-node and one bottom node, then the branch type is classified as BB.

The length of a branch, L, is the number of edges between its two end-nodes. **The end-edges** of a branch are termed **the first-edge** e_{First}, and **the last-edge** e_{Last}. The first-edge has the first-node and the last-edge has the last-node. If the branch has one edge then the first-edge and the last–edge are the same. If the branch has two edges then one edge is the first-edge and the other is the last-edge.

Let B represents a branch with EB be the set of edges of the branch. Let L be the length of B, and let v_{First} be the first-node of B and v_{Last} be the last-node in B. Similarly let e_{First} be the branch first-edge in B and let e_{Last} be the last-edge in B. The data that identifies the i^{th} branch in the spanning tree can then be written as follows:

$$B_i = \{ v_{First}, v_{Last}, BB, EB, L_i, e_{First}, e_{Last} \}; \qquad (4.12)$$

For example, the data of the branches of the spanning tree in figure 4.14b are shown in Table 4.1.

The ith branch	v_{First}	v_{Last}	Branch Type	EB	L_i	e_{First}	e_{Last}
1	v_2	v_3	BB	$\{e_1, e_2\}$	2	e_1	e_2
2	v_3	v_4	BJ	$\{e_4\}$	1	e_4	e_4
3	v_5	v_4	BB	$\{e_3\}$	1	e_3	e_3
4	v_4	v_7	BB	$\{e_5, e_6\}$	2	e_5	e_6
5	v_3	v_{11}	BJ	$\{e_{10}\}$	1	e_{10}	e_{10}
6	v_{11}	v_{13}	BB	$\{e_9, e_8, e_7, e_{11}, e_{12}\}$	5	e_7	e_{12}
7	v_{11}	v_{14}	BB	$\{e_{13}\}$	1	e_{13}	e_{13}

Table 4.1 The data of the branches of a spanning tree of the IEEE-14 network

Let the total length of a BJ branch be the length of the BJ branch plus the lengths of all BB and BJ branches, from the junction-node of the BJ branch, to the last bottom-node.

Let S_{iBB} be the sum of the lengths of the BB branches at the junction-node of B_i.

Then $S_{iBB} = L_{BB\text{-}1} + L_{BB\text{-}2} + \cdots$. (4.13)

Let S_{rBJ} be the length of the rth BJ branch to the bottom-node.

Then $S_{rBJ} = L_{rBJ} + S_{rBB}$. (4.14)

The total length of the B_i branch may then be defined as:

$S_i = L_i + S_{iBB} + S_{rBJ}$. (4.15)

For example, in Table 4.1, B_2 is a BJ branch with $L_2 = 1$. The junction-node, v_4, of B_2 has two BB branches B_3 and B_4. The length of B_3 is $L_3 = 1$ and the length of B_4 is $L_4 = 2$. Applying equation (4.13) gives the sum of lengths of the BB branches at B_2, i.e.

$$S_{2BB} = L_{BB\text{-}3} + L_{BB\text{-}4} = 1 + 2 = 3 .$$

Since there is no BJ branches at the junction-node, v_4, of B_2, then $L_{2BJ} = 0$ and equation (4.14) cannot be used, i.e.

$$S_{2BJ} = 0 .$$

Thus the total length of B_2 is obtained by using equation (4.15), i.e.

$$S_2 = L_2 + S_{2BB} + S_{2BJ} = 1 + 3 + 0 = 4$$

4.8 Overview of the spanning tree partitioning technique

The initial goal of the partitioning technique is to partition a spanning tree, into k balanced sub-spanning trees. The final goal of the partitioning technique is to partition the network into k subsystems such that the number of internal nodes of the k subsystems is balanced, and the number of boundary nodes in the network is balanced with the number of internal nodes of the i^{th} subsystem.

The technique begins by checking the network connectivity. If the network has a spanning tree then the network is connected. The row reduction method, as Sectioned 4.2.1, is then used to obtain a spanning tree T from A. The number of edges in T is $m_t = n - 1$ edges.

The partitioning technique, instead of searching for the k-1 cut-edges, finds one sub-spanning tree at a time and then cuts one edge. It terminates when k sub-spanning trees are obtained. Thus, it cuts k-1 cut-edges from the spanning tree.

The partitioning technique finds the sub-spanning trees by finding the branches of the sub-spanning tree, such that the total length of the branches of the sub-spanning tree is either m_p or m_p+1. Table 4.1 is used to find the branches as explained in Section 4.9.

The partitioning technique is flexible in selecting the starting branch. It **starts looking for a BB branch with maximum length.** If all the BB branches have the same lengths, then the partitioning technique starts from a BB branch with maximum length and with the first order in the table.

The partitioning technique is designed as a general partitioning technique, i.e. to partition every spanning tree into k equal or balanced sub-spanning trees and to cut only one edge. Thus, if the selected BB branch gives a sub-spanning tree so that the technique has to cut more than one edge, then the technique will select the next BB branch with maximum length or next in order in the table.

The partitioning technique then proceeds to find the branches of the first sub-spanning tree as explained in Section 4.9. After finding the first sub-spanning tree the partitioning technique cuts only one edge. The end-nodes of the cut-edge are boundary nodes, one boundary node belongs to the first sub-spanning tree and the second boundary node belongs to the new sub-spanning tree.

The technique then rearranges Table 4.1 by deleting the used branches. Then it repeats the process of finding a new BB branch with maximum length, then finding the branches of the new sub-spanning tree and then cutting one edge.

The process terminates when k sub-spanning trees are obtained, i.e. when the modified Table 4.1 is empty.

4.9 Finding the branches of a sub-spanning tree

A sub-spanning tree of m_p edges consists of one or more connected branches. Finding the branches of a sub-spanning tree is based on finding the first branch in each sub-spanning tree, and then by using its last-node, i.e. the junction node, to find the BB and/or the BJ branches where their lengths plus the first branch length balance with m_p. The first branch is a BB branch with maximum length. The length of B_1, first branch, is L_1 edges. Three cases are considered.

Case 1:

If $m_p > L_1$, then this sub-spanning tree has more than one branch. The last-node of B_1 is a junction-node. It may have BB branches or BJ branches or both. If it has BB branches and the length of any one of them is $> m_p$, then this BB branch is a sub-spanning tree. Thus, starting from the bottom node of this BB branch cut the $(m_p + 1)^{th}$ edge. If the lengths of the BB branches are $< m_p$ then the sum of the lengths of the BB branches is given by equation (4.13):

$$S_{BB} = L_{BB\text{-}1} + L_{BB\text{-}2} + \ldots;$$

and the total length of the first branch is given by equation (4.15):

$$S_1 = L_1 + S_{BB};$$

If $S_1 > m_P$ then cutting any of those BB branches produces unconnected and unbalanced sub-spanning trees, thus this spanning tree is rejected.

If $S_1 = m_P$, then B_1 and the BB branches form a sub-spanning tree, thus the other BJ branches must be cut.

If $S_1 < m_P$, then B_1 and the BB branches are part of the sub-spanning tree, and one or more BJ branches have to be added. Let B_2 be a BJ branch that is to be added, then:

Let $S_2 = S_1 + L_2$, then S_2 will be compared with respect to m_P as done with S_1.

Case 2:

If $m_p < L_1$ then B_1 is the first sub-spanning tree.

$S_1 = L_1$;

The $(L_1+1)^{th}$ edge is the cut-edge.

Case 3:

If $m_p = L_1$, then branches at the junction of B_1 may be of BB type or BJ type or they may be of both types. There are two possible cases related to the suitability of partitioning this spanning tree.

If the length of each BB branch is less than or equal to m_p then cutting any one of those BB branches gives unbalanced sub-spanning trees. Thus this spanning tree is not suitable and partitioning terminates.

If the length of any of the BB branches is more than m_p, then the partitioning technique will consider it the starting branch as in case 2.

If there are only BJ branches, then the B_1 branch is the first sub-spanning tree.

$S_1 = L_1$;

The first-edge of each BJ branch must be cut. This may and may not give equal or balanced results.

Thus, the partitioning technique finds the branches of each sub-spanning tree. The junction-node, at the end of each BJ branch, is used to find the other branches (or the remaining edges).

The nodes of the first-edge are of different types. One of them is always a bottom node and the other node is either a branch node or a junction node. The bottom node is considered as v_1 and the other node is v_2.

These steps of finding the branches of a sub-spanning tree are as follows:

1- Find from equation (4.8) m_P, the number of edges of the sub-spanning tree

2- Form the branches table as shown in Table 4.1.

3- From the branches table find B_i with maximum length.

4- If $L_i > m_P$, then

   a. Starting from the first-edge in B_i cut the $(m_P + 1)^{th}$ edge.

   b. Rearrange the table by deleting B_i or by deleting $(m_P + 1)$ edges from B_i.

   c. Go to step 3 to find a new sub-spanning tree

5- Find

   a. n_{BB}, the number of BB branches at the junction of B_i.

   b. n_{BJ}, the number of BJ branches at the junction of B_i.

6- If $L_i < m_P$, then

  a. Check the BB branches at the junction of B_i
  b. If the length of any of the n_{BB} branches is > m_P, then starting from the bottom-node of this BB branch cut the $(m_P + 1)^{th}$ edge. Delete the $(m_P + 1)$ edges of the BB branch from the table. Rearrange the table.
  c. By using equation (4.13) find S_{BB}, the sum of the lengths of the remaining BB branches.
  d. By using equation (4.15), find $S_i = L_i + S_{BB}$.
  e. If $S_i = m_P$ and if $n_{BJ} = 1$ then
    A. Cut the first-edge of the BJ branch.
    B. Delete the BJ, B_i and the remaining BB branches from the table.
    C. Rearrange the table.
    D. Go to 3.
  f. If $S_i = m_P$ and if $n_{BJ} > 1$ then
    A. Find the total lengths of each BJ branch to the bottom-node, i.e. let $S_{i1}, S_{i2}, \cdots, S_{in_{BJ}}$ be the total lengths of each of the n_{BJ} branches to its bottom-nodes.
    B. If the total length of any BJ branch is less than m_P then terminate.
    C. If the total length of every BJ branch is more than m_P, then cut the BJ branches from the first edge.
    D. Rearrange the table.
    E. Go to 3
  g. If $S_i < m_P$ then
    A. Find the total lengths $S_{i1}, S_{i2}, \cdots, S_{in_{BJ}}$ of each of n_{BJ} BJ branches to the bottom-node.
    B. Let $S_{2h} = S_i + S_{ih}$ for $h = 1,2,\ldots,n_{BJ}$.
    C. If $S_{2h} = m_P$ for $h = 1,2,\ldots,n_{BJ}$, then cut the other BJ

branches from the first-edge.

D. Rearrange the table.

E. Go to 3.

7- If $L_i = m_P$ then

a. If the length of any of BB branches is > m_P, then the partitioning technique will consider it B_i, the first branch with maximum length equalling L_i.

b. If any of BB branches has length $\leq m_P$, then cutting any edge gives unbalanced sub-spanning trees. Thus this spanning tree is rejected.

c. If there are only BJ branches, then

A. Find the total lengths $S_{i1}, S_{i2}, \cdots, S_{in_{BJ}}$ of each of n_{BJ} BJ branches to the bottom-nodes.

B. If the length of every BJ branch is > m_P then cut the first-edge of the BJ branches.

C. Rearrange the table.

D. Go to 3.

Example (4.4): Using the branches to partition the spanning tree

The IEEE 14-bus network shown in Figure 4.13 is to be partitioned using the spanning tree given in Figure 4.14. The spanning tree has $m_t = 13$ edges.

Let k = 2.

The number of cut edges = k-1 = 1.

Each sub-spanning tree has $m_P = 6$ edges.

The partitioning technique starts by finding the branches with maximum length. From Table 4.1, B_6 has the maximum length, $L_6 = 5$. Since $L_6 < m_P$ then check the branches at the junction of B_6. The junction–node of B_6 is v_{11}.

The junction-node has one BB branch and one BJ branch, i.e. $n_{BB} = 1$ and $n_{BJ} = 1$. From Table 4.1, the BB branch is B_7, and the BJ branch is B_5.

The BB branch is B_7, and $L_7 = 1 < m_P$, then $S_{BB} = 1$.

$$S_6 = L_6 + S_{BB} = 5 + 1 = m_P,$$

which indicates that the first-edge of the B_5 branch must be cut. Since B_5 has only one edge then B_5 will be deleted from the table. The first sub-spanning tree consists of the two branches B_6 and B_7.

The edges of the second sub-spanning tree = $\{e_7,\ e_8,\ e_9,\ e_{11},\ e_{12},\ e_{13}\}$.

The nodes of the first sub-spanning tree

$= \{v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}\}$.

Rearranging Table 4.1, after deleting B_5, B_6 and B_7, gives Table 4.2.

The i th branch	v_{First}	v_{Last}	Branch Type	EB	L_i	e_{First}	e_{Last}
1	v_2	v_4	BB	$\{e_1, e_2, e_4\}$	3	e_1	e_4
3	v_4	v_5	BB	$\{e_3\}$	1	e_3	e_3
4	v_4	v_7	BB	$\{e_5, e_6\}$	2	e_5	e_6

Table 4.2 Modification of Table 4.1; k=2.

From Table 4.2 B_1 has the maximum length, $L_1 = 3$. There are two BB branches at the junction-node of B_1. Those are B_3 and B_4. Thus

$$S_{BB} = L_3 + L_4 = 1 + 2 = 3;$$

$$S_1 = L_1 + S_{BB} = 3 + 3 = m_P.$$

The edges of the second sub-spanning tree = $\{e_1,\ e_2,\ e_3,\ e_4,\ e_5,\ e_6\}$.

The nodes of the first sub-spanning tree = the nodes of the first subsystem =

$=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$

Having found the nodes of each subsystem, the property of the cut matrix as given by example (4.2) is used to find the cut-edges between the two subsystems.

$V_1 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$.

$V_2 = \{v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}\}$.

From A, the sum of the nodes columns of $V_1 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ gives the following cut-edges= $\{e_{10}, e_{14}, e_{15}\}$.

The end nodes of the cut edges are the boundary nodes $\{v_3, v_4, v_6, v_8, v_{11}\}$;

The number of boundary nodes is $n_b = 5$;

The internal nodes in each subsystem are obtained by deleting the boundary nodes from the subsystem nodes, i.e. the internal nodes of $V_1 = \{v_1, v_2, v_5, v_7\}$;

The number of internal nodes in V_1 is $n_{1r} = 4$;

The internal nodes of $V_2 = \{v_9, v_{10}, v_{12}, v_{13}, v_{14}\}$, and $n_{2r} = 5$.

Since the difference between the internal nodes of the k=2 subsystems is 1, the internal nodes of the two subsystems are balanced. Since the number of boundary nodes equals the number of internal nodes of the second subsystem, the boundary nodes are balanced with the internal nodes.

Example 4.5: Using the spanning tree branches to partition the IEEE-14 network for k=3: as for Example 4.4, but with k = 3.

The number of cut edges $k = k - 1 = 3 - 1 = 2$ edges;

From equation (4.8), $m_P = \left\lceil \frac{14-3}{3} \right\rceil = 3$; and $R = 2$;

Thus the k sub-spanning tree edges are $\{3, 4, 4\}$.

From Table 4.1, B_6 has the maximum length.

Since $L_6 > m_P = 3$,

and

$L_6 > (m_P + 1) = 4$,

then the first sub-spanning tree has $m_{P-1} = m_P + 1 = 4$ edges from B_6.

$S_{BB} = 0$;

$S_6 = m_{P-1} + S_{BB} = 4 + 0 = 4$;

The edges of the first sub-spanning tree are $E_{SPS-1} = \{e_8, e_7, e_{11}, e_{12}\}$.

The nodes of the first sub-spanning tree are $V_{SPS-1} = \{v_8, v_9, v_{10}, v_{12}, v_{13}\}$.

Since $(m_{P-1} + 1) = 5$, the 5^{th} edge of B_6, namely e_9, is a cut-edge.

The cut-edges between the first subsystem and the other subsystems are obtained by adding the nodes columns in A, i.e.

The cut-edges = $\{e_{10}, e_{14}, e_{15}, e_{17}, e_{18}, e_{19}\}$.

Since B_6 consists of $\{e_8, e_7, e_{11}, e_{12}\}$ and the cut-edge e_9, then B_6 will be deleted from Table 4.1. The updated table is shown in Table 4.3.

The i^{th} branch	v_{First}	v_{Last}	Branch Type	EB	L_i	q_{First}	q_{Last}
1	v_2	v_3	BB	$\{e_1, e_2\}$	2	e_1	e_2
2	v_3	v_4	BJ	$\{e_4\}$	1	e_4	e_4
3	v_4	v_5	BB	$\{e_3\}$	1	e_3	e_3
4	v_4	v_7	BB	$\{e_5, e_6\}$	2	e_5	e_6
5	v_3	v_{14}	BB	$\{e_{10}, e_{13}\}$	2	e_{10}	e_{13}

Table 4.3 Modification (1) of Table 4.1; k=3.

From Table 4.3, the first branch is B_1, with $L_1 = 2$. The junction-node, v_3, of B_1 has one BB branch, B_5, and one BJ branch, B_2. Then, $n_{BB} = 1$ and $n_{BJ} = 1$. First, the technique considers the BB branch B_5:

$S_{5BB} = L_5 = 2$;

$S_1 = L_1 + S_{5BB} = 2 + 2 = 4$;

Since $S_1 = m_P + 1 = 3 + 1 = 4$; and since $n_{BJ} = 1$ then the branches in S_1, i.e. B_1 and B_5, represent the second sub-spanning tree.

The edges of the second sub-spanning tree are $E_{SPS\text{-}2} = \{e_1, e_2, e_{10}, e_{13}\}$.

The nodes of the second sub-spanning tree are $V_{SPS\text{-}2} = \{v_1, v_2, v_3, v_{11}, v_{14}\}$.

Thus B_2, the BJ branch is cut from the first-edge, i.e. e_4.

The cut-edges = $\{e_4, e_5, e_6, e_{13}, e_{18}, e_{19}\}$.

From the cut-edges, the boundary nodes are = $\{v_3, v_4\}$.

Note that B_2 has one edge only, the cut-edge e_4.

Table 4.3 may now be rearranged by deleting B_1, B_5 and the cut-edge e_4, which is B_2. The result is shown in Table 4.4.

The ith branch	v_{First}	v_{Last}	Branch Type	EB	L_i	q_{First}	q_{Last}
3	v_4	v_5	BB	$\{e_3\}$	1	e_3	e_3
4	v_4	v_7	BB	$\{e_5, e_6\}$	2	e_5	e_6

Table 4.4 Modification (2) of Table 4.1; k=3.

The technique proceeds looking for the third sub-spanning tree, starting from B_4, which has the maximum length, $L_4 = 2$. At the junction node v_4, of B_4 there is only one BB branch, i.e. B_3.

Thus

$S_{3BB} = L_3 = 1$;

$S_3 = L_3 + S_{3BB} = 2 + 1 = 3$;

Since $S_3 = m_P = 3$, then the branches in S_3 represent the third sub-spanning tree.

The edges of the third sub-spanning tree are $E_{SPS\text{-}3} = \{e_3, e_5, e_6\}$.

The nodes of the third sub-spanning tree are $V_{SPS\text{-}3} = \{v_4, v_5, v_6, v_7\}$.

The cut-edges = $\{e_4, e_5, e_6, e_{10}, e_{15}\}$.

This completes the partitioning procedure.

4.10 Simulation results

The spanning tree partitioning technique has been applied to partition several networks including the standard IEEE-14, IEEE-30 and IEEE-57 networks given in Appendix C. The source code of the partitioning technique was written in Matlab.

The spanning tree partitioning technique partitions the spanning tree of the IEEE 14-bus into equal or balanced k sub-spanning trees, following which the sets of boundary nodes and internal nodes in each subsystem are identified. The results are checked for satisfaction of the DSE constraints, i.e. equal or balanced cases. If the results are satisfied with the DSE constraints, then the results are saved, and the method continues with k increased by 1. If the results do not satisfy the DSE constraints, the procedure terminates.

Tables 4.5-4.7 show the results of balanced partitioning spanning trees of the IEEE 14-bus, the IEEE 30-bus and the IEEE 57-bus.

k	Number of cut-edges in T	m_P	Number of cut-edges in G	n_b	V_b	n_{ir}	V_{ir}
2	1	6	3	5	3,4,6,8,11	$n_{1r} = 4$; $n_{2r} = 5$	$V_{1r} = \{1,2,5,7\}$; $V_{2r} = \{9,10,12,13,14\}$

Table 4.5 The results of balanced partitioning the IEEE 14-bus

k	Number of cut-edges in T	m_P	Number of cut-edges in G	n_b	V_b	n_{ir}	V_{ir}
2	1	14	4	7	4,6,9,10, 12,24,25	$n_{1r} = 12$; $n_{2r} = 11$	$V_{1r} = \{1,2,3,5,7,8, 11,26,27,28,29,30\}$; $V_{2r} = \{13,14,15, 16,17,18,19,20,21, 22,23\}$

Table 4.6 The results of balanced partitioning the IEEE 30-bus

k	Number of cut-edges in T	$m_{P\text{-}i}$	Number of cut-edges in G	n_b	V_b	n_{ir}	V_{ir}
2	1	27; 28	8	14	7,9,19,20, 29,38,41, 42,44,45, 48,49,55, 56	$n_{1r} = 21$; $n_{2r} = 22$	$V_{1r} = \{1,2,3,4,5, 6,8,10,11,12,13, 14,15,16,17,18, 43,46,47,50,51\}$ $V_{2r} = \{21,22,23, 24,25,26,27,28, 30,31,32,33,34, 35,36,37,39,40, 52,53,54,57\}$

Table 4.7 The results of balanced partitioning the IEEE 57-bus

4.11 Performance evaluation

The network partitioning process principally involves a search for the minimum number of cut-edges among the network edges. The decision making process is used to evaluate whether the number of boundary nodes, which are the end-nodes of the cut-edges, is equal or balanced with the number of internal subsystem nodes.

Using the spanning tree partitioning technique to partition the network reduces the search-time and reduces the number of decision-making operations drastically.

The technique starts by constructing the branches table as Table 4.1. The number of operations to find the length of the i^{th} branch is $O(L_i)$, and since the total length of all branches is m_t, then the total number of operations to find all branches is $O(m_t)$.

The partitioning technique then uses the data in the table to find the starting branch, i.e. a branch with maximum length. One comparison operation, to the lengths of the branches, is sufficient to find such a branch.

In the partitioning technique, the decision-making process is a mathematical addition operation and a comparison operation. First, the sum of the lengths of branches of one direction is obtained, and compared with m_P, the edges of a sub-spanning tree. The decision is taken only at junction-nodes.

Thus the technique replaces the wide range searching process with a few steps of decision-making. The number of the decision-making steps is related to number of junction-nodes in the spanning tree.

The technique evaluates the suitability of the obtained spanning tree. If the obtained spanning tree is suitable it terminates when k balanced sub-spanning trees are obtained, otherwise it terminates without partitioning the spanning tree.

The check of the obtained k subsystems for meeting the DSE requirements also involves a matrix addition operation and comparison operation. When the matrix addition operation is performed on the n_i nodes columns from A, the result is the cut-edges between the i th subsystem and the other subsystems. For one subsystem, the matrix addition operation, maximally, takes $O(\frac{n}{k}-1)$. Thus for the k parallel subsystems it takes the same number of operations as for the one subsystem. Then, O(2k) set operations are used to obtain the boundary nodes and the internal nodes of each subsystem. Balanced partitioning is obtained by comparing the number of internal nodes of the k subsystems with the boundary nodes. Thus, the total number of operations taken by the partitioning technique is $O(\frac{n}{k}-1+2k)$.

4.12 Limitations of the partitioning technique

Performance of the partitioning technique is limited by a number of factors.

Limitation 1:

For k partitions, the technique is designed to cut (k-1) edges in the spanning tree. However, not every spanning tree can be partitioned into k **balanced** sub-spanning trees. If more than (k-1) edges are cut from a spanning tree, then there are more than k sub-spanning trees. Thus, if the technique is directed to cut more than (k-1) edges from the spanning tree, the procedure will terminate and the spanning tree will be rejected as unsuitable.

Limitation 2:

A network has many spanning trees. To overcome limitation 1, another spanning tree has to be obtained. As described so far, the technique obtains a spanning tree by using the row reduction method. To obtain another spanning tree the network incidence matrix needs to be permutated. The cost of permutation is undesirable. In an attempt to overcome this difficulty, alternative methods for obtaining a sequence of spanning trees have been investigated. A further difficulty is that there is no guarantee that each of the new spanning tree matrices will have the 'descending staircase' shape described in Section (4.3.2). Thus, the partitioning technique needs to be modified for use with a new spanning tree matrix.

Limitation 3:

The principal steps in the partitioning procedure described in this chapter may be summarized as follows:

Step 1: Partitioning of the obtained spanning tree into k balanced sub-spanning trees.

Step 2: Finding the boundary nodes and the internal nodes of each subsystem.

Step 3: The decision step. It checks whether the partitioned k subsystems satisfy the DSE requirements or not.

Step 3 illustrates a disadvantage of this procedure, shared by other partitioning techniques such as those described in Chapter 3, when applied to the DSE problem. The constraints which apply due to DSE are not incorporated into the partitioning algorithm; the decision on suitability of the result for DSE is made only at the very last moment in the procedure. If the result is unsuitable, much computational effort has been expended for little return.

These limitations lead to a revised approach of partitioning the network, in which the objective is to seek a substantial reduction in the computational effort required: (i) to generate new spanning trees; and (ii) to obtain a balanced partition of a network whilst meeting the DSE constraints. Emphasis is to be placed on identification of the sets of boundary nodes and the k-balanced internal nodes much earlier in the overall procedure. This aspect is addressed in Chapter 5. Generation of new spanning trees and the revised partitioning procedure are investigated in Chapter 6.

4.13 Chapter review

In this chapter, a new and computationally efficient technique for partitioning a spanning tree has been developed to satisfy the DSE requirements, i.e. to balance the total number of boundary nodes from the k subsystems with the number of subsystem internal nodes.

The partitioning technique described in this chapter is suitable for application to networks of general configuration and size. The technique is based on using the spanning tree properties rather than a purely heuristic procedure. It uses the spanning tree branches to find the connected k sub-spanning trees by cutting k-1 edges. The partitioning technique is designed to obtain balanced sub-spanning trees only, where 'balance' is in the sense of equal, or near-equal, numbers of edges in the sub-spanning trees.

The network has many different spanning trees. Partitioning a spanning tree of a network is sensitive to the obtained spanning tree. Some of the partitioned spanning trees may give balanced k sub-spanning trees and others may not give balanced sub-spanning trees. Some of the balanced k sub-spanning trees may satisfy the DSE requirements and others may not satisfy the DSE requirements.

The partitioning technique is fast and flexible in selecting the starting branch, i.e. it uses a table such as Table 4.1 to find the i^{th} branch that has the maximum length to start every new sub-spanning tree.

Once the k sub-spanning trees are obtained, the technique uses a fast operation to find the boundary nodes between the k subsystems, namely a matrix addition operation. Then the internal nodes of the k subsystems are obtained. Finally, the technique uses a single comparison operation to test the suitability of the k subsystems for DSE.

The performance of the technique has been tested on several IEEE networks. Following the DSE check, if the obtained spanning tree is suitable, then the result is accepted; if the obtained spanning tree is not suitable, the result is rejected. On successful completion of the procedure, a database representation of the partitioned subsystems can then be used by a DSE algorithm.

Chapter 5

The conditions of ideal balanced partitioning

5.1 General

Network partitioning, as stated in Chapter 1, is a conditional division operation. The DSE restrictions defined in Chapter 2 introduce three important constraints on the partitioning of a network. In this chapter, the DSE restrictions are examined in further detail, formalized in a mathematical form, and the conditions of ideal balanced partitioning are introduced.

5.2 The partitioning restrictions of DSE

The decomposition of state estimation into two levels with parallel computation at one level, introduced in chapter (2), introduces three important restrictions on partitioning of a network. Firstly, partitioning the network into k subsystems requires that: (i) every subsystem is defined by its internal nodes and boundary nodes; (ii) no overlapping occurs between the sub-system nodes; and (iii) all nodes in the global network are present in the partitioned model. Secondly, in order that the k parallel processors compute their tasks without unnecessary delay, their load should ideally be balanced in size. Balancing the k parallel processors necessitates that the network must be partitioned into k subsystems of identical sizes. Thirdly, determining k that satisfies the decomposed SE is of great significance. If k=1 (the global case), all computations will be done in one processor at lower level; if k=n, then all computations will be done in the upper level processor, since all nodes are then of boundary type. Both cases are attended by the complexity associated with the integrated state estimation problem, particularly for large networks. The number of subsystems k has a direct relation with the boundary nodes and an indirect relation with the internal nodes. In general, as k increases the number of boundary nodes increases and the number of internal nodes decreases. The SE requirements necessitate that the size of computation at the upper level should be no greater than that at the lower level. Accordingly, the best k is defined as that which minimizes the number of subsystem internal nodes whilst providing balance between internal nodes of all subsystems and the total number of boundary nodes. Thus, the task incorporating the DSE restrictions can be summarized as follow:

1. Define the k subsystems by partitioning with no overlapping.
2. Balance the internal nodes of the k subsystems.
3. Balance the global boundary nodes with the internal nodes.
4. Seek the maximum value of k such that balance can be maintained.

5.2.1 Defining the k subsystems

When the network is partitioned into k non-overlapping subsystems, each subsystem is partly characterized by its nodes, n_i in number. Whether balance is present or not, the sum of n_i across all values of i must equal n, the total number of nodes in the network.

Let V be the set of network nodes with n elements. Then

$$V = \{v_1, v_2, \ldots\ldots, v_n\}. \tag{5.1}$$

Let S_i represent the subset of nodes of the i^{th} subsystem and n_i be the number of elements of S_i. If k>1, then V is partitioned into k subsets of nodes such that:

$$V = \{S_1, S_2, \ldots, S_k\}; \tag{5.2}$$

and

$$V = S_1 \oplus S_2 \oplus \ldots\ldots \oplus S_k; \tag{5.3}$$

which implies that

$$S_i \cap S_j = 0 \quad \text{for } i, j = 1:k \text{ and } i \neq j.$$

Hence $n = \sum_{i=1}^{k} n_i$. (5.4)

Let the subset of nodes of the i^{th} subsystem be classified into two subsets, the subset of **boundary nodes** and the subset of **internal nodes**. Let V_{ib} be the subset of boundary nodes of the i^{th} subsystem with n_{ib} the number of elements of V_{ib}. Let V_{ir} be the subset of internal nodes in the i^{th} subsystem with n_{ir} be the number of elements of V_{ir}.

Then $n_i = n_{ib} + n_{ir}$. (5.5)

Let V_b be the set of **global boundary nodes** in the network with n_b be the number of elements of V_b, then:

$$V_b = V_{1b} \oplus V_{2b} \oplus ..... \oplus V_{kb}; \quad (5.6)$$

and $n_b = \sum_{i=1}^{k} n_{ib}$; (5.7)

Let V_r be the set of **global internal nodes** in the network with n_r the number of elements of V_r, then

$$V_r = V_{1r} \oplus V_{2r} \oplus ..... \oplus V_{kr},$$

and $n_r = \sum_{i=1}^{k} n_{ir}$. (5.8)

Thus, from equation (5.3) and equation (5.4), n, the network nodes, equals the sum of global boundary nodes and the total number internal nodes in the network, i.e.

$$n = n_b + n_r, \quad (5.9)$$

5.2.2 Balancing the internal nodes

Two or more subsystems have **balanced** internal nodes if the difference between them is no more than one. It is required by the DSE to balance the internal nodes of the k subsystems. Ideal balancing is achieved when

$$n_{1r} = n_{2r} = \ldots\ldots = n_{kr} \qquad (5.10)$$

Let n_p be the number of **internal nodes** per partition, i.e.

$$n_p = \left\lfloor \frac{n_r}{k} \right\rfloor, \qquad (5.11)$$

where $\lfloor \; \rfloor$ indicates rounding down to the nearest integer. If f_k is the remainder,

$$f_k = n_r - k.n_p; \qquad (5.12)$$

and the range of f_k is:

$$0 \le f_k \le k-1 \qquad (5.13)$$

Let the allowable difference between the internal nodes of the k subsystems to be one node, then

$$n_{pl} = n_p; \qquad (5.14)$$

and

$$n_{pu} = n_{pl} + 1. \qquad (5.15)$$

Then, if $f_k > 0$, then balancing is achieved with

$$n_{ir} = n_{pl} = n_p \quad \text{for i=1, ... , k-}f_k; \qquad (5.16a)$$

and

$$n_{ir} = n_{pu} = n_p + 1 \quad \text{for } i = k - f_k + 1, \cdots, k. \qquad (5.16b)$$

The set R of subsystems then consists of the disjoint subsets

$$R_L : (k - f_k) \text{ subsystems of size } n_{pl}; \qquad (5.17a)$$

And

$$R_U : f_k \text{ subsystems of size } n_{pu}. \qquad (5.17b)$$

If $f_k = 0$, then the balancing is achieved with

$$n_{ir} = n_{pl} = n_p \quad \text{for i=1, ... , k.} \qquad (5.18a)$$

The set R of subsystems then consists of the disjoint subsets

$$R = R_L : \text{k subsystems of size } n_{pl} = n_p. \qquad (5.18b)$$

5.2.3 The balancing number of boundary nodes

It is required by the DSE to partition the network such that n_b is balanced with n_{ir}. The value of n_b has a direct relationship with k, whilst the value of n_{ir} has an indirect relationship with k. When $k=1$, $n_b=0$ and $n_{ir}=n$; (i.e. $i=1$). As k increases, n_b increases and n_{ir} decreases. As k approaches n, n_b approaches n and n_{ir} approaches zero. Since, in a connected network, each subsystem must contain at least one boundary node, then the lower limit of n_b is

$$n_b \geq k. \tag{5.19}$$

An upper limit on n_b is provided by the balancing requirements for the DSE, i.e.

$$n_b \leq n_{pu};\ f_k > 0. \tag{5.20a}$$

$$n_b \leq n_{pl};\ f_k = 0. \tag{5.20b}$$

Hence, the range of n_b is given by:

$$k \leq n_b \leq n_{pu};\ f_k > 0.$$

$$k \leq n_b \leq n_{pl};\ f_k = 0.$$

5.2.4 The possible range of k

The lower limit of k is $k=2$. Since the relation between n_b and k is direct, and since the DSE requires $n_b \leq n_{ir}$, then equation (5.19) gives the upper limit of k, which balances n_b and n_{ir}. Accordingly, equations (5.20) give the range of k which balances the k n_{ir}'s at the lower level, and balances between the global boundary nodes at the upper level with n_{ir} at the lower level. This range is

$$n_b \geq k \geq 2 \tag{5.21}$$

Having introduced the partitioning restrictions of the DSE, applying these restrictions on the different connections a network may have is discussed in Section (5.3).

5.3 The possible range of cut-edges

Since $V_b = \{V_{1b}, V_{2b}, \cdots, V_{kb}\}$ is the set of global boundary nodes of the partitioned network, then let E_b **be only the set of cut-edges** between the V_b nodes such that the removal of E_b disconnects the network into k subsystems, and let m_b be the number of elements of E_b. Note that the edges between the boundary nodes of the ith subsystem are not cut-edges. For example, Figure 5.1 shows the boundary nodes and the set of cut-edges for a three subsystems partitioning. The set of cut-edges is $E_b = \{e_2, e_3, e_5, e_6, e_7\}$ and $m_b = 5$. The edges e_1 and e_4 are not cut-edges.

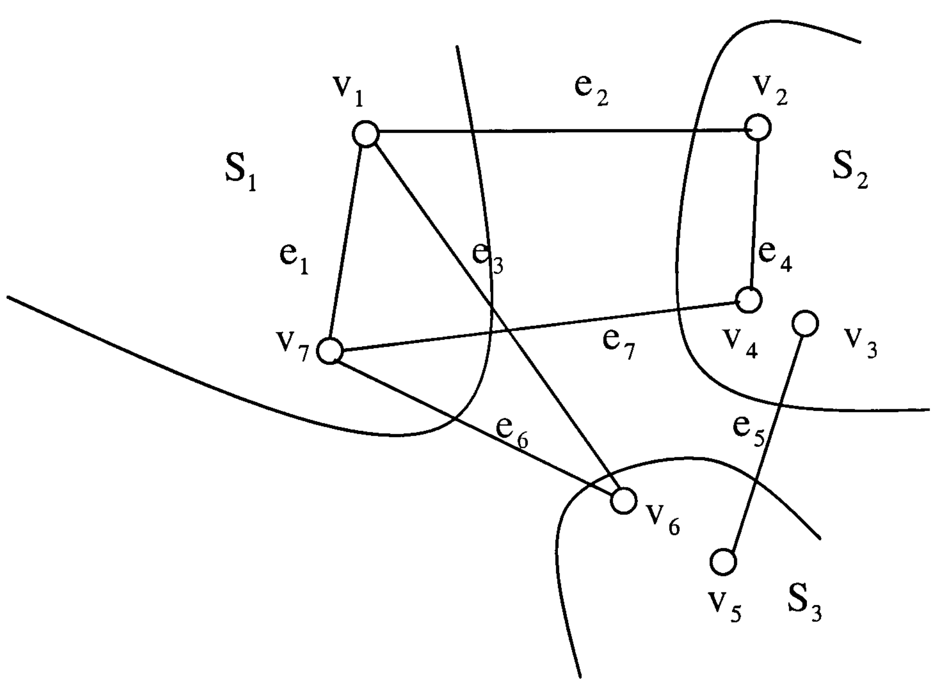


Figure 5.1 The set of cut-edges between a three subsystems

A cut-edge is termed an **independent cut-edge** if its end-nodes are not shared by any other cut-edges. For example, e_5, in Figure 5.1, is an independent cut-edge. Two or more cut-edges are **termed dependent cut-edges** if they are shared by one or more boundary node. For example, in Figure 5.1, e_2, e_3, e_6 and e_7 are dependent cut-edges, they share v_1, v_6 and v_7.

There is a relation between the number of cut-edges m_b, the number of boundary nodes n_b and the number of subsystems k. The set of cut-edges may be dependent, independent or combination of dependent and independent cut-edges. The dependent set of cut-edges may form a cycle and they may not form a cycle. The number of boundary nodes may be odd or even accordingly. If the cut-edges are independent, then

$$m_b = \left\lceil \frac{n_b}{2} \right\rceil . \qquad (5.22)$$

If the cut-edges are dependent and they are not in a cycle, then they form a tree, and the number of cut-edges between the nodes of V_b is

$$m_b = n_b - 1 . \qquad (5.23)$$

Let the cycle formed by the cut-edges only be termed **the boundary-cycle** and let c_b be the number of boundary cycles in network. If the cut-edges are dependent and $m_b \geq n_b$, then the cut-edges are forming one boundary-cycle, for which

$$c_b < c ; \qquad (5.24)$$

and

$$c_b = m_b - n_b + 1 . \qquad (5.25)$$

In this case, the distribution of the cut-edges between the boundary nodes has a direct relationship with k the number of subsystems, i.e.

if $n_b = k$ then $n_{ib} = 1$, thus $m_b = n_b = k$ and $c_b = 1$;

if $n_b > k$ then $n_{ib} \geq 1$. If $m_b \geq n_b$ then $c_b \geq 1$. In this case, the distribution of the cut-edges between the set of boundary nodes V_b has a direct relationship with the distribution of the V_b nodes between the k subsystems, i.e. the number and distribution of the cut-edges depend on k and n_{ib} for $i = 1, 2, \ldots, k$.

For example, let $n_b = 6$, then

If $k = 2$, then n_{ib} for $i = 1, 2$ can be one of the following combinations:

Case1: $n_{1b} = 1$ and $n_{2b} = 5$;

Case2: $n_{1b} = 2$ and $n_{2b} = 4$;

Case3: $n_{1b} = 3$ and $n_{2b} = 3$.

Case1: $n_{1b} = 1$ and $n_{2b} = 5$. The cut-edges are dependent, but they cannot form a cycle, then

$$m_b = n_b - 1 = 5.$$

Case2: $n_{1b} = 2$ and $n_{2b} = 4$. Then from every boundary node in V_{1b} there are four cut-edges to the four nodes in V_{2b}, one cut-edge to each node in V_{2b}. Since V_{1b} has $n_{1b} = 2$, then there are $(n_{1b} = 2)$ x $(n_{2b} = 4)$ = 8 cut-edges, i.e.

$$m_b = (n_{1b}).(n_{2b}) = (2).(4) = 8.$$

Case3: $n_{1b} = 3$ and $n_{2b} = 3$. Then from every boundary node in V_{1b} there are three cut-edges to the $n_{2b} = 3$ nodes in V_{2b}, one cut-edge to each node in V_{2b}. Then the maximum possible number of cut-edges between the two subsystems is:

$$m_b = (n_{1b}).(n_{2b}) = (3).(3) = 9.$$

If k=3, then n_{ib} for $i = 1, 2$ can be one of the following combinations:

Case1: $n_{1b} = 1$ and $n_{2b} = 1$ $n_{3b} = 4$;

Case2: $n_{1b} = 1$ and $n_{2b} = 2$ $n_{3b} = 3$;

Case3: $n_{1b} = 2$ and $n_{2b} = 2$ $n_{3b} = 2$;

Case1: $n_{1b} = 1$ and $n_{2b} = 1$ $n_{3b} = 4$.

Then $$\begin{aligned} m_b &= (n_{1b}).(n_{2b}) + (n_{1b}).(n_{3b}) + (n_{2b}).(n_{3b}) \\ &= (1).(1) + (1).(4) + (1).(4) = 9; \end{aligned}$$

Case2: $n_{1b} = 1$ and $n_{2b} = 2$ $n_{3b} = 3$.

Then $$\begin{aligned} m_b &= (n_{1b}).(n_{2b}) + (n_{1b}).(n_{3b}) + (n_{2b}).(n_{3b}) \\ &= (1).(2) + (1).(3) + (2).(3) = 11; \end{aligned}$$

Case3: $n_{1b} = 2$ and $n_{2b} = 2$ $n_{3b} = 2$.

Then $$\begin{aligned} m_b &= (n_{1b}).(n_{2b}) + (n_{1b}).(n_{3b}) + (n_{2b}).(n_{3b}) \\ &= (2).(2) + (2).(2) + (2).(2) = 12; \end{aligned}$$

Let $m_{b\text{-max}}$ be the maximum number of cut edges between the nodes of V_b in the k subsystems, then

$$\begin{aligned} m_{b\text{-max}} &= (n_{1b}).(n_{2b}) + (n_{1b}).(n_{3b}) + \ldots + (n_{1b}).(n_{kb}) \\ &+ (n_{2b}).(n_{3b}) + (n_{2b}).(n_{4b}) + \cdots + (n_{2n}).(n_{kb}) \\ &\vdots \\ &+ (n_{(k-1)b}).(n_{kn}) \end{aligned} \tag{5.26}$$

This shows that, for a single value of n_b, there may be different numbers of cut-edges.

5.4 The partitioning restrictions and the network different connections

One of the difficulties facing designing a general partitioning technique is that a network of m edges and n nodes has many different possible connections. The m edges can be connected in many different ways between the n nodes to form the network. When the different connections of a network are partitioned into k to satisfy the DSE restrictions, some of the different connections can give the balanced partitioning results, and many other connections cannot give the balanced partitioning results.

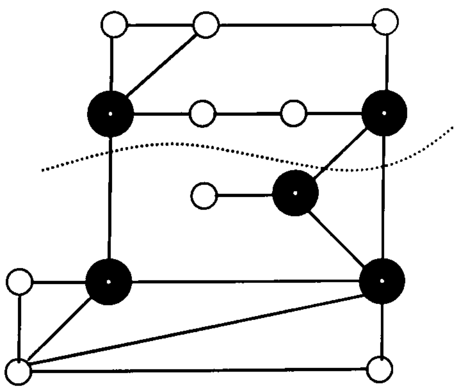


Figure 5.2a A network with 14 nodes and 20 edges

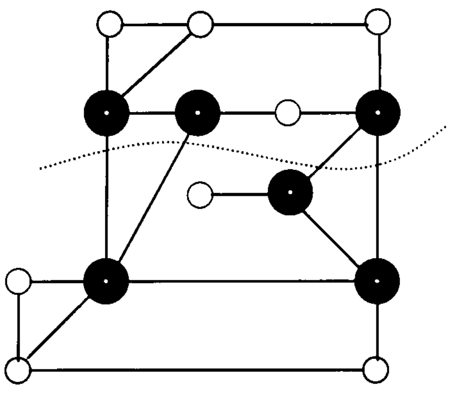


Figure 5.2b Another network with 14 nodes and 20 edges

For example, Figure 5.2a is the IEEE 14-bus with m=20 edges and with n=14 nodes. Figure 5.2b is another possible connection with m=20 edges and with n=14 nodes. If those two different network connections are partitioned into two parts, such that the number of internal nodes in the i^{th} part is equal or balanced with the number of the global boundary nodes, then they may give the equal or balanced partitioning results and they may give different partitioning results. For example the connection of Figure 5.2a gives the balanced partitioning results. These results are $n_b = 5$, $n_{1r} = 4$ and $n_{2r} = 5$. The connection of Figure 5.2b, however, cannot give balanced partitioning results. The partitioning results of Figure 5.2b are $n_b = 6$ and $n_{1r} = n_{2r} = 4$. It is also possible to obtain balanced partitioning results by changing the end nodes of one or more edges.

There are few possible connections of G = (14, 20) other than Figure 5.2a that can give balanced partitioning results as Figure 5.2a, and there are many other possible connections of G = (14, 20) that cannot give balanced partitioning results as Figure 5.2b.

The possibility to know that a network connection can or cannot give balanced partitioning results, prior to use of a partitioning technique, remains an unsolvable problem for general network.

The concern here is on those connections, which can give equal or balanced partitioning results. Those connections share common properties between them.

The first property is related to the existence of the equal or balanced partitioning values between the boundary nodes and the internal nodes in the network, i.e. the n_b value and the n_{ir} values for i=1,2, . . , k. The second property is related to the existence of the exact number of cut-edges that gives this number of boundary nodes in that network connection. The ideal conditions of partitioning a network are discussed in Sections (5.5).

5.5 The conditions of ideal balanced partitioning

An ideal balanced partitioning of a network is a conditional partitioning in which the number of global boundary nodes is balanced with the number of internal nodes in the i^{th} subsystem for $i = 1,2,...,k$.

Thus the ideal balanced partitioning conditions can be formulated as follows:

Condition 1: Let the conditional partitioning be termed **equal partitioning** if

(1.1) $n_b = n_p$;

(1.2) $f_k = 0$, i.e. $n_{ir} = n_p$ for $i = 1,2,...,k$.

Condition 2: Let the conditional partitioning be termed **balanced partitioning** if the difference is one between n_b and n_p. This has two cases:

Case 1:

(2.1) $n_b = n_{pl}$;

and the internal nodes of the k subsystems have the difference of one, i.e.

(2.2) $f_k > 0$, i.e. $n_{ir} = n_p = n_{pl}$ for $i = 1,2,...,j$,

and $n_{ir} = n_p + 1 = n_{pu}$ for $i = j+1,...,k$.

Case 2:

(3.1) $n_b = n_p + 1$;

and the internal nodes of the k subsystems are either equal, i.e.

(3.2) $f_k > 0$, i.e. $n_{ir} = n_p = n_{pl}$ for $i = 1,2,...,k$

or they have only a difference of one, i.e.

(3.2) $f_k > 0$, i.e. $n_{ir} = n_p = n_{pl}$ for $i = 1,2,...,j$,

and $n_{ir} = n_p + 1 = n_{pu}$ for $i = j+1,...,k$.

In all cases, n is given by equation (5.9), i.e. $n = n_b + n_r$.

Applying these conditions for a global ideal balanced-partitioning gives the following:

From condition (1.2), since $n_{ir} = n_p$ for $i = 1,2,...,k$, then $n_r = k.n_p$.

Equation (5.9) can be written as follows: $n_r = n - n_b$.

Substituting condition (1.2) in equation (5.9) gives: $k.n_p = n - n_b$.

Substituting condition (1.1), i.e. $n_b = n_p$, and solving for n_b:

$$k.n_b = n - n_b;$$

Therefore the exact value of n_b that satisfies condition 1 is given by

$$n_b = \frac{n}{k+1}.$$

The upper and lower limit of n_b can be determined by equations (5.20a) and (5.20b) according to the value of f_k.

If $f_k = 0$, $k \le n_b \le n_{pl}$.

If $f_k > 0$, $k \le n_b \le n_{pu}$.

In all cases

$$n_b \le \frac{n}{k+1}. \qquad (5.28)$$

Equation (5.28) gives the upper limit of n_b, global boundary nodes in the network, which is balanced with n_p the internal nodes of a partition. In practice, n_b is preferable to be $n_b \le \frac{n}{k+1}$ for efficient computation. The global boundary nodes, n_b, is an integer number, and the result of dividing n by (k+1) is not always an integer number, hence the division result is rounded down to the lower integer $\left\lfloor \frac{n}{k+1} \right\rfloor$ and rounded up to $\left\lceil \frac{n}{k+1} \right\rceil$ the upper integer, giving the range of n_b, i.e.

$$n_{bl} = \left\lfloor \frac{n}{k+1} \right\rfloor, \qquad (5.28a)$$

and

$$n_{bu} = \left\lceil \frac{n}{k+1} \right\rceil;$$

And, in all cases,

$$n_{bu} = n_{bl} + 1. \qquad (5.28b)$$

This theoretical n_b will be equal or balanced with n_p the internal nodes of the i^{th} part. The following theorem establishes that the network set of nodes possess the global balance property.

Having introduced the conditions for ideal balanced partitioning and the possible range of cut-edges between the boundary nodes that gives this ideal balanced partitioning, those conditions and the possible range of number of the cut-edges are the bases of Theorem (5.1).

Theorem (5.1)

If a network G=(n, m) is partitioned into k parts, k>1, such that

(1) $n_b = \frac{n}{k+1} \geq k$

and

(2) $n_p = \frac{n - n_b}{k} \geq k - 1,$

then

(1) $n_b = n_p$ or $n_b \cong n_p$.

Also, if $c_b = 0$, i.e. the cut-edges only do not form a cycle, then the range of the number of cut edges is

(2) $\left\lceil \frac{n_b}{2} \right\rceil \leq m_b \leq (n_b - 1)$,

and if $c_b > 0$, i.e. the cut-edges only form at least one cycle, then the range of the number of the cute-edges is

(3) $n_b \leq m_b \leq m_{b\text{-max}}$.

Proof:

Let the network set of nodes be given as described in Section (5.2.1), i.e.

$$V = \{v_1, v_2, \ldots\ldots, v_n\}. \tag{5.1}$$

Let V be partitioned into k>1 subsystems, as described in Section (5.2.1), i.e.

$$V = \{S_1, S_2, \ldots, S_k\}, \tag{5.2}$$

$$V = S_1 \oplus S_2 \oplus \ldots\ldots \oplus S_k. \tag{5.3}$$

Let n, the network nodes be classified and defined as described in section (5.2.1), i.e.

$$n = \sum_{i=1}^{k} n_i. \tag{5.4}$$

$$n_i = n_{ib} + n_{ir}. \tag{5.5}$$

$$n_b = \sum_{i=1}^{k} n_{ib}. \tag{5.7}$$

$$n_r = \sum_{i=1}^{k} n_{ir}. \tag{5.8}$$

and $n_r = n - n_b$. $\quad$ (5.9)

Let n_p be the number of internal nodes per subsystem as defined in Section 5.2.2, i.e.

$$n_p = \left\lfloor \frac{n_r}{k} \right\rfloor = \left\lfloor \frac{n - n_b}{k} \right\rfloor. \tag{5.11}$$

$$f_k = n_r - k.n_p. \tag{5.12}$$

$$0 \leq f_k \leq k - 1. \tag{5.13}$$

Let n_{pl} and n_{pu} are as defined in Section 5.2.2

$$n_{pl} = n_p; \tag{5.14}$$

$$n_{pu} = n_{pl} + 1. \tag{5.15}$$

The proof consists of establishing firstly, necessary conditions then sufficient conditions. The necessary conditions consists of determining firstly the values of n_b and n_p. Then determining the balance between n_b and n_p.

The first necessary condition is to determine the balanced value of n_b:

if $n_b = \frac{n}{k+1} = k$, then each subsystem has exactly one boundary node;

if $n_b = \frac{n}{k+1} > k$, then each subsystem has at least one boundary node.

The second necessary condition is to determine the value of n_p by dividing the global internal nodes, $n_r = n - n_b$, between the k subsystems such that:

(1) The number of internal nodes in each subsystem is equal or greater than (k-1); and

(2) The k subsystems internal nodes are equal or balanced.

Case 1:

If $n_b = k$, the balanced partitioning is achieved only if $n_p = n_{pl} = \left\lfloor \frac{n - n_b}{k} \right\rfloor = k - 1$;

and the equal partitioning is achieved only if $n_p = n_{pu} = \left\lfloor \frac{n - n_b}{k} \right\rfloor = k$.

Case 2:

If $n_b > k$, then the balanced value of n_p is achieved only if one of the following two cases occurs.

Case 1: $f_k > 0$, then

$$n_{ir} = n_{pl} = \left\lfloor \frac{n - n_b}{k} \right\rfloor > k \quad \text{for } i = 1 : k - f_k; \tag{5.16a}$$

and

$$n_{ir} = n_{pu} = n_p + 1 \quad \text{for } i = k - f_k + 1 : k; \tag{5.16b}$$

Then the set R of subsystems then consists of the disjoint subsets

$$R_L : (k - f_k) \text{ subsystems of size } n_{pl}; \tag{5.17a}$$

And

R_U: f_k subsystems of size n_{pu}. (5.17b),

Case 2: $f_k = 0$, then equal partitioning is achieved with

$n_{ir} = n_{pl} = n_p \quad \text{for } i = 1:k;$

and

$R = R_L$: k subsystems of size $n_{pl} = n_p$. (5.18)

and

$n_r = kn_p$.

Now, the necessary conditions for global balance between the boundary nodes and the internal nodes can be formulated as follows:

1. $n_b = n_p$.

2. $n_{1r} = n_{2r} = \cdots n_{1r} = n_p = n_{pl}$, i.e. $n_r = kn_p$ and $f_k = 0$.

Then by substituting in the network nodes defined by equation (5.9), i.e.

$n = n_b + n_r$

and solving for n_b,

$n = n_b + kn_p = n_b + kn_b$.

Then,

$$n_b = \frac{n}{k+1}.$$

The upper limit on n_b is given by equation (5.20) to satisfy the balancing requirements for the DSE, i.e.

$n_b \le n_{pu}$; $f_k > 0$.

$n_b \le n_{pl}$; $f_k = 0$.

Thus

$$n_b \le \frac{n}{k+1}.$$

The sufficient conditions:

The sufficient conditions are combination of two parts: the number of cut-edges between the boundary nodes, and the dependent and independent relationships between those cut-edges.

It is suffice to know that:

if the cut-edges are independent, then $c_b = 0$, and $$m_b = \left\lceil \frac{n_b}{2} \right\rceil;$$

and if the cut-edges are dependent and $c_b = 0$, then the cut-edges form a tree

and $$m_b = n_b - 1.$$

If the cut-edges are dependent or mixed of dependent and independent and $c_b > 0$, then

$$n_b \le m_b \le m_{b\text{-max}}; \text{ where } m_{b\text{-max}} \text{ is defined in equation (5.27)}$$

and $$c_b = (m_b - n_b + 1) \le c.$$

●

if the number of cut edges is more than n_b then $c_b \ge 1$.

5.6 Theoretical balanced partitioning results

The theoretical partitioning balanced values, i.e. n_b and $n_{1r}, n_{2r}, \ldots, n_{kr}$ that satisfy the DSE restrictions for any network can be obtained by applying theorem (5.1). For example, the theoretical balanced partitioning values for the IEEE-14 network are calculated for k=2 and for k=3.

5.6.1 Theoretical balanced results for k=2

The IEEE-14 has n = 14. Let k = 2, then

1. Calculating the theoretical n_b is done by equation (5.28), i.e.

$$n_b = \left\lfloor \frac{n}{k+1} \right\rfloor;$$

Substituting n=14 and k=2 gives the following result

$$n_b = \left\lfloor \frac{14}{2+1} \right\rfloor = 4;$$

then

$$n_{bl} = n_b = 4;$$

$$n_{bu} = n_b + 1 = 5.$$

2. Calculating the internal nodes is done by equation (5.9), i.e.

   $n_r = n - n_b$;

   2.1 Case $n_{bl} = n_b = 4$, substituting the values of n and n_b, gives the following:

   $n_r = 14 - 4 = 10$, internal nodes.

   The number of internal nodes in each subsystem is given by equation (5.11), i.e.

   $n_p = \left\lfloor \frac{n_r}{k} \right\rfloor$.

   Substituting the values gives the following:

   $n_p = \left\lfloor \frac{10}{2} \right\rfloor = 5$; and $f_k = 0$.

   Thus

   $n_{1r} = n_{2r} = n_p = 5$.

This theoretical $n_b = 4$ is the minimum value which provides balance between the two levels, i.e. the global boundary nodes at the upper level and the equal internal nodes of the k subsystem at the lower level as shown in Figure 5.3.

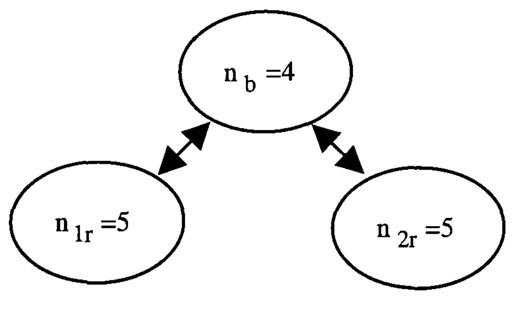


Figure 5.3 Theoretical results of partitioning the IEEE-14 network when k=2 and $n_b = n_{bl} = 4$

2.2 Case $n_{bu} = n_b + 1 = 5$,

Substituting the values of n and n_b, gives the following:

The total internal nodes are $n_r = n - n_{bu} = 14 - 5 = 9$;

The number of internal nodes per partition $n_p = \left\lfloor \frac{n_r}{k} \right\rfloor = \left\lfloor \frac{9}{2} \right\rfloor = 4$;

$f_k = 9 - 2x4 = 1$.

Therefore $n_{1r} = 4$ and $n_{2r} = 5$, as shown in Figure 5.4.

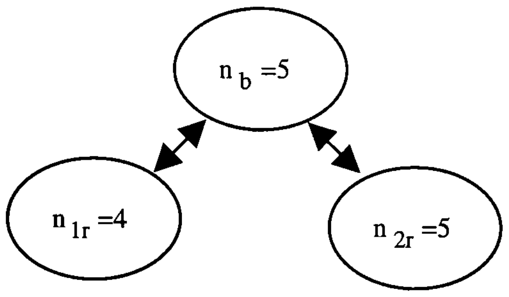


Figure 5.4 Theoretical results of partitioning the IEEE-14 network when k=2 and $n_b = n_{bu} = 5$

These partitioning values k, n_b and n_{ir} for i=1,2, satisfy the DSE restrictions.

5.6.2 Theoretical balanced results for k=3

Let k = 3, then

1. The theoretical n_b is calculated by equation (5.28) for k=3, gives:

$$n_b = \left\lfloor \frac{14}{3+1} \right\rfloor = 3;\ n_{bl} = n_b = 3;\ n_{bu} = n_{bl} + 1 = 4.$$

Case 1: $n_b = n_{bl} = 3$

$$n_r = n - n_{bl} = 14 - 3 = 11;$$

$$n_p = \left\lfloor \frac{11}{3} \right\rfloor = 3; \text{ and } f_k = 2;$$

R_L: 1 subsystem $n_{1r} = n_p = 3$;

R_U:2 subsystems $n_{ir} = n_p + 1 = 4$; for i=2,3

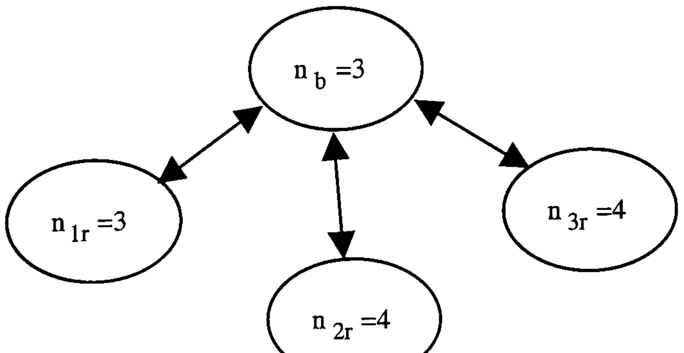


Figure 5.5 Theoretical results of partitioning the IEEE-14 network when $n_b = n_{bl} = 3$

These theoretical partitioning values shown in Figure 5.5 satisfy the DSE restrictions.

Case 2: $n_b = n_{bu} = 4$

$n_r = n - n_{bu} = 14 - 4 = 10$;

$n_p = \left\lfloor \frac{10}{3} \right\rfloor = 3$; and $f_k = 1$

R_L: 1 subsystem $n_{ir} = n_p = 3$; for i=1,2

R_U: 2 subsystems $n_{ir} = n_p + 1 = 4$; for i=3.

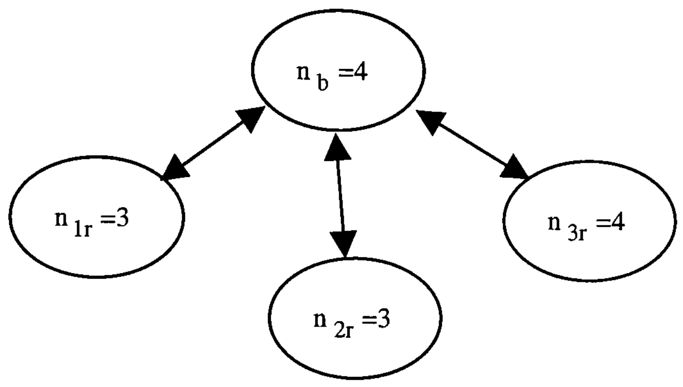


Figure 5.6 Theoretical results of partitioning the IEEE-14 network when $n_b = n_{bu} = 4$

These theoretical partitioning values shown in Figure 5.6 satisfy the DSE restrictions.

5.7 Chapter review

The introduction of Theorem (5.1), the ideal balancing partitioning theorem, defines and determines without using a partitioning technique, the balanced partitioning values, i.e. n_b, n_{ir} for $i = 1, 2, ..., k$ and the range of m_b.

The goal of the existing partitioning procedures is to find the balanced partitioning values. The existing methods for finding these values are heuristic, uncertain and time consuming.

Knowing the ideal balanced partitioning values, as described by Theorem (5.1) changes the direction, i.e. the goal and the methodology, of the partitioning techniques. These values are now known, thus the new goal of the partitioning procedure is not to find these values but to check of the existence of these values in the given network.

The goal of the partitioning procedure now is to use these values and to decide whether the given network has these values or not and if the network has these values which set of edges is the set of cut-edges which need to be removed to give these ideal balanced partitioning values.

The progress towards these goals requires more understanding about the natural network properties. Thus identification of the suitable cut-edges and how to find them is postponed to Chapter 8 and more network properties are introduced in Chapter 7.

Chapter 6 presents a new fast method for partitioning a network, i.e. to obtain the ideal balanced partitioning values. The technique illustrates an application of Theorem (5.1).

Chapter 6

A fast maximum degree technique

6.1 General

In this chapter, a new and fast partitioning technique is described to find the balanced partitioning values, as defined by theorem (5.1), of a directed network. The proposed partitioning technique finds first the global boundary nodes then the internal nodes of the k subsystems.

The algorithm of the maximum degree partitioning technique is based on finding two minimum covering sets of nodes with higher degree, one covering set from the directed network and one covering set from a spanning tree of the network. The intersection of the two sets gives a subset of nodes termed the common nodes. The common nodes are connected in the network and they are not always connected in the spanning tree. The set of common nodes represents the minimum global boundary nodes in the network. The connected subset of common nodes in the spanning tree represents the minimum global boundary nodes in the spanning tree. The sets of edges between the sets of common nodes in the network and in the spanning tree are termed the set of common edges in the network and the set of common edges in the spanning tree. Both sets of common edges contain the cut-edges and contain edges that are not cut edges.

Cutting (k-1) edges from the common edges of the spanning tree partitions the spanning tree into k sub-spanning trees and guarantee that the end-nodes of the cut-edges are from the boundary nodes of the spanning tree. Each sub-spanning tree represents a subsystem. The nodes of each sub-spanning tree consist of the internal nodes and the boundary nodes.

The equal or balanced partitioning is achieved by selecting (k-1) edges from the common edges, which gives the most balanced k sub-spanning trees.

The range of selecting the set of cut edges is k or less. After finding the k sub-spanning trees, the technique proceeds to find the remaining boundary nodes for each subsystem from the eliminated edges, and then starts a new spanning tree.

6.2 An overview of the maximum degree technique

Following the generation of a spanning tree, the method proceeds, firstly with the partitioning of the spanning tree, and then with partitioning of the network. The extent to which the result meets the objectives of partitioning for DSE is therefore dependent on the choice of spanning tree. The bulk of the computation involved lies in the generation of each spanning tree. Even moderate sized networks are characterized by very large numbers of possible spanning trees. A novel feature of the method to be described lies in the very rapid generation of a high proportion of all possible spanning trees. The opportunity is then presented for selection of the resulting network decomposition which best meets the objectives of partitioning for DSE. In this sense, the method to be described is optimal.

The maximum degree technique is based on obtaining two covering sets, one covering set from the network and one covering set from the spanning tree, such that the number of nodes in each covering set is minimum and the nodal degree of each set is the highest. The nodes of each subset contain connected and unconnected nodes. The unconnected nodes, in each subset, are deleted from the subset. The remaining nodes in each subset are connected and with maximum degree. The intersection of the two remaining subsets gives a new subset of nodes with maximum degree. The new subset represents the minimum boundary nodes for that spanning tree. The set of edges between the minimum boundary nodes is termed common edges. A spanning tree is partitioned into k sub-spanning trees by cutting (k-1) edges from the spanning tree. The set of common edges consists of one or more subsets of (k-1) edges. Each subset can partition the spanning tree into k sub-spanning trees. The equal or balanced partitioning is achieved by selecting one subset of the cut subsets, which gives the most balanced k sub-spanning trees. The range of selecting the cut set of edges is k or less. After finding the k sub-spanning trees, the technique proceeds to find the remaining boundary nodes for each subsystem from the eliminated edges, and then starts a new spanning tree.

The row reduction method, introduced in Section 4.3.1 is used to obtain a spanning tree of the given network. Then a new method, described in Section 6.4.1 is used to identify the network cycles. Following this, a permutation of the edges of each cycle is used to derive all possible spanning trees.

6.3 The number of spanning trees in a network

If the row reduction method, introduced in Section 4.3.1 is performed on the incidence matrix A, then every elimination step has a meaning for the graph, and, at the end of the elimination process, a graph without loops is produced. This is a spanning tree for the directed network. Its edges span the graph, and its rows span the row space of matrix A.

The number of cycles in a graph is $c = m - n + 1$ [64]. The value of c is also the number of eliminated edges from the graph to obtain one spanning tree [64], one edge from each cycle. Adding one of the eliminated edges (or rows) to the spanning tree would close a loop. These facts are used to obtain the network spanning trees. The standard IEEE-14 network shown in Figure 6.1 is used to illustrate these facts.

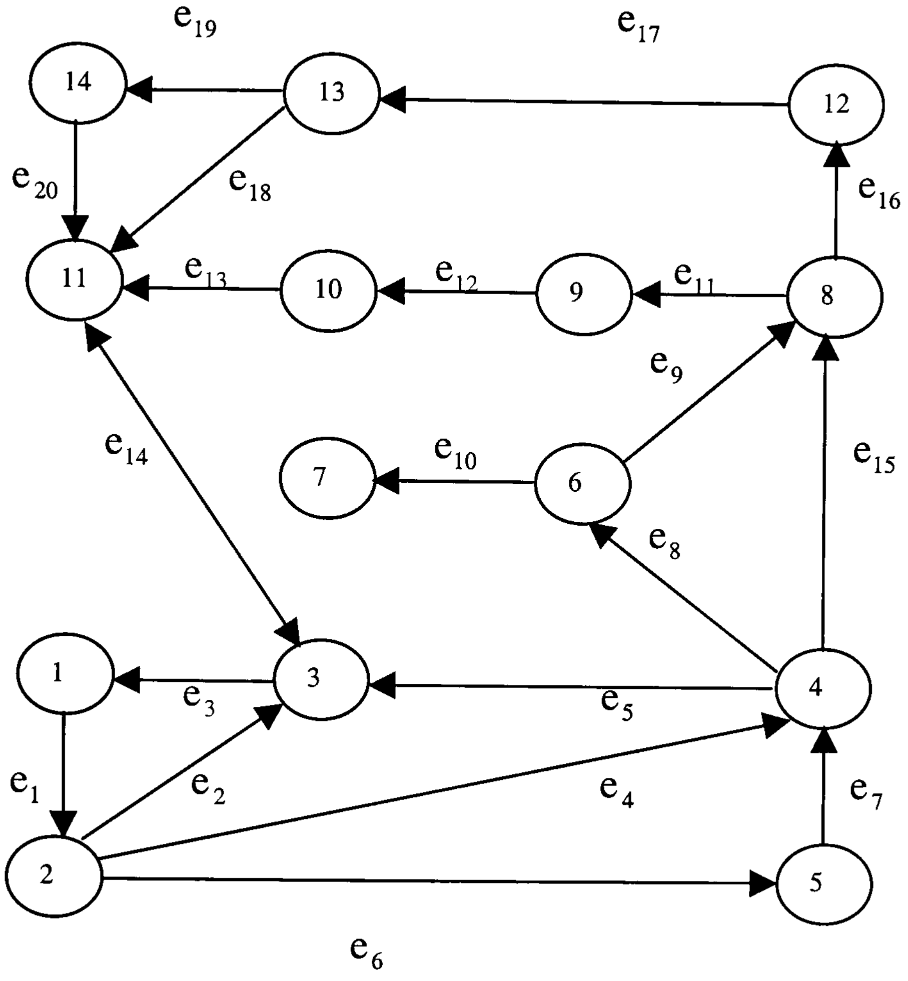


Figure 6.1 The IEEE 14 network

Any spanning tree is obtained by eliminating m-n+1 edges from the network. Following elimination, the nodes degree structure is modified. When an edge is eliminated, the degree of each of its node will reduce by one. If more than one edge is eliminated from a node, then its degree will reduce by the number of eliminated edges. Different spanning trees have different nodal degree structures.

Since the rank of the incidence matrix A is (n-1) [64], the removal of any column of A will give a (m x (n-1)) matrix B of the same rank [64]. B is termed the reduced incidence matrix. The total number of spanning trees of the network equals the determinant of matrix B^TB [64]. In the case of the IEEE 14-node network, this gives a total number of 3909 different possible spanning trees.

6.4 The edge-edge connectivity matrix

Gaussian elimination takes O (n^3), [64], operations to obtain a spanning tree. Different spanning trees may be obtained by permutating the incidence matrix, and then by applying the elimination method.

An alternative, faster way to obtain all possible spanning trees is introduced. The new method is based firstly on finding the network cycles such that each cycle is defined by its edges. Secondly, deleting one edge from each cycle such that no edge is repeated generates a spanning tree. Repeatedly, changing one edge from the set of eliminated edges and then deleting the set of eliminated edges from A generates a new spanning tree. Obtaining the network cycles is described in Section 6.4.1, and generating not only one spanning tree but also all possible spanning trees is described in Section 6.4.2.

6.4.1 The network cycles

The cycle definition is introduced in Appendix A. The number of independent cycles in the network is $c = m - n + 1$ cycles. To obtain one spanning tree, $c = m - n + 1$ edges must be eliminated from the network, one edge from each cycle. If, in the other side, the spanning tree of a network is given and it is required to obtain the c cycles, then to obtain one cycle, one of the eliminated edges must be added to the spanning tree. To obtain the c cycles, the c eliminated edges, (i.e. the co-spanning tree edges) must be added to the spanning tree. This fact, i.e. adding the c eliminated edges to the spanning tree produces the c cycles, is used to obtain the network cycles.

The number of cycles, c, in a network may be very large, and obtaining the edges of the c cycles manually is time consuming and not a practical proposition. Thus a fast method to obtain the network cycles, i.e. the edges of each cycle is described.

Let E be the (m x m) edge-edge connectivity matrix defined as follows:

$$E = AA^T.$$

The elements of E may be obtained simply from logical operations. The entries of E are:

2 if i = j ; 1 if the ith edge has a connection with the jth edge and they have the same direction;

-1 if the ith edge has a connection with the jth edge but with different direction;

0 if the ith edge has no connection with the jth edge.

Let S be the eliminated edges matrix of dimension (c x m) such that each row is a zero row, apart from two (+1) elements, one at the ith eliminated edge and one at the ith start-edge. The **start-edge** is a spanning tree edge having a common node with the ith eliminated edge. For example, the edges e_1 or e_3 in the spanning tree of Figure 6.2 can be a start edge with respect to the eliminated

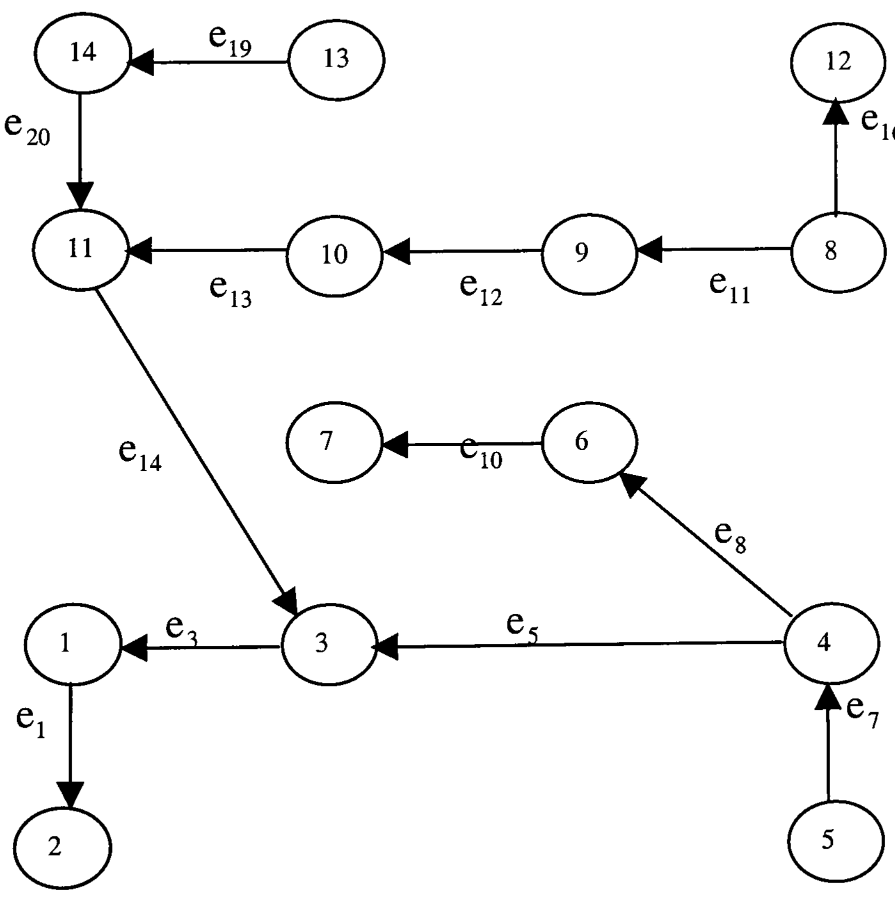


Figure 6.2 A spanning tree of the IEEE-14 network

edge e_2. The network cycles can then be characterized by a matrix C, where

C = S.E.

This operation may be efficiently performed with exploitation of sparsity, in which case multiplication takes O (4m) operations. For the IEEE 14-node network, the network cycles matrix C is:

$$C = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The non-zero elements in each row of C represent one cycle. The cycles are given below:

Cycle1={e_1, e_2, e_3};

Cycle2={e_2, e_4, e_5};

Cycle3={e_4, e_6, e_7};

Cycle4={e_8, e_9, e_{15}};

Cycle5={$e_5, e_8, e_9, e_{11}, e_{12}, e_{13}, e_{14}$};

Cycle6={$e_{11}, e_{12}, e_{13}, e_{16}, e_{17}, e_{18},$};

Cycle7={e_{18}, e_{19}, e_{20}};

6.4.2 Generating all spanning trees

Selecting one edge from each cycle, such that no edge is repeated, generates a co-spanning tree. Deleting the co-spanning tree edges from the incidence matrix, A, generates a spanning tree. For example, Figure 6.3 is a co-spanning tree of the IEEE-14 network.

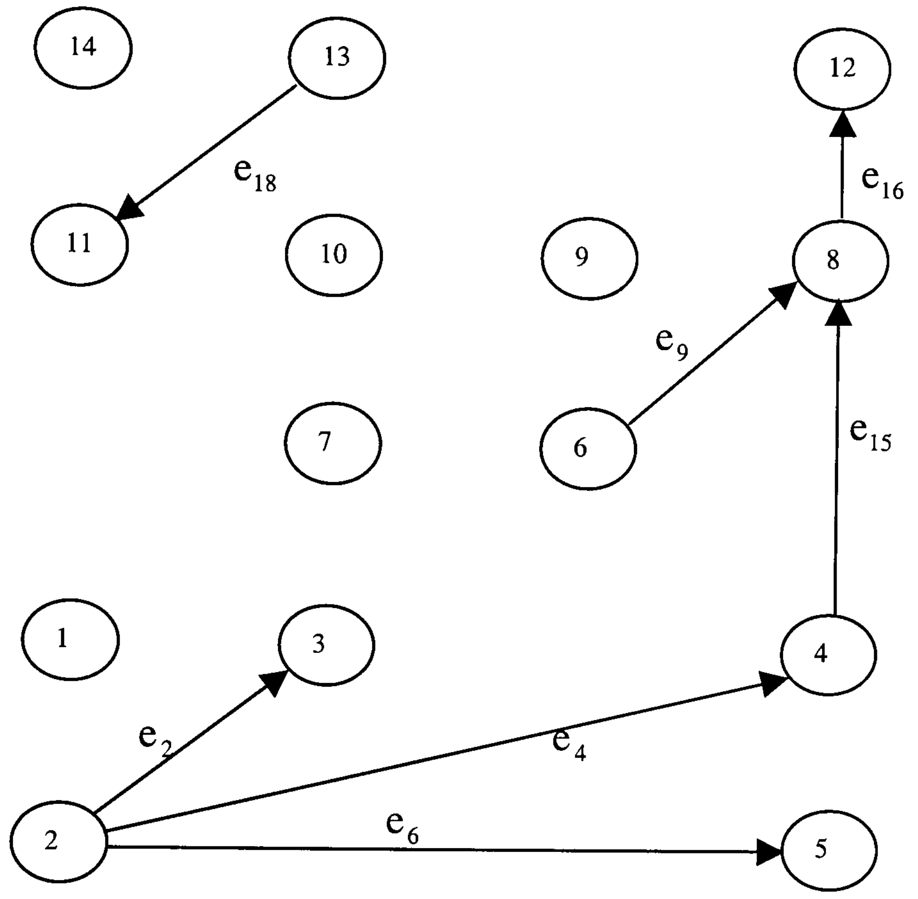


Figure 6.3. A co-spanning tree

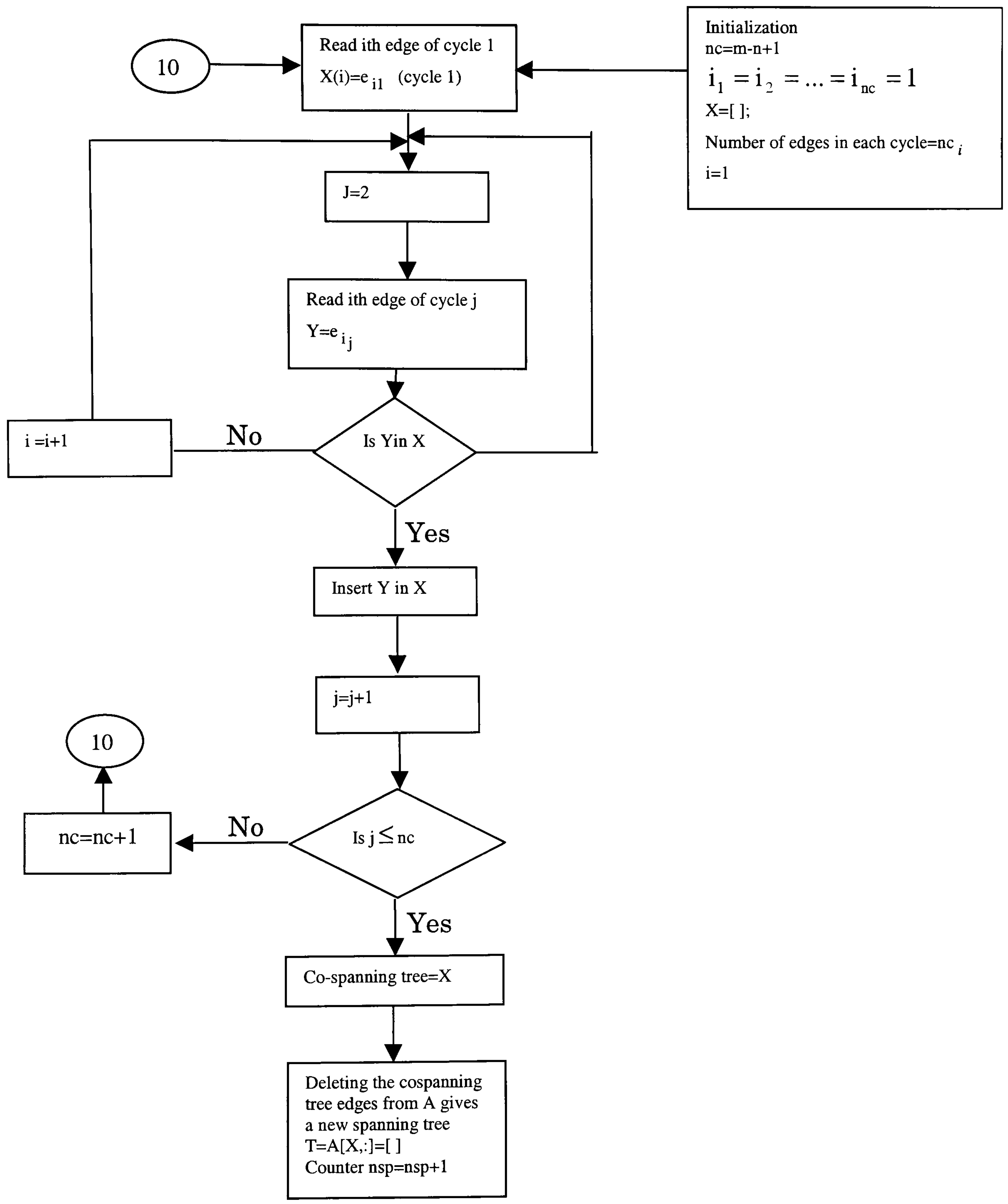
The set of the eliminated edges is:

$$\{, e_2, e_4, e_6, e_9, e_{15}\ e_{16}, e_{18}\};$$

The corresponding spanning tree is shown in Figure 6.2. The set of edges of the spanning tree is:

$$\{e_1, e_3, e_5, e_7, e_8, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{17}, e_{19}, e_{20}\}$$

Changing any one of the co-spanning tree edges without repetition gives a new co-spanning tree; deleting the co-spanning tree edges from A generates a new spanning tree. The generation of all possible spanning trees is given in Flowchart 6.1.



Flowchart 6.1 Generating a new spanning tree

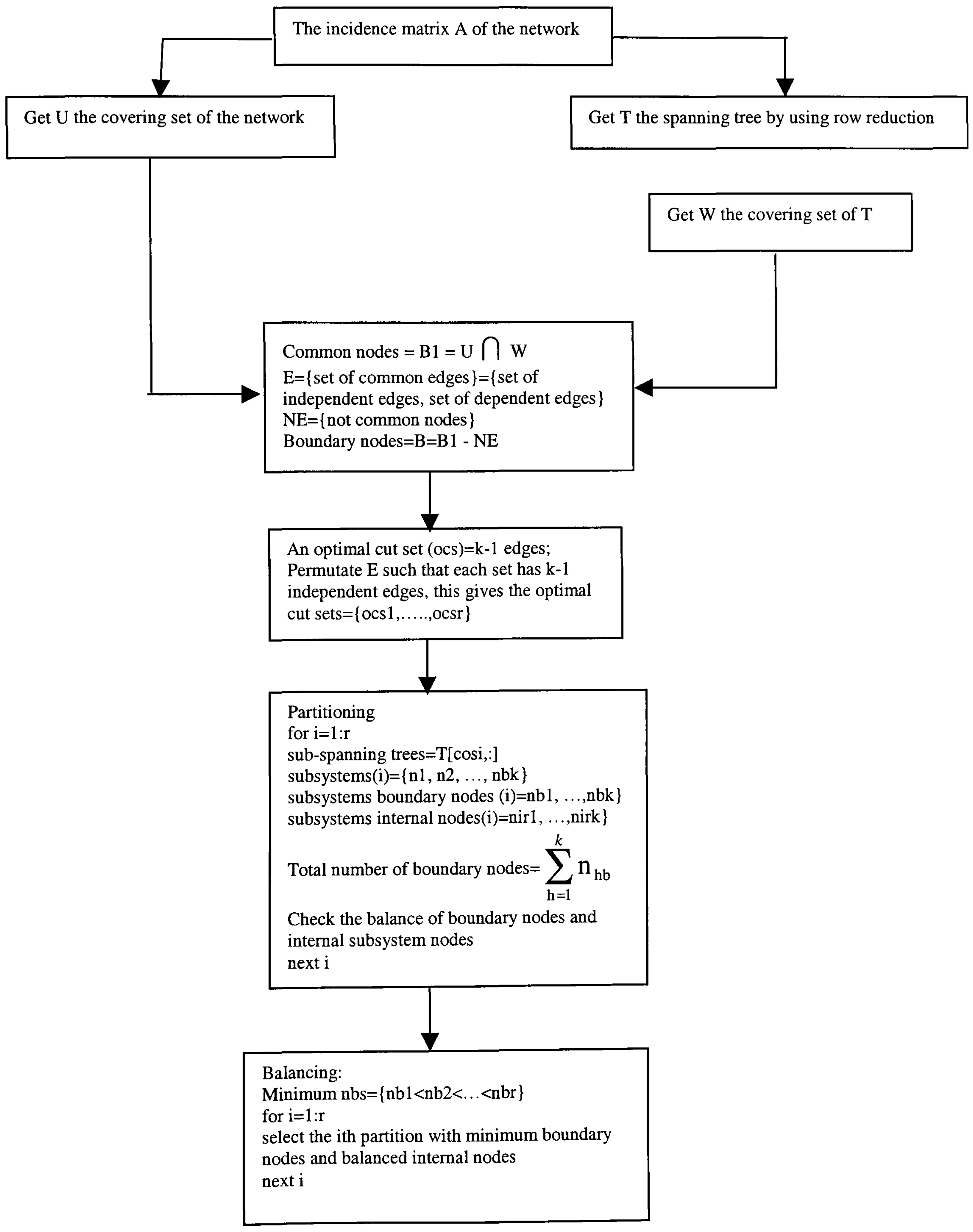
6.5 The maximum nodal degree technique

A method is now described for automatic partitioning of firstly, a spanning tree and secondly, the network from which it has been derived. The technique is based on using the covering set principle to find the minimum covering set of nodes, with highest degree in the network and in a spanning tree of the network. Thus the technique is called the maximum nodal technique. Starting from the identification of the node(s) with highest degree, the network nodes are listed by degree in descending order. This list is then searched to obtain a set of nodes, minimum in number but of highest degree, which covers the network. The covering set is then used to partition the network. The maximum nodal technique is illustrated in Flowchart 6.2 with reference to the IEEE-14 network.

The covering set principle is introduced in Section 6.5.1. Then, an algorithm to find the minimum covering set with highest nodal degree in a network such as the IEEE-14 network is described in Section 6.5.2. The algorithm of finding the minimum covering set with highest nodal degree in a spanning tree of a network is described in Section 6.5.3. Finding the minimum global boundary nodes in the network and the set of edges between them is described in Sections 6.5.4 and 6.5.5.

6.5.1 The covering set principle

The set of nodes connected directly with a node v is termed the neighbouring nodes of node v. In this case, node v is said to be the **covering node** and the set of neighbouring nodes is termed **the covered set of nodes**. Thus, **the network covering set** is a subset U of nodes such that all the network nodes are covered, i.e. directly connected to the nodes in U. $|U|$ denotes the number of elements (i.e. nodes) in U. The network has more than one covering set. A covering set is a minimum covering set if $|U| \leq |U'|$, where U' is


The incidence matrix A of the network
Get U the covering set of the network
Get T the spanning tree by using row reduction
Get W the covering set of T
Common nodes = B1 = U $\bigcap$ W
E={set of common edges}={set of independent edges, set of dependent edges}
NE={not common nodes}
Boundary nodes=B=B1 - NE
An optimal cut set (ocs)=k-1 edges;
Permutate E such that each set has k-1 independent edges, this gives the optimal cut sets={ocs1,.....,ocsr}
Partitioning
for i=1:r
sub-spanning trees=T[cosi,:]
subsystems(i)={n1, n2, ..., nbk}
subsystems boundary nodes (i)=nb1, ...,nbk}
subsystems internal nodes(i)=nir1, ...,nirk}
Total number of boundary nodes= $\sum_{h=1}^{k} n_{hb}$
Check the balance of boundary nodes and internal subsystem nodes
next i
Balancing:
Minimum nbs={nb1<nb2<...<nbr}
for i=1:r
select the ith partition with minimum boundary nodes and balanced internal nodes
next i

Flowchart 6.2 Partitioning and Balancing any covering set of G. In this case, |U| is termed **the node covering number of G**.

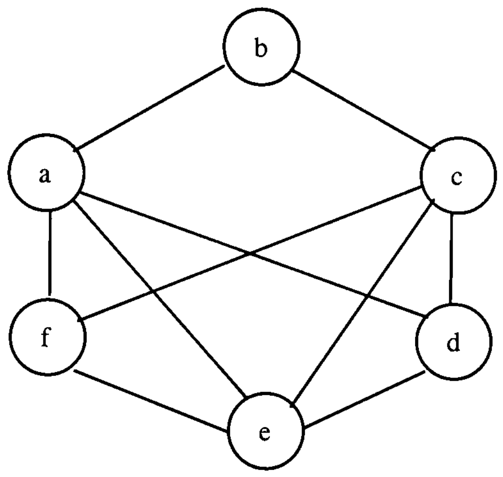


Figure 6.4
A network to illustrate the covering set principle

For example, in Figure 6.4

Node a covers {b, d, e, f};
Node b covers {a, c};
Node c covers {b, d, e, f};
Node d covers {a, c, e};
Node e covers {d, c, a, f};
Node f covers {a, c, e}.

The set {a, c} covers {b, d, e, f} only, and it does not cover nodes a and c. Thus the set {a, c} does not cover all the network nodes.
The set {a, c, f} covers {a, b, c, d, e, f}. Thus, the set {a, c, f} covers the network.
The following are different covering sets: set {a, c, e}, {a, c, e, f}, {b, d, e, f}, and {a, c, e}. Of these, {a, c, f} and {a, c, e} are the minimum covering sets; thus, the node covering number is 3. Of those two sets, the set {a, c, e} has maximum degree. Thus, the set {a, c, e} is the minimum covering set with maximum degree.

6.5.2 An algorithm to find the minimum covering set in the network

The maximum node degree technique is based on finding a subset of nodes U with highest possible degree such that this subset U is a covering set. Since the number of possible covering sets increases with network size, the method used for computation of the minimum covering set is crucial to the practical usefulness of such techniques. A new, fast algorithm for computation of the minimum covering set is now described with reference to the IEEE-14 network. Table (6.1) shows the IEEE-14 network nodes and the degree of each node.

Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Degree	2	4	4	5	2	3	1	4	2	2	4	2	3	2

Table 6.1 Node-degrees

Reorder table (6.1) in a descending order such that the node with maximum degree is first and the node with minimum degree is last. Table (6.2) represents table (6.1) after reordering.

Degree	5	4	4	4	4	3	3	2	2	2	2	2	2	1
v_i	4	2	3	8	11	6	13	1	5	9	10	12	14	7
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Table 6.2 Ordered node-degrees

Let U_i and Y_i be variable sets indexed by i, with U_0=[ ] and Y_0=[ ] (i.e. empty sets) respectively. X_i is an i-indexed set of neighboring nodes (i.e. connected directly by edges) associated with node v_i. U_i and Y_i are defined recursively by:

$$U_i = U_{i-1} \cup v_i; \qquad (6.1)$$

And

$$Y_i = Y_{i-1} \cup X_i(v_i); \qquad (6.2)$$

It is also necessary to define a test function q_i, defined by

$$q_i = \text{setdiff}(X_i(v_i), Y_{i-1}); \qquad (6.3)$$

which is a set of elements in $X_i(v_i)$ which are not in Y_i.

Flowchart 6.3 shows the algorithm of the maximum degree technique. The following steps describe how the maximum degree algorithm searches for the network minimum covering set, which has maximum degree, from Table 6.2:

Algorithm (6.1)

Let i=1, then

1- The algorithm starts by reading v_i and $X_i(v_i)$, the neighbouring nodes of v_i, from Table 6.2.

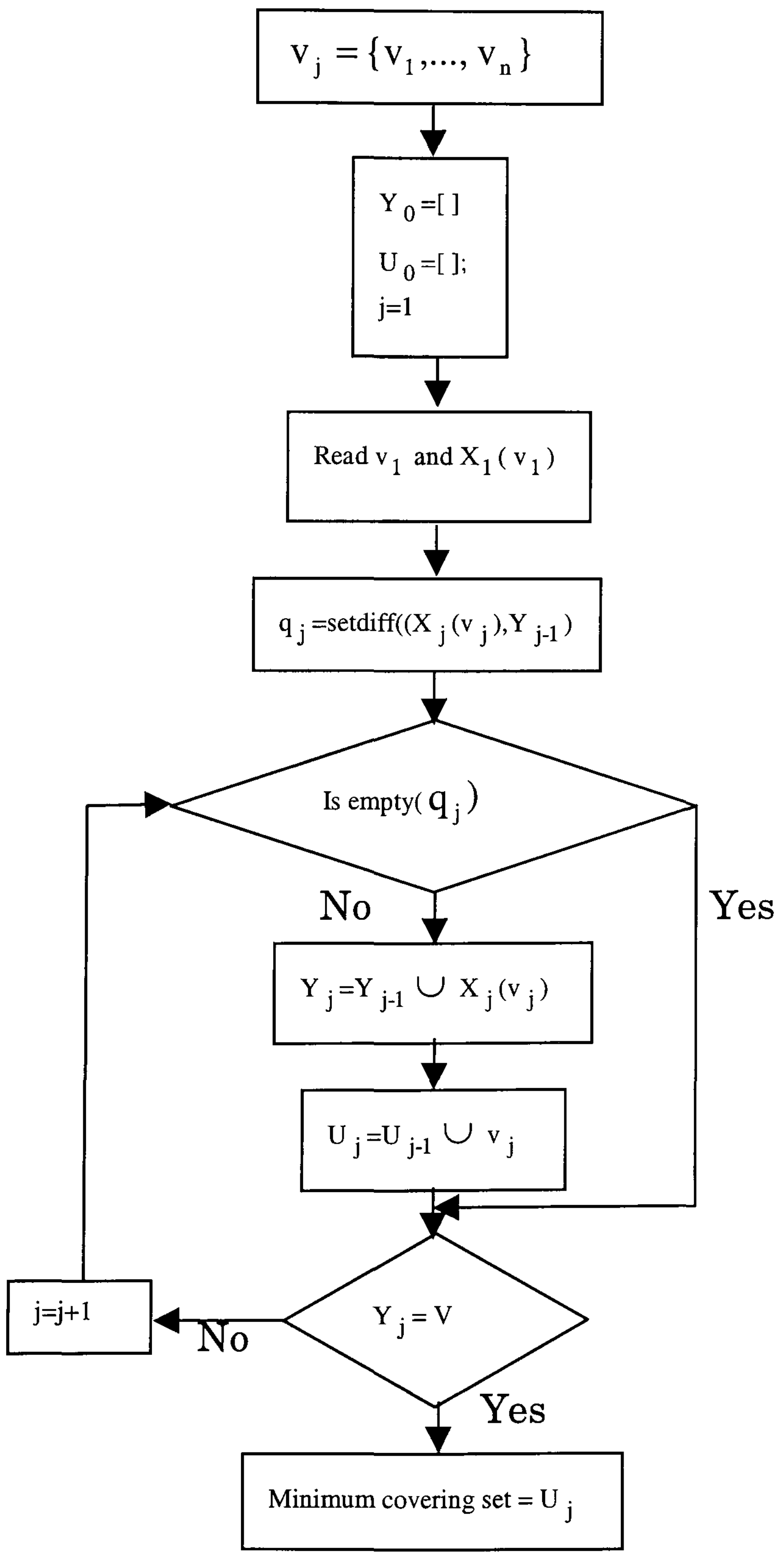
2- Then it uses equation (6.2) to check that q_i is not empty, i.e. if any of the neighbours of v_i is not in Y_{i-1}.

3- If q_i is empty, then the algorithm does not update Y_i, and it does not insert v_i into U_i.

4- If q_i is not empty, the algorithm uses equation (6.2) to update Y_i, and it uses equation (6.1) to insert v_i into U_i, the covering set.

5- After updating Y_i, the algorithm check if Y_i equals V, the set of network nodes.

6- If Y_i= V the algorithm terminates, else the algorithm update, i, i.e. i=i+1, and then repeat, the previous steps starting from step 1. At termination $|U_i| = i$.


$v_j = \{v_1, \ldots, v_n\}$
Y_0=[ ]
U_0=[ ];
j=1
Read v_1 and $X_1(v_1)$
q_j=setdiff(($X_j(v_j)$,Y_{j-1})
Is empty(q_j)
No
Yes
$Y_j=Y_{j-1} \cup X_j(v_j)$
$U_j=U_{j-1} \cup v_j$
$Y_j = V$
j=j+1
No
Yes
Minimum covering set = U_j

Flowchart 6.3. The minimum covering set algorithm

Table 6.3 shows the result of applying the maximum nodal algorithm for the IEEE-14 network.

i	degree	v_i	$X_i\ (v_i)$	q_i	Y_i	U_i	$Y_i = V$
1	5	4	2,3,5,6,8	[ ]	2,3,5,6, 8	4	No
2	4	2	1,3,4,5	1,4	1,2,3,4, 5,6,8	4,2	No
3	4	3	1,2,4,11	11	1,2,3,4, 5,6,8,11	4,2,3	No
4	4	8	4,6,9,12	9,12	1,2,3,4, 5,6,8,9, 11,12	4,2,3,8	No
5	4	11	3,10,13, 14	10,13, 14	1,2,3,4, 5,6,8,9, 10,11,12, 13,14	4,2,3,8,11	No
6	3	6	4,7,8	7	1,2,3,4, 5,6,7,8, 9,10,11, 12,13,14	4,2,3,8,11,6	Yes

Table 6.3 Searching sequence for the minimum covering set

Applying this algorithm to the IEEE-14 network gives the following covering subset U={2,3,4,6,8,11}. Finding each node takes three set operations. The maximum number of nodes the covering set may have in a network is (n/2) [69]. This method is the first step in reducing the partitioning problem size from n nodes to ≤ n/2 It uses set operations to find the minimum covering subset therefore it takes O (3n/2) operations.

6.5.3 An algorithm to find the minimum covering set in the spanning tree

Let W be the minimum covering set of nodes, with higher degrees, in the spanning tree such that all the spanning tree nodes are covered. It is possible to find W by using the maximum nodal algorithm given explained in Section 6.5.2 and by using Flowchart 6.3. The steps of obtaining the minimum covering set of the spanning tree of Figure 6.3 are similar to Algorithm (6.1). Nodes with highest degree are selected first. If two nodes have the same degree then the nodes order is used, i.e. the node with first order is selected first.

The steps are shown in Table 6.4. For example, W of the spanning tree of the IEEE-14 network shown in Figure 6.2, is:

W={1,3,4,6,8,9,11,14}

I	degree	v_i	$X_i\ (v_i)$	q_i	Y_i	W_i	$Y_i = V$
1	3	3	1,4,11	[ ]	1,4,11	3	No
2	3	4	3,5,6	3,5,6	1,3,4,5, 6,11	3,4	No
3	3	11	3,10,14	10,14	1,3,4,5, 6,10,11, 14	3,4,11	No
4	2	1	2,3	2	1,2,3,4,5, 6,10,11, 14	1,3,4,11	
5	2	6	4,7	7	1,2,3,4,5, 6,7,10,11, 14	1,3,4,6,11	No
6	2	8	9,12	9,12	1,2,3,4, 5,6,7,9, 11,12,14	1,3,4,6,8,11	No
7	2	9	8,10	7	1,2,3,4, 5,6,7,8, 9,10,11, 12,14		no
8	2	14	11,13	13	1,2,3,4,5, 6,7,8,9, 10,11,12, 13,14	1,3,4,6,8, 9,11,14	Yes

Table 6.4 Finding the minimum covering set in the spanning tree.

6.5.4 Finding the boundary nodes and the cut-edges

The intersection of the two subsets U and W gives a new subset B1 termed **the common node subset**, i.e.

B1=U $\cap$ W.

For example, from the IEEE-14 network

U={2,3,4,6,8,11};

and from the spanning tree of Figure 6.2,

W={1,3,4,6,8,9,11,14}

Then the intersection of the two sets gives the common nodes:

B1={3,4,6,8,11}.

In the network, the nodes of B1 are always connected, and they are not always connected in every spanning tree. The set of edges between the common nodes in the network is termed $E_{G\text{-common}}$, **the set of common-edges**. The set of $E_{G\text{-common}}$ edges contains the cut-edges and it may contain edges or branches other than the cut-edges.

Since cutting any branch or an edge of a branch results in at least one new boundary node that is not in B1, then all branches in $E_{G\text{-common}}$ are not cut-edges, thus they are deleted from $E_{G\text{-common}}$. The following example illustrates this step.

The set of $E_{G\text{-common}}$ between the B1 nodes in the IEEE-14 network given in Figure 6.1 is:

$E_{G\text{-common}}=\{e_{14}, e_5, e_{15}, e_8, e_9, \{e_{11}, e_{12}, e_{13}\}\}$.

Thus, the B1 nodes are connected to each other, by an edge or by a **branch**:

Between node 3 and node 11 there is one edge, i.e. e_{14}.

Between node 3 and node 4 there is one edge, i.e. e_5.

Between node 4 and node 8 there is one edge, i.e. e_{15}.

Between node 4 and node 6 there is one edge, i.e. e_8.

Between node 6 and node 8 there is one edge, i.e. e_9.

Between node 8 and node 11 there is one branch, i.e. $\{e_{11}, e_{12}, e_{13}\}$.

Since the branch is not a cut-edge, then deleting the branch from $E_{G\text{-common}}$ reduces $E_{G\text{-common}}$ to:

$E_{G\text{-common}} = \{e_{14}, e_5, e_{15}, e_8, e_9\}$.

As a result of eliminating (m-n+1) edges from the network, the B1 nodes **in the spanning tree** are not always connected. The unconnected nodes are deleted from B1. The remaining nodes in B1 are connected. Let B be the subset of connected nodes in B1, and let the set of common edges between the B nodes, in the spanning tree, is termed $E_{T\text{-common}}$.

For example, B1={3,4,6,8,11},

The following nodes are connected to each other:

Between node 3 and node 4 there is one edge, i.e. e_5;

Between node 3 and node 11 there is one edge, e_{14};

Between node 4 and node 6 there is one edge, i.e. e_8.

Between node 8 and node 11 there is one branch, i.e. $\{e_{11}, e_{12}, e_{13}\}$.

Then the nodes of B1 in the spanning tree of Figure 6.2 are connected.

Thus,

B=B1= {3, 4, 6, 8, 11}.

And the edges between the B nodes are

$E_{T\text{-common}} = \{e_{14}, e_5, e_8, \{e_{11}, e_{12}, e_{13}\}\}$.

The nodes of B are the global boundary nodes of the spanning tree of Figure 6.2 V_b = B = B1={3,4,6,8,11}.

. They are connected to each other in the spanning tree, and their number is

minimum. The number of nodes of the set B is n_b, the minimum number of global boundary nodes. The value of n_b in this case is $n_b = 5$.
Deleting the branches from $E_{T\text{-common}}$ reduces $E_{T\text{-common}}$ to:

$$E_{T\text{-common}} = \{e_{14}, e_5, e_8\}.$$

The edges of $E_{T\text{-common}}$ may be dependent or independent. For example, from the spanning tree in Figure 6.2, the edges e_5 and e_{14} are dependent and nodes 3 is the boundary node between them. The edges e_5 and e_8 are dependent and the boundary node between them is node 4. The edges e_{14} and e_8 are independent.

The number of cut-edges that partitions the spanning tree into k sub-spanning trees is (k-1). Thus for k=2, then the number of cut-edges=1, and if k=3, the number of cut edges equals 2. The maximum degree technique cuts (k-1) edges from $E_{T\text{-common}}$ in turn until the balanced partitioning is achieved. For example, for k=2, the number of cut-edges = 1. Then each one of the $E_{T\text{-common}}$ edges will be cut in turn, i.e. e_{14}, e_5 and e_8.

The value of n_b, obtained by using the maximum degree method, is in the acceptable range as defined by equation (5.22). This characteristic is used to obtain the range of k, which balances the k parts.

Since networks have different sizes and topology, generating all their spanning trees is not a practical solution. Therefore, Procedure 6.1, described below, is to narrow the band of selecting the spanning tree, and to determine n_b and the range of k. The procedure is based on using the intersection of the minimum covering set of the network and the minimum covering set of a spanning tree as a threshold for the minimum global boundary nodes.

Procedure (6.1):

```
     Initialization: k=2;
     Compute U;          The network minimum covering set.
   5 Obtain a new T;     A new spanning tree
     Compute W; The spanning tree minimum covering set
     $n_b = U \cap W$;
     $n_r = n - n_b$;
  10 $n_p = \left\lfloor \frac{n_r}{k} \right\rfloor$;
     If $(n_b = n_p)$ or $(n_b = n_p + 1)$
          $f_k = n_r - k.n_p$;
          If $f_k = 0$, then compute the equal partitions
               $n_{ir} = n_p$; for $i = 1, \cdots, k$;
          elseif $f_k > 0$, then compute the balanced partition
               $n_{ir} = n_p$; for $i = 1, \cdots, k - f_k$;
               $n_{ir} = n_p$; for $i = k - f + 1, \cdots, k$;
          end;
          save (k, $n_b$, $n_{ir}$; for $i = 1, \cdots, k$;);
          k=k+1;
          go to 10;
elseif $n_b > (n_p + 1)$
Go to 5;
End.
```

6.5.5 Finding the k subsystems internal nodes

Cutting one of the cut-edges disconnects the spanning tree into two sub-spanning trees.

The cut operation is performed in the spanning tree matrix T by deleting the set of cut-edges from T (i.e. by setting the rows of the cut-edges equal to zero). The resultant k sub-spanning trees are connected. The nodes of each sub-spanning tree equal the union of the end-nodes of the edges of the sub-spanning tree. The nodes of each sub-spanning tree represent a subsystem. For example, if k=2, then the $E_{T\text{-common}} = \{e_{14}, e_5, e_8\}$ will be cut in turn.

Let $V_b = \{3,4,6,8,11\}$.

If e_5 is cut, then

$$E_1 = \{e_1, e_3, e_{11}, e_{12}, e_{13}, e_{14}, e_{16}, e_{19}, e_{20}\};$$

the end-nodes are

$$V_1 = \{v_1, v_2, v_3, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}\};$$

$$V_{1b} = V_b \cap V_1 = \{v_3, v_8, v_{11}\};$$

V_{1r} is obtained by deleting V_{1b} from V_1, i.e.

$V_{1r} = \{v_1, v_2, v_9, v_{10}, v_{12}, v_{13}, v_{14}\}$; and $n_{1r} = |V_{1r}| = 7$

and

$$E_2 = \{e_7, e_8, e_{10}\};$$

$$V_2 = \{v_4, v_5, v_6, v_7\};$$

$$V_{2b} = V_b \cap V_2 = \{v_4, v_6\};$$

Also V_{2r} is obtained by deleting V_{2b} from V_2, then

$V_{2r} = \{v_5, v_7\}$; and $n_{2r} = |V_{2r}| = 2$

The results of cutting e_5, (i.e. $n_{1r} = 7, n_{2r} = 2$ and $n_b = 5$) are unbalanced partitioning values. Thus e_5 is not accepted by the maximum degree technique as a cut-edge.

The maximum degree technique cuts the next edge in $E_{T\text{-common}}$.

If e_{14} is cut, then

$$E_1 = \{e_1, e_3, e_5, e_7, e_8, e_{10}\};$$

$$V_1 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\};$$

$$V_{1b} = V_b \cap V_1 = \{v_3, v_4, v_6\};$$

Then V_{1r} is obtained by deleting V_{1b} from V_1, then

$V_{1r} = \{v_1, v_2, v_5, v_7\}$; and $n_{1r} = |V_{1r}| = 4$;

and

$E_{SB\text{-}2} = \{e_{11}, e_{12}, e_{13}, e_{16}, e_{19}, e_{20}\}$;

$V_2 = \{v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}\}$

$V_{2b} = V_b \cap V_2 = \{v_8, v_{11}\}$;

Also V_{2r} is obtained by deleting V_{2b} from V_2, then

$V_{2r} = \{v_9, v_{10}, v_{12}, v_{13}, v_{14}\}$; and $n_{2r} = |V_{2r}| = 5$.

The results of cutting e_{14}, (i.e. $n_{1r} = 4, n_{2r} = 5$ and $n_b = 5$) are balanced partitioning values. Thus e_{14} is accepted by the maximum degree technique as a cut-edge.

After finding the balanced partitioning values, the maximum partitioning technique terminates.

6.6 Simulation Results

The maximum degree technique has been applied to partition the spanning trees of the IEEE-14 network. The IEEE-14 network has 3909 different spanning trees. The technique finds the possible range of k and the corresponding value of n_b that satisfies the DSE restrictions. The technique partitions the spanning trees by deleting (k-1) edges from the common edges between the B nodes in each spanning tree. Once a spanning tree has been partitioned for a given k, then n_b the number of boundary nodes and n_{ir} the number of internal nodes, for i=1,2,...,k, are easily obtained. For each k, the simulation results are compared with the theoretical balanced partitioning values obtained in section 5.6.

Simulation results for k=2

Partitioning all the 3909 different spanning trees for k=2 gives the following results: Out of the 3909 spanning trees, 413 spanning trees gave n_b =4., 1304 spanning trees gave n_b =5 and 1208 spanning trees gave n_b =6. The remaining spanning trees gave n_b >> 6.

1. Case n_b = 4

The topology of the IEEE-14 network is such that none of partitioned spanning trees give $n_b = 4$ with balanced internal nodes.

2. Case $n_b = 5$

The maximum degree partitioning technique gave 1304 spanning trees that gave $n_b = 5$. Of the 1304 spanning trees, 514 spanning trees gave balanced n_{ir} for i=1,2 as shown in Figure 6.5.

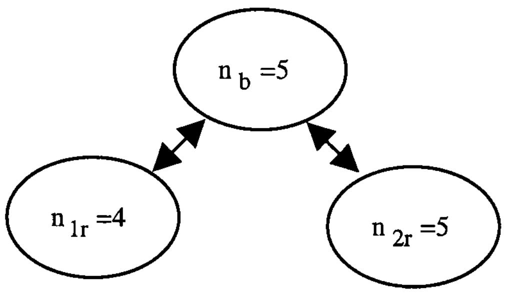


Figure 6.5 Simulation balanced results

Comparing the simulation and the theoretical results

1. The simulation result of n_b is $n_b=4$. This simulation value, i.e. $n_b=4$, is in the acceptable range of n_b given by (5.17). It equals the theoretical result n_{bh}.

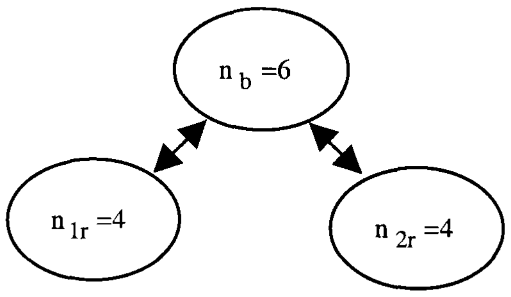
2. The simulation values of n_{ir} for i=1,2 are balanced and they equal the theoretical values.

3. Thus, These simulation results, i.e. n_{bu} and n_{ir} for i=1,2, do satisfy the DSE restrictions (5.16) and (5.28). Therefore, the balanced partitioning values that satisfy the DSE restrictions exit in the IEEE 14-bus network for k=2 and $n_b=5$.

3. Case $n_b = 6$

The next n_b, in the simulation results, equals $n_b=6$. The total number of spanning trees, which gave this value, is 1208.

2. The simulation results, of the balanced and unbalanced subsystems internal nodes, are shown in Figures 6.6a and 6.6b.



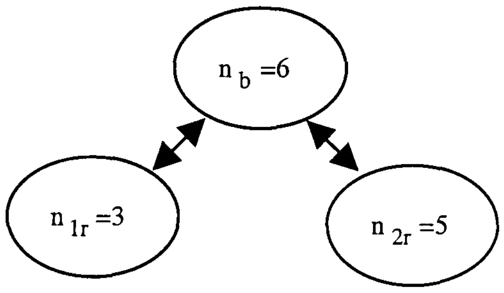


Figure 6.6a Figure 6.6b

The unbalanced results when $n_b = 6$

Comparing the simulation and the theoretical results

1. This value of $n_b = 6$ is not in the theoretical range and it does not fulfill the DSE restrictions (5.17).
2. Both internal nodes simulation results do not satisfy restrictions (5.11), (5.12) and (5.17) of the DSE restrictions.

3. This n_b does not satisfy the global balance restrictions of the DSE. Therefore $n_b = 6$ is not acceptable by the DSE.

All simulation results of $n_b > 6$ do not satisfy restriction (5.14), therefore they are not acceptable by the DSE.

6.7 Chapter review

The major conclusion in this chapter is the application of Theorem (5.1). It is the forward step to simplify the network-partitioning problem. By using Theorem (5.1) it is possible to know, without using any partitioning technique, the balanced partitioning values of any given network for a given k. It leads to further investigation of the network properties to find the number of cut edges.

A network has many different covering sets of nodes. The algorithm of the maximum degree partitioning technique gives a very fast way of finding the minimum covering set in the network and in a spanning tree. The technique can avoid the sensitivity of the spanning tree, i.e. the intersection of the minimum covering set of the network with the minimum covering set of a range of the network spanning trees gives the minimum global boundary nodes that satisfy the DSE restrictions and the ideal balanced partitioning conditions.

Using the network cycles and the set operations speeds up the process of obtaining and partitioning different spanning trees.

Chapter 7

The edge state phenomenon

7.1 General

The concept of partitioning a network into small-sized sub-networks has been introduced in Chapter 1. The partitioning techniques introduced in Chapters 3 and 4 can be classified as heuristic methods to solve the partitioning problem. To date, most of the partitioning techniques presented in the literature have heuristic bases [13, 16, 22]. The reason behind using heuristic methods is that the existing graph theory is not yet complete. Many of the graph properties are not known.

For example, the traditional definition of a graph G=(V, E) of a network, is that it consists of two sets: a finite set V of elements called nodes and a finite set E of elements called edges. A pair of nodes terminates each edge. Only a few natural relationships between V and E have been derived, for example the sum of the degrees of all nodes equals twice the number of edges, and the relationship between the number of cycles and the numbers of edges and nodes. Also many important relationships have been derived to solve problems other than these relating to the graph characteristics and relationships, such as the cut concept, Hamiltonian Graphs, connectivity, matching and colouring.

Naturally, a connected network has n nodes connected by m edges. The network may have cycles and it may not. A node, in the connected network, may be of one degree or of two degrees or of more degrees. An edge, in the same network, may be connected to a node of one degree or to a node of two degrees or to a node of more degrees, and it may form, with other edges, a cycle and it may not. This is the natural state of a network with its nodes, edges and cycles.

Knowing the natural state of the network and its nodes edges and cycles, in graph theory, there are no relations that distinguish and specify between the edges and the nodes as they exist naturally in the network. The non-availability of these precise definitions and relationships are sufficient not only for using heuristic methods to solve related problems, such as the network partitioning problem, but also to classify it as NP-Hard [25].

Having stated that, in this chapter, a natural phenomenon that does exist in every graph is introduced. The phenomenon is that if an edge is in a network, in the two-dimensional plane, then the edge must belong to a zero cycle or one cycle or two cycles. The phenomenon is also valid in the planer graph.

The phenomenon leads to a new approach of studying a graph of a network and classifying its edges, nodes and cycles accordingly, and exploring some of the natural laws and relationships which govern the graph. The phenomenon will be named "**The Edge State Phenomenon**".

In this chapter, the new approach "**The edge state phenomenon** " is introduced.

7.2 Types of network edges

When a network is partitioned into k sub-networks certain edges will be cut, the remaining edges not being cut. A cut edge may belong to one cycle, or may be a common edge between two cycles, or may not belong to a cycle. In this section, a classification of the network edges is introduced.

In graph theory, an edge is defined as a line connecting two nodes. A node with no edges is termed an isolated node and its degree equals zero. The degree of a node with one edge is one, thus the node is termed a one-degree node. A cycle is defined as the shortest closed loop (i.e. the closed-loop contains the least number of edges) starting from a node and ending at the same node. Every cycle has three edges or more. The network consists of those nodes and edges.

Accordingly, some of the network edges are connected to one-degree nodes, some belong to one cycle only, some are between two cycles and others do not belong to any cycle. This natural phenomenon is used to classify the network edges as follows: If an edge is connected to a one-degree node, then it is termed a one-degree edge. If an edge belongs to only one cycle, then it is termed an external edge. If an edge is a common edge between two cycles, then it is termed an internal edge. If an edge does not belong to any cycle and not connected to a one-degree node, then it is termed a bridge edge.

Let E_S be the set of one-degree edges in the network, and let m_S be the number of one-degree edges in E_S, i.e.

$$m_S = |E_S|. \tag{7.1}$$

Let E_I be the set of internal edges in the network, and let m_I be the number of internal edges in E_I, i.e.

$$m_I = |E_I|. \tag{7.2}$$

Let E_X be the set of external edges in the network, and let m_X be the number of external edges in E_X, i.e.

$$m_X = |E_X|. \tag{7.3}$$

Let E_B be the set of bridge edges in the network, and let m_B be the number of bridge edges in E_B, i.e.

$$m_B = |E_B|. \tag{7.4}$$

The total number of edges in the network, m, can be written as the sum of the four types, (7.1) to (7.4):

$$m = m_S + m_I + m_X + m_B. \tag{7.5}$$

For example, in the network of Figure 7.1,

$m_S = m_I = m_B = 1$, and

$m_X = 8$, so that, for equation (7.5),

$m = 1 + 1 + 1 + 8 = 11$.

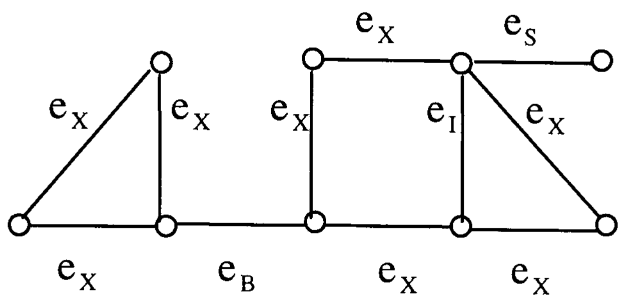


Figure 7.1 A network to illustrate the edge types

7.3 Types of network nodes

Based on the classification of network edges, the network nodes can similarly be classified into those four types or combinations of them.

7.3.1 General

If a node has only one type of edge then the node type is classified according to the type. If a node has two types or more, then the node is classified as a combination of these types. In Section 7.3.2 the combined nodes types precedence in classification is explained.

A node may or may not belong to a cycle. If it belongs to a cycle then the node is either an external node or an internal node. If it does not belong to a cycle the node is either a bridge node or a one-degree node.

If no external nodes exist then no internal nodes exist, but if external nodes exist then it is not necessary that internal nodes should exist. Therefore the external type has higher precedence than internal type. The same principle is true with the bridge edges and the one-degree edges. Therefore, if a node has the four types, then the external type always has highest precedence in ordering, the bridge type next highest, then the internal type, and lastly the one-degree type.

If the node does not belong to a cycle then the node is either a bridge node or a one-degree node. The one-degree node always has one one-degree edge, while the bridge node has at least two edges. The other end of the one-degree edge can be connected to a bridge node, an internal node or an external node. The other end of the bridge edge is always connected to an external node or a bridge node but never to an internal node. Thus, in a network with cycles, the external node has higher precedence than the bridge node. Since there is no edge between the bridge node and the internal node and since both of them come after the external node in the precedence order, then the bridge node and the internal node both have the second order in precedence after the external node, then the one-degree node.

The two edges are either two bridge edges, or one bridge edge and one one-degree edge. Therefore, the bridge node can have a one-degree edge, but it does not take an external edge or an internal edge. Therefore, the bridge type precedes the one-degree type.

7.3.2 Node-type definitions

The following types of node are defined:

(i) S node:

A node is termed S node (or one-degree node) if

(1) It does not belong to a cycle,

(2) It has only one edge.

Let the set of S nodes be denoted V_S, and let $n_S = |V_S|$ be the number of elements in V_S. In Figure 7.2 $V_S = \{v_2, v_{12}\}$, and $n_S = 2$.

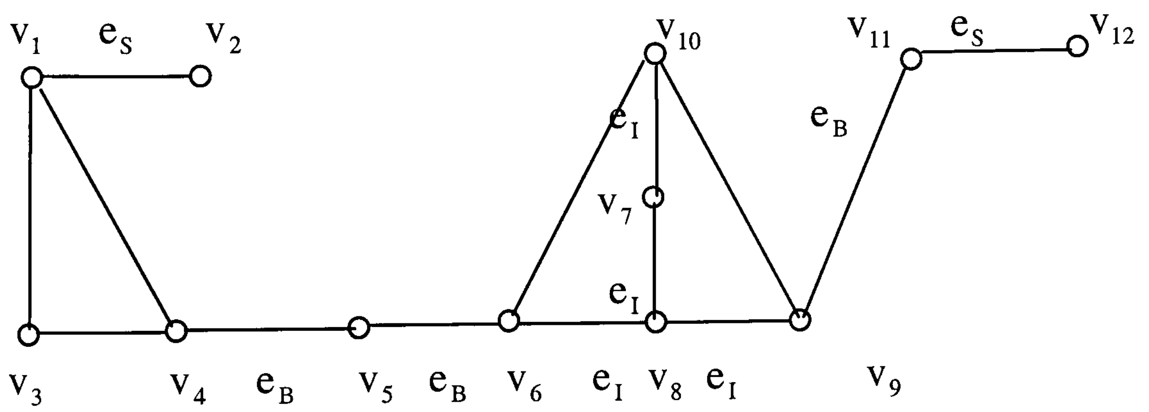


Figure 7.2 To A network to illustrate the nodes types

(ii) B node:

A node is termed a B node (or a bridge node) if

(1) It does not belong to a cycle,

(2) Its degree is two or more,

(3) At least one of its edges is of a bridge type.

Let the set of B nodes be denoted V_B, and let $n_B = |V_B|$ be the number of elements in V_B. In Figure 7.2 $V_B = \{v_5, v_{11}\}$, and $n_B = 2$.

(iii) I node:

A node is termed an I node (or internal node) if

(1) Its degree is more than one,

(2) The number of cycles it has equals its degree.

(3) All its edges are of internal type.

Let the set of I nodes be denoted V_I, and let $n_I = |V_I|$ be the number of elements in V_I. In Figure 7.2 $V_I = \{v_7\}$, and $n_I = 1$.

A node is termed an external node if all edges are external. Every external node has an even number of external edges. If an external node has two external edges, then it belongs to one cycle. If the external node has four edges, then it belongs to two cycles; and so on. Therefore, an external node is classified according to its degree and the number of cycles it belongs to. This leads to definition (iv).

(iv) Xd node:

A node is termed an Xd node (or external node of degree d), where $d = 2, 4, \cdots$, if

(1) It belongs to d/2 cycles,

(2) Its degree equals d,

(3) The d edges are of external type,

Let the set of Xd nodes, for particular even values of d, be denoted V_{Xd}, and let $n_{Xd} = |V_{Xd}|$ be the number of elements in V_{Xd}.

Define a set V_{XD} of nodes as the union of the Xd sets for $d = 2,4,\cdots$.

Then, $V_{XD} = \{V_{X2} \oplus V_{X4} \oplus \cdots\}$, and let $n_{XD} = |V_{XD}| = n_{X2} + n_{X4} + \cdots$.

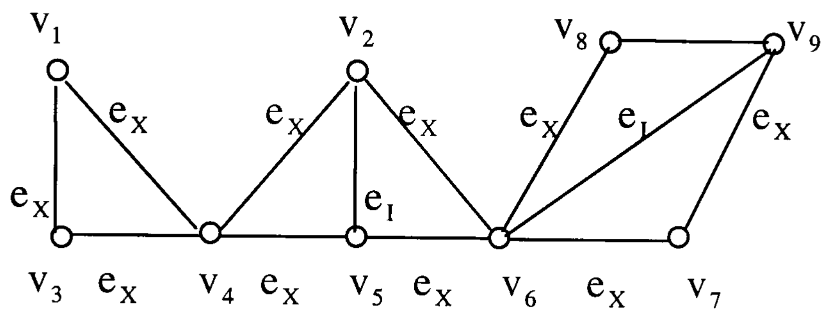


Figure 7.3 A network to illustrate the Xd nodes

In Figure 7.3, the Xd sets are:

$V_{X2} = \{v_1, v_3, v_7, v_8\}$, with $n_{X2} = 4$,

$V_{X4} = \{v_4\}$, with $n_{X4} = 1$,

and set V_{XD} is:

$V_{XD} = \{V_{X2}, V_{X4}\} = \{v_1, v_3, v_4, v_7, v_8\}$, with

$n_{XD} = 4 + 1 = 5$.

Note that the number of cycles at an Xd node equals d/2.

In addition to Xd nodes, a network may contain nodes that have two or more external edges, together with edges of other types. This gives rise to the need for the following additional definitions:

(v) XdI node:

A node is termed an XdI node (or external-internal node of degree d) where $d = 2,4,\cdots$, if

(1) It belongs to more than one cycle,

(2) Its degree is greater than d,

(3) It has d external edges,

(4) It has at least one internal edge.

Let the set of XdI nodes, for particular even values of d, be denoted V_{XdI}, and let $n_{XdI} = |V_{XdI}|$ be the number of elements in V_{XdI}.

Define a set V_{XDI} of nodes as the union of the XdI sets for $d = 2,4,\cdots$.

Then, $V_{XDI} = \{V_{X2I} \oplus V_{X4I} \oplus \cdots\}$, and let $n_{XDI} = |V_{XDI}| = n_{X2I} + n_{X4I} + \cdots + n_{XdI}$. In Figure 7.3, the XdI sets are:

$V_{X2I} = \{v_2, v_5, v_9\}$ with $n_{X2I} = 3$,

$V_{X4I} = \{v_6\}$ with $n_{X4I} = 1$,

and the V_{XDI} is

$V_{XDI} = \{V_{X2I} \oplus V_{X4I}\} = \{v_2, v_5, v_6, v_9\}$, with

$n_{XDI} = 4$.

(vi) XdB node:

A node is termed an XdB node (or external bridge node of degree d) where $d = 2,4,\cdots$ if

(1) It belongs to d/2 cycles,

(2) Its degree is greater than d

(3) It has d external edges.

(4) It has at least one bridge edge.

Let the set of XdB nodes, for particular value of d, be denoted V_{XdB}, and let $n_{XdB} = |V_{XdB}|$ be the number of elements in

Define the set V_{XDB} of nodes as the union of the XdB sets for $d = 2,4,\cdots$.

Then, $V_{XDB} = \{V_{X2B} \oplus V_{X4B} \oplus \cdots\}$, and let $n_{XDB} = |V_{XDB}| = n_{X2B} + n_{X4B} + \cdots$.

In Figure 7.4 the set of XdB nodes are:

$V_{X2B} = \{v_7\}$ with $n_{X2B} = 1$,

$V_{X4B} = \{v_8\}$ with $n_{X4B} = 1$.

The set V_{XDB} is

$V_{XDB} = \{V_{X2B} \oplus V_{X4B}\} = \{v_7, v_8\}$ with

$n_{XDB} = 2$.

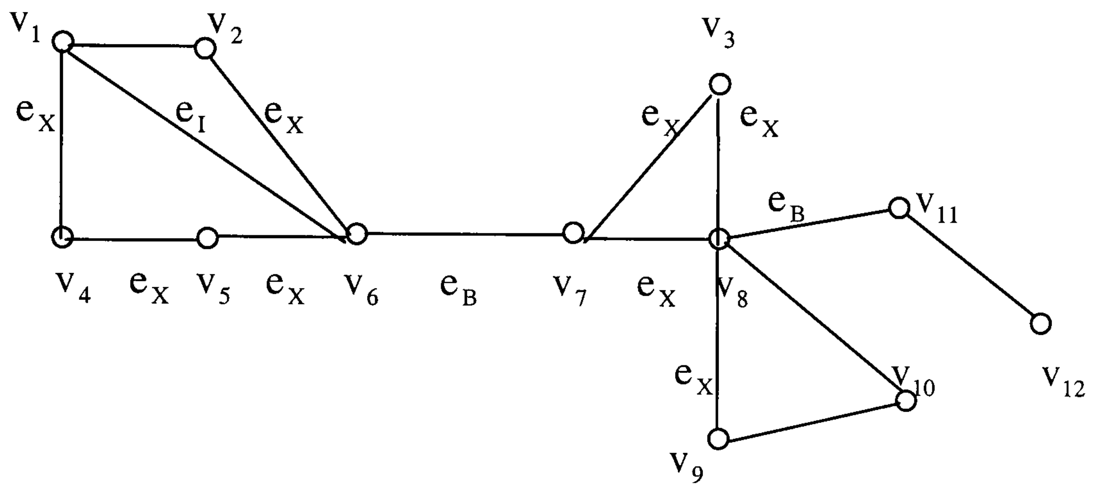


Figure 7.4 A network to illustrate the XdB nodes

(vii) XdBI node:

A node is termed an XdBI if:

(1) It belongs to more than d/2 cycle,

(2) Its degree is greater than d,

(3) It has d external edges.

(4) It has at least one bridge edge and at least one internal edge

Let the set of XdBI nodes, for a particular value of d, be denoted V_{XdBI}, and let $n_{XdBI} = |V_{XdBI}|$ be the number of elements in V_{XdBI}.

Define the set V_{XDBI} of nodes as the union of the XdBI sets for $d = 2, 4, \cdots$.

Then, $V_{XDBI} = \{V_{X2BI} \oplus V_{X4BI} \oplus \cdots\}$, and let $n_{XDBI} = |V_{XDBI}| = n_{X2BI} + n_{X4BI} + \cdots$.

In Figure 7.4, the set of XdBI is:

$V_{X2BI} = \{v_6\}$ with $n_{X2BI} = 1$.

The set of XDBI nodes is

$V_{XDBI} = \{ V_{X2BI}\} = \{v_6\}$ with

$n_{XDBI} = 1$.

7.3.3 The network node equation

Let V_X represents the union of all sets of nodes having external nodes, and let n_X represent the total number of elements in V_X. Since any node in V_X is a member of one and one only of the sets $\{V_{XD}, V_{XDI}, V_{XDB}, V_{XDBI}\}$, then

$$V_X = \{V_{XD} \oplus V_{XDI} \oplus V_{XDB} \oplus V_{XDBI}\}; \tag{7.6}$$

and $$n_X = n_{XD} + n_{XDI} + n_{XDB} + n_{XDBI}; \tag{7.7}$$

Let V represents the union of all sets of nodes in the network, and let n represent the total number of elements in V. Since any such node is a member of one and one only of the sets $\{V_X, V_I, V_B, V_S\}$, then

$$V = \{V_X \oplus V_I \oplus V_B \oplus V_S\}, \tag{7.8}$$

and n, the total number of nodes in the network, is given by:

$$n = n_S + n_X + n_I + n_B, \tag{7.9}$$

7.3.4 The network degree equation

The network degree equation, given in Appendix A.1, is the sum of the degree of the network nodes, i.e.

$$D_G = 2m = \sum_{i=1}^{n} \text{degree}(v_i) \tag{A.1}$$

Since the set V includes all the types as given in equation (7.8), and since the elements of each type are not included in any other type then the degree of each type can be defined as follows:

The degree of the external nodes is

$$D_X = 2m_X = \sum_{i=1}^{n_X} \text{degree}(v_{Xi}); \tag{7.10}$$

The degree of the internal nodes is

$$D_I = 2m_I = \sum_{i=1}^{n_I} \text{degree}(v_{Ii}); \tag{7.11}$$

The degree of the bridge nodes is

$$D_B = 2m_B = \sum_{i=1}^{n_B} \text{degree}(v_{Bi}); \tag{7.12}$$

The degree of the one-degree nodes is

$$D_S = 2m_S = \sum_{i=1}^{n_S} \text{degree}(v_{Si}); \tag{7.13}$$

Thus

$$D_G = D_X + D_I + D_B + D_S \tag{7.14}$$

7.4 Types of network cycles

In this section, the definition and properties of network cycles are reviewed.

7.4.1 A walk, a trail and a path

A walk, W, in a network of m edges and n nodes, is a finite sequence of nodes and edges $v_1, e_1, v_2, e_2, \cdots, v_{k-1}, e_{k-1}, v_k$ beginning and ending with a node such that v_i and v_{i+1} are the end nodes of the edge e_i, $1 \le i \le k$. Note that in a walk, edges and nodes can appear more than once. The length of a walk w, is the number of edges in the walk. A walk is **closed** if $v_1 = v_k$; otherwise the walk is **open.**

In the network of Figure 7.5, the sequence $v_1, e_1, v_2, e_2, v_3, e_4, v_6, e_9, v_5, e_5, v_3$ is an open walk, whereas the sequence $v_1, e_1, v_2, e_2, v_3, e_4, v_6, e_9, v_5, e_3, v_2, e_6, v_4, e_7, v_1$ is a closed walk.

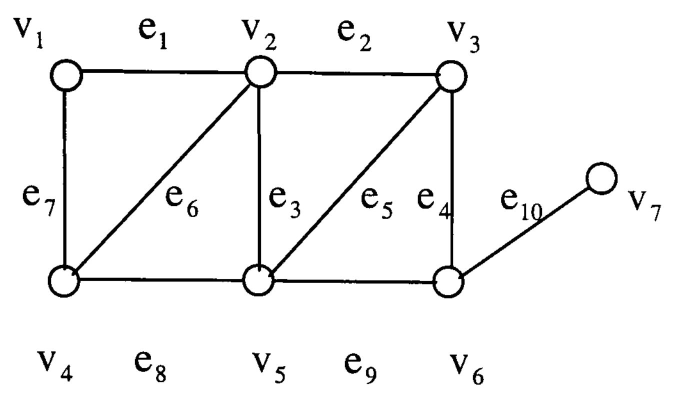


Figure 7.5 A network to illustrate the walk and the path

A walk is a **trail** if all its **edges** are distinct, i.e. each edge appears only once in the walk. A walk is a **path** if all its **nodes** are distinct, i.e. each node appears only once in the walk. A trail is termed Tr and a path is termed P. A trail or a path may be **open** or **closed.**

The properties of a closed trail are:

(1) Its edges are distinct, but nodes are not necessarily distinct,

(2) A closed trail is a closed path if all its nodes except the end nodes are distinct.

(3) One-degree nodes and edges do not exist in a closed trail,

(4) If a closed trail consists of only all external edges of a network, then the closed trail is termed an external closed trail.

In Figure 7.5, the sequence $v_1,e_1,v_2,e_2,v_3,e_4,v_6,e_9,v_5,e_5,v_3$ is an open trail, whereas the sequence $v_1,e_1,v_2,e_2,v_3,e_5,v_5,e_3,v_2,e_6,v_4,e_7,v_1$ is a closed trail.

The properties of a **closed path**:

(1) All nodes and edges are distinct.

(2) The minimum length is three edges,

(3) The number of edges equals the number of nodes

(4) Every node has at least two edges,

(5) $e_1e_2\cdots e_{k+1}$ and $e_{k+1}e_k\cdots e_1$ denote the same path,

(6) One-degree nodes and bridge nodes do not exist in the closed path,

(7) One-degree edges and bridge edges do not exist in the closed path.

(8) If a closed path consists of all external edges in a network, then the closed path is termed an external closed path.

(9) If a network has nodes of the following types $\{V_{Xd}, V_{XdI}, V_{XdB} V_{XdBI}\}$ for d >2, then the external closed path cannot contain all external edges of the network.

Let V_P be the set of nodes of a path P, and let n_P be the number of elements of V_P. Let E_P be the set of edges of a path P, and let m_P be the number of elements of E_P.

If a path P is open then P is termed P_o

$$n_{Po} = m_{Po} + 1, \qquad (7.15)$$

and if the path P is closed, then P is termed P_C

$$n_{Pc} = m_{Pc}. \qquad (7.16)$$

An example of an open path in Figure 7.5 is $P_o = \{v_1, e_1, v_2, e_2, v_3, e_4, v_6\}$.

The set of nodes of P_o is $V_{po} = \{v_1, v_2, v_3, v_4\}$ and $n_{po} = 4$, and the set of edges of P_o is $E_{Po} = \{e_1, e_2, e_4\}$ and $m_{po} = 3$.

An example of a closed path is $P_c = \{v_1, e_1, v_2, e_3, v_5, e_8, v_4, e_7, v_1\}$.

The set of nodes of P_c is $V_{pc} = \{v_1, v_2, v_4, v_5\}$ and $n_{pc} = 4$, and the set of edges of P_c is $E_{Pc} = \{e_1, e_3, e_7, e_8\}$ and $m_{pc} = 4$.

Remark (7.1):

If, in a network, the length of an external closed path equals the length of an external closed trail, then $E_B = \{\Phi\}$ and $V_{XD} = V_{XDI} = \{\Phi\}$ for d>2.

7.4.2 A cycle

The cycle definition is introduced in Appendix A. A cycle, in a network, is a closed path with a minimum length. The cycle path starts and ends at the same node, so that every edge and every node is counted once only. The length of a cycle is defined as the number of edges. In a cycle, the number of edges equals the number of nodes. The cycle is usually denoted $e_1e_2 \cdots e_k$ (instead of $e_1e_2 \cdots e_ke_1$), and $e_1e_2 \cdots e_k$ and $e_2 \cdots e_ke_1$ denote the same cycle.

Let $V_{XCi} = \{v_{XC1}, v_{XC2}, \cdots, v_{XCn_{XCi}}\}$ be the set of external nodes of the i th cycle, and let the number of elements of V_{XCi} be:

$$n_{XCi} = |V_{XCi}|. \tag{7.17}$$

Let $V_{ICi} = \{v_{IC1}, v_{IC2}, \cdots, v_{ICn_{ICi}}\}$ be the set of internal nodes of the i th cycle, and let the number of elements of V_{ICi} be:

$$n_{ICi} = |V_{ICi}|. \tag{7.18}$$

Then the set of nodes of the i th cycle is:

$$V_{Ci} = \{V_{XCi} \oplus V_{ICi}\}, \tag{7.19}$$

and, the number of elements of V_{Ci} is:

$$n_{Ci} = |V_{Ci}| = n_{XCi} + n_{ICi}. \tag{7.20}$$

Let $E_{XCi} = \{e_1, e_2, \cdots, e_{m_{XCi}}\}$ be the set of external edges of the i th cycle, and let the number of elements of E_{XCi} be:

$$m_{XCi} = |E_{XCi}|. \tag{7.21}$$

Let $E_{ICi} = \{e_1, e_2, \cdots, e_{m_{ICi}}\}$ be the set of internal edges between the i th cycle and j th cycles for $j = 1,2,\ldots,c$ and $j \neq i$, and let the number of elements of E_{ICi} be defined as follows:

$$m_{ICi} = |E_{ICi}| = \sum_{\substack{j=1, \\ j\neq i}}^{c} m_{IC(i,j)} \; ; \qquad (7.22)$$

Then the set of edges of the i th cycle is:

$$E_{Ci} = \{E_{XCi} \oplus E_{ICi}\}; \qquad (7.23)$$

and the total number of elements of E_{Ci} is:

$$m_{Ci} = |E_{Ci}| = m_{XCi} + m_{ICi} \, . \qquad (7.24)$$

Since the cycle is a closed path, i.e. it starts and ends at the same node, and since the nodes of the closed path are distinct, then the edges of the closed path are distinct. Each node in the closed path must have at least two distinct edges, one out of the node, and one into the node. The out-edge, of the k th node, is an in-edge of the following node (the first node) in the closed path, and the in-edge of the k th node is an out-edge of the previous (k-1) node.

Therefore P, the closed path must have at least three nodes. Each node has two distinct edges, and each edge has two distinct nodes. The three nodes have three distinct edges. Thus a closed path with k nodes has k distinct edges. If the length of the closed path is minimum, the closed path is a cycle, and thus the number of edges of the cycle equals the number of nodes, i.e.

$$m_{Ci} = n_{Ci} \, . \qquad (7.25)$$

Remark (7.2):

If a network has n nodes and m edges and m = n, then it has one and only one cycle.

7.4.3 Cycle types

The edges of a cycle may be:

(1) all external edges, or

(2) all internal edges, or

(3) a mix of external and internal edges.

If all edges of a cycle are external, then the cycle is termed **an external cycle**; if all edges of a cycle are internal, then the cycle is termed **an internal cycle**; and if the cycle edges consist of both external and internal edges then the cycle is termed a **mixed-cycle**.

Let c_X be the number of external cycles in the network,

Let c_I be the number of internal cycles in the network,

Let c_{XI} be the number of mixed cycles in the network,

Then c the total number of cycles in the network can be written as:

$$c = c_X + c_I + c_{XI}\,. \tag{7.26}$$

A cycle has the properties:

(1) Nodes and edges are distinct.

(2) Every node has at least two edges.

(3) The number of edges equals the number of nodes.

(4) The length of the cycle is minimum.

(5) $\{e_1e_2\cdots e_k\}$ and $\{e_2\cdots e_ke_1\}$ denote the same cycle.

(6) Bridge edges and one-degree edges do not exist in a cycle.

Define g_i, the circumference of the i^{th} cycle as the number of edges of the i^{th} cycle. Since the number of edges and the number of nodes in a cycle are equal as given in equation (7.25), then:

$$g_i = m_{Ci} = n_{Ci}. \qquad (7.27)$$

The circumference of an external cycle is:

$$g_{XCi} = m_{XCi} = n_{XCi}. \qquad (7.28)$$

The circumference of an internal cycle is:

$$g_{ICi} = m_{ICi} = n_{ICi}. \qquad (7.29)$$

The circumference of the i^{th} mixed-cycle is:

$$g_{XCi} = m_{XCi} + m_{ICi} = n_{XCi} + n_{ICi}. \qquad (7.30)$$

Let g_C be the sum of the c circumferences of the c cycles in G, then

$$g_C = \sum_{i=1}^{C} g_i ; \qquad (7.31)$$

7.4.4 The network external closed path

In Theorem (7.1), conditions are established under which all external edges of a network G are in a single closed path. This closed path is termed P_{GX}, the network external path, and the length of P_{GX} is p_{GX}

Theorem (7.1)

If, in a network, m>n, $E_S = \{\Phi\}, E_B = \{\Phi\}$, $V_{XD} = \{\Phi\}$ for $d > 2$,

$V_{XDI} = \{\Phi\}$ for $d > 2$ then

(1) $m_X = m - m_I$ (7.32)

(2) $n_X = n - n_I = n_{X2} + n_{X2I}$, (7.33)

(3) $p_{GX} = m_X = n_X$ (7.34)

(4) $c_{XI} = n_{X2I}$ (7.35)

(5) $c_I = c - c_{XI}$ (7.36)

Proof:

Let G be a network with m edges and n nodes, then

$$m = m_S + m_I + m_X + m_B$$

$$V = \{V_X \oplus V_I \oplus V_B \oplus V_S\}$$

$n = n_S + n_I + n_X + n_B$, and

$$V_X = \{V_{XD} \oplus V_{XDI} \oplus V_{XDB} \oplus V_{XDBI}\}$$

$$n_X = n_{XD} + n_{XDI} + n_{XDB} + n_{XDBI}.$$

Since m>n then c>1. Thus $m_I > 0$ and $n_I \geq 0$.

Since each one-degree node has one-degree edge, then $n_S = m_S$.

Since $E_S = \{\Phi\}$ then $m_S = 0$ and $n_S = 0$.

Since $E_B = \{\Phi\}$ then $m_B = 0$ and $n_B = 0$.

Thus $V_{XdB} = \{\Phi\}$, $V_{XdBI} = \{\Phi\}$, $n_{XDB} = 0$ and $n_{XDBI} = 0$.

Thus m consists of external and internal edges only

(1) $m_X = m - m_I$,

and n also consists of external nodes and internal nodes only:

$$V = \{V_X \oplus V_I\};$$

and $n = n_X + n_I$.

Furthermore,

Since $V_{Xd} = \{\Phi\}$ for $d > 2$, then $V_{Xd} = \{V_{X2}\}$ and $n_{XD} = n_{X2}$,

Since $V_{XdI} = \{\Phi\}$ for $d > 2$, then $V_{XdI} = \{V_{X2I}\}$ and $n_{XDI} = n_{X2I}$.

Thus the external nodes consist of:

$$V_X = \{V_{X2} \oplus V_{X2I}\},$$

and

(2) $n_X = n - n_I = n_{X2} + n_{X2I}$.

Each external edge, by definition, belongs to a cycle. This cycle is either an external cycle or a mixed-cycle.

Since $m_I > 0$

and $E_B = \{\Phi\}$.

and $V_{Xd} = \{V_{X2}\}$,

and $V_{XdI} = \{V_{X2I}\}$,

Then, there are no external cycles in the network,

and the external closed trail is the external closed path.

Thus all external nodes and edges belong to mixed-cycles,

and they form an external closed path or an external closed trail.

The length of the external closed trail equals the length of the external closed path.

Hence,

(3) $p_{GX} = m_X = n_X = m - m_I$.

Each external node of V_{X2}, has a degree equals to two, and it belongs to only one cycle. Each node of V_{X2I} has two external edges and internal edges, and it belongs to more than one cycle. Therefore, each external edge between two V_{X2I} nodes belongs to a different cycle. Thus, n_{X2I}, the number of the V_{X2I} nodes gives the number of mixed cycles in the network.

(4) $c_{XI} = n_{X2I}$,

Thus, the number of internal cycles can be obtain as follows:

(5) $c_I = c - c_{XI}$.

■

7.5 Membership of edge sets

As introduced in Section 7.2, there are four edge sets, namely the one-degree edge set, the bridge edge set, the internal edge set and the external edge set. The existence of some of these sets, in a given network, depends on other sets. For example, it is not possible to have the internal set without having the external set. Also it is not possible to have the one-degree set without having the external or bridge set. Furthermore, the set of edges of each type will be classified into subsets based on the end-nodes of each edge.

7.5.1 The number of one-degree edges

The number of one-degree edges in a network can be determined from the incidence matrix of the network. Since every column represents a node, a column that contains only one non-zero element represent a one-degree node. The sum of such columns gives the number of one-degree edges in the network.

7.5.2 The number of external and internal edges

The number of internal edges m_I in a network depends on the network configuration. A network of m edges and n nodes has m-n+1 cycles. If the network has a ladder shape as shown in Figure 7.6, then between adjacent cycles there is one internal edge, and the total number of internal edges between all the cycles is $m_I = m - n$.

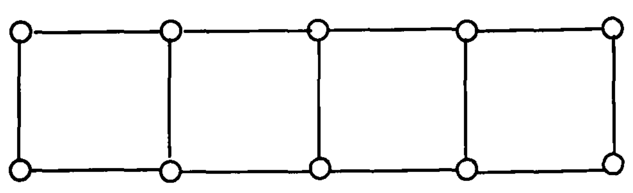


Figure 7.6 A network of ladder shape

If, however, the network has a configuration such as that shown in Figure 7.7, then the number of internal edges equals the number of cycles.

$$m_I = m - n + 1$$

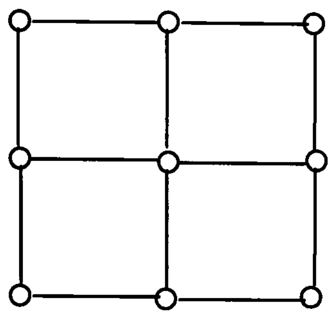


Figure 7.7 A network in which the number of internal edges equal the number of cycles

Some networks may have no internal edges. For example, the following network shown in Figure 7.8 has no internal edges, but has an external node with degree four as a common node between the two cycles.

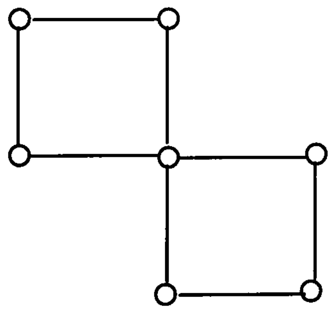


Figure 7.8 A network with external node of degree four

The number of internal edges in a class of networks is established in Theorem (7.2).

Theorem (7.2):

If a network has $E_S = \{\Phi\}$, $E_B = \{\Phi\}$, $V_{Xd} = \{\Phi\}$ for d>2 and m>n, then

(1) $m_X = m - m_I$

(2) $n_I = n - m_X$ (7.37)

(3) $g_C = m_X + 2m_I$ (7.38)

(4) $m_I = g_C - m$ (7.39)

(5) $g_C = 2m - n_X$ (7.40)

(6) $m_I = \frac{1}{2}(g_C - n + n_I)$ (7.41)

Proof:

Let G be a network with m edges and n nodes, then

$m = m_X + m_I + m_B + m_S$;

$n = n_X + n_I + n_B + n_S$;

$n_X = n_{XD} + n_{XDI} + n_{XDB} + n_{XDBI}$;

Since $E_S = \{\Phi\}$, then $m_S = 0$ and $n_S = 0$.

Since $E_B = \{\Phi\}$, then $m_B = 0$ and $n_B = 0$.

Also $V_{Xd} = \{\Phi\}$ and $n_{XdB} = 0$;

And $V_{XdBI} = \{\Phi\}$ and $n_{XdBI} = 0$.

Therefore, $n_{XDB} = 0$ and $n_{XDBI} = 0$.

Since $V_{Xd} = \{\Phi\}$ for d>2, then $n_{Xd} = 0$, for d>2, and $n_{XD} = n_{X2}$.

Therefore,

$m = m_X + m_I$,

and

$n = n_X + n_I$,

and

$n_X = n_{XD} + n_{XDI} = n_{X2} + n_{XDI}$, Hence the total number of external edges, or the circumference of the network external path is:

(1) $m_X = g_{GX} = m - m_I$,

and

$m_X = n_X = n_{X2} + n_{XDI}$,

The total number of internal nodes in the network is then

(2) $n_I = n - m_X$,

and it also equals:

$n_I = n - n_X = n - (n_{X2} + n_{XDI})$,

Since $m > n$, then from 1 and 2

$m_I > n_I$,

if $n = n_X$, then $n_I = 0$,

and if $n > n_X$ then $n_I > 0$.

Thus $n_I \geq 0$.

From 1, if $m = m_X$ then,

$m_I = 0$,

and there is one and only one closed path in the network, thus $c = 1$.

If $m > m_X$ then, $m_I > 0$ and $c > 1$.

Thus $c \geq 1$.

The circumference of the i^{th} mixed-cycle is given in equation (7.28):

$$g_{XCi} = m_{XCi} + \sum_{\substack{j=1, \\ j \neq i}}^{c} m_{IC(i,j)} \; .$$

The total circumference of the c cycles is given by equation (7.30):

$$g_C = \sum_{i=1}^{C} g_i \; ;$$

Since there is at least one internal edge between any two cycles, then each internal edge will be counted twice, once in each cycle, and the two end-nodes of the internal edge will also be counted twice, while each external edge or node will be counted once. Therefore, the sum of edges in the c cycles can be written as:

$$g_C = (m_{XC1} + \sum_{j=2}^{c} m_{IC(1,j)}) + (m_{XC2} + \sum_{\substack{j=1, \\ j\neq 2}}^{c} m_{IC(2,j)}) + \cdots + (m_{XCc} + \sum_{j=1}^{c-1} m_{IC(c,j)}),$$

If there is no internal edge between the ith and jth cycles, then

$$m_{IC(i,j)} = 0,\ V_{IC(i,j)} = \{\Phi\},\ \text{and}\ n_{IC(i,j)} = 0,$$

and if there is an internal edge between the ith and jth cycles,

then $m_{IC(i,j)} = m_{IC(j,i)}$,

then, g_C can be written in the following form:

$$g_C = \sum_{i=1}^{c} m_{XCi} + \sum_{i=1}^{c} \sum_{\substack{j=i+1 \\ j\neq i}}^{c-1} m_{IC(i,j)};$$

Since every external edge occurs once and once only in a cycle, then the sum of external edges in the c cycles equals m_X the total number of external edges in the network, i.e.

$$\sum_{i=1}^{c} m_{XCi} = m_X,$$

Since every internal edge share two and only two cycles, then every internal edge will be counted twice, once in each cycle, thus

$$\sum_{i=1}^{c} \sum_{\substack{j=i+1 \\ j\neq i}}^{c-1} m_{IC(i,j)} = 2m_I$$

then

(3) $\quad g_C = m_X + 2m_I$

Since (1) gives $m = m_X + m_I$,

Then $\quad g_C = m + m_I$

Therefore,

$$g_C > m,$$

and

(4) $m_I = g_C - m.$

The circumference also equals the sum of external nodes and internal nodes of i th mixed-cycle as given by equation (7.30), i.e.

$$g_i = \left| \left\{ V_{XCi} \bigcup_{\substack{j=1,\ldots,c \\ j\neq i}} V_{IC(i,j)} \right\} \right|,$$

the sum of the c circumferences is

$$g_C = \sum_{i=1}^{c} \left| \left\{ V_{XCi} \bigcup_{\substack{j=1,\ldots,c \\ j\neq i}} V_{IC(i,j)} \right\} \right|,$$

$$g_C = \left|\{(V_{XC1} \bigcup_{j=2,\ldots,c} V_{IC(1,j)})\}\right| + \left|\{(V_{XC2} \bigcup_{\substack{j=1,2,\ldots,c \\ j\neq 2}} V_{IC(2,j)})\}\right| + \cdots + \left|\{(V_{XCc} \bigcup_{j=1,2,\ldots,c-1} V_{IC(c,j)})\}\right|,$$

$$g_C = \sum_{i=1}^{c} \left|\{V_{XCi}\}\right| + \sum_{i=1}^{c} \left| \left\{ \bigcup_{\substack{j=1,\ldots,c \\ j\neq i}} V_{IC(i,j)} \right\} \right|,$$

$$n_{XC} = \sum_{i=1}^{c} \left|\{V_{XCi}\}\right|$$

The set of external nodes of the i th mixed-cycle consists of V_{X2I} as in V_{XdI} type and V_{X2} as in V_{Xd} type, and the union of the c sets includes all the external nodes in the network.

$$n_X = \left| V_X = V_{XC} = \left\{ \bigcup_{i=1,\ldots,c} V_{XCi} \right\} \right|.$$

Every node of the V_{X2} type has a degree equal two, and it is used by only one cycle, i.e. degree $(v_{X2})-1$. Every node of V_{X2I} type is shared by y cycles, where y equals the degree of the i^{th} node from the V_{X2I} type minus one, i.e. degree$(v_{X2I})-1$. Thus, the sum of the external nodes of the c cycles equals the sum of the cycles they belong to, i.e.

$$n_{XC} = \sum_{i=1}^{n_X} \text{degree}(v_{XCi}) - n_X ,$$

In the internal nodes side,

$$n_I = \left| V_I = V_{IC} = \left\{ \bigcup_{i=1,\dots,c} V_{ICi} \right\} \right|$$

the number of cycles associated with an internal node equals its degree.
The sum of internal nodes of the c cycles equals the sum of the internal nodes degree, i.e.

$$\sum_{i=1}^{c} \left| \left\{ \bigcup_{\substack{j=1,\dots,c \\ j\neq i}} V_{IC(i,j)} \right\} \right| = \sum_{i=1}^{n_I} \text{degree}(v_{ICi});$$

Therefore

$$g_C = \sum_{i=1}^{n_X} \text{degree}(v_{XCi}) - n_X + \sum_{i=1}^{n_I} \text{degree}(v_{ICi}),$$

but

$$2m = \sum_{i=1}^{n_X} \text{degree}(v_{XCi}) + \sum_{i=1}^{n_I} \text{degree}(v_{ICi}),$$

Thus

$$(5) \qquad g_C = 2m - n_X ,$$

Substitute from (3), then

$$g_C = 2m - (n - n_I) = 2m - n + n_I ,$$

$$g_C = 2(m_X + m_I) - n + n_I ,$$

$$g_C = 2(n - n_I + m_I) - n + n_I ,$$

$$g_C = n - n_I + 2m_I,$$

Thus

$$m_I = \frac{1}{2}(g_C - n + n_I). \tag{6}$$

■

7.5.3 The number of bridge edges

Let a network have $c > 1$ cycles, with no bridge edges between them as shown in Figure 7.9a. Then if the c cycles are separated such that a bridge edge is inserted between each pair of adjacent cycles, then the maximum number of bridge edges between the c cycles is $c - 1$ bridge edges, as shown in Figure 7.9b

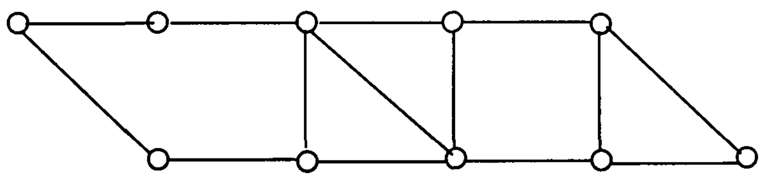


Figure 7.9a A network without bridge edges

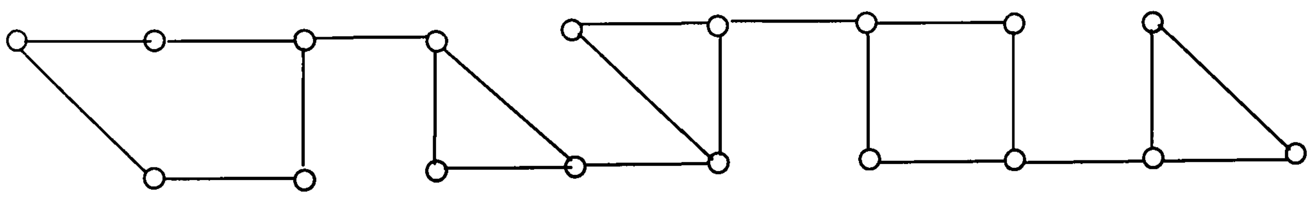


Figure 7.9b A network with bridge edges

The following theorem proves that, in any network, if $g_C = n$ then the number of bridge edges is $c - 1$.

Theorem 7.3: (The bridge edges)

In any network, if $c > 1$, $V_S = \{\Phi\}$, $V_I = \{\Phi\}$, $E_I = \{\Phi\}$and $V_B = \{\Phi\}$ then

$$g_C = n, \tag{7.42}$$

and

$$m_B = (c - 1). \tag{7.43}$$

Proof:

Let G be a network with m edges and n nodes.

Then

$$m = m_S + m_I + m_X + m_B;$$

$$n = n_S + n_X + n_I + n_B;$$

$$n_X = n_{XX} + n_{XI} + n_{XB} + n_{XBI} + n_{BX} + n_{BXI};$$

Since $V_S = V_I = V_B = E_I = \{\Phi\}$,

then $m_S = n_S = 0$; $n_I = 0$;

$m_I = 0$; and $n_{XdI} = n_{XdBI} = 0$;

and $n_B = 0$.

Thus

$$m = m_X + m_B,$$

and the total number of external edges in the c cycles is

$$m_X = m - m_B.$$

Also n is reduced to $n = n_X$, and n_X is reduced to $n_X = n_{XD} + n_{XDB}$.

Thus the total number of external nodes in the c cycles is $n_X = n_{XD} + n_{XDB} = n$.

Since c>1 and $n_I = 0$ then all cycles are of external types.

The circumference of the i^{th} external cycle is given by equation (7.21), i.e.

$$g_i = m_{XCi};$$

and

$$g_i = n_{XCi},$$

The total circumference of the c cycles is given by equation (7.24).
Thus

(1) $$g_C = \sum_{i=1}^{c} m_{XCi} = \sum_{i=1}^{c} n_{XCi} = m_X = n_X = n.$$

Since there are c cycles, and since $m_I = 0$, then

$$m_B = m - m_X;$$

since $m_X = n$, then

$$m_B = m - n;$$

but $c - 1 = m - n$, thus

(2) $$m_B = c - 1.$$

■

It can be noted that if one of the c-1 bridge edges is to be converted to an internal edge, then two edges and two nodes must be deleted from the network. The bridge edge and an external edge, with its end-nodes, at one of the end-nodes of the bridge edge. For example, in Figure 7.10a, e_1 is a bridge edge, it has two external edges (e_2, e_3) at one end-node and it has two external edges (e_4, e_5) at the other end-node. When e_1 is deleted from the network, one of the external edges must be deleted with its end-nodes. When one of the external edges is deleted the other dependent external edge will be an internal edge, i.e. if e_3 is deleted, then e_4 will be an internal edge. And if e_2 is deleted then e_5 will be an internal edge.

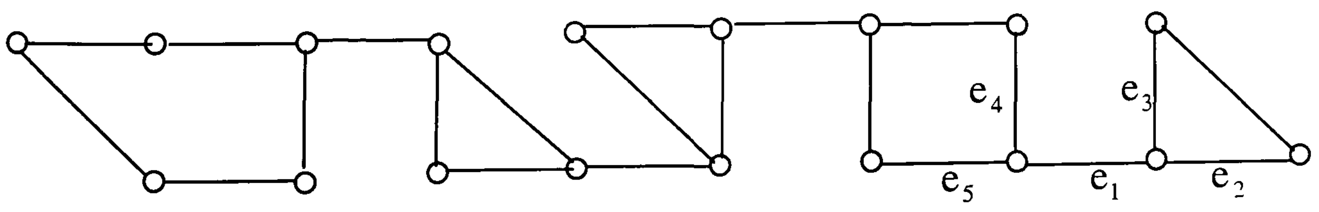


Figure 7.10a A network with n=18,m=22, g_C=18

Figure 7.10b shows the network after converting a bridge edge to an internal edge.

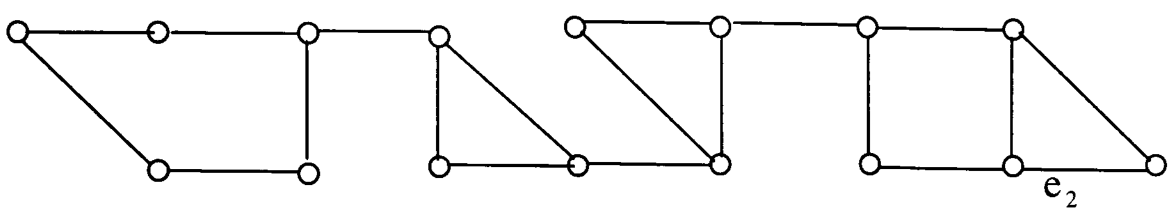


Figure 7.10b Converting a bridge edge to an internal edge

Every time a bridge edge is converted into an internal edge, n, the total number of nodes in the network, reduces by two nodes, and m, the total number of edges in the network, also reduces by two edges. The number of cycles, c, does not change, and g_C, the sum of the circumferences of the c cycles, also does not change.

When a network has a combination of internals and bridges, this fact can be used to find the number of internal edges and bridge edges in the network.

Theorem (7.4)

In a network, if $m_B > 0$ and $m_I > 0$, then

(1) $p_{GX} = m_{XC} = m_X = n_X$. (7.44)

(2) $m_B = D_X - 2n_X - m_{XI}$ (7.45)

(3) $g_C = m_X + 2m_I = n_{XC} + n_{IC}$. (7.46)

Proof:

If $m_I = 0$ then from Theorem (7.3) $g_C = n$ and $m_B = (c-1)$.

If $m_I > 0$ then c>1.

The circumference of the i^{th} mixed-cycle is given by equation (7.30), i.e.

$$g_{XCi} = m_{XCi} + m_{ICi} = n_{XCi} + n_{ICi},$$

The sum of the c circumferences, i.e. $g_C = \sum_{i=1}^{c} g_i$, is given by equation (7.31).

Applying (7.31) on (7.30) gives the following:

$$g_C = \sum_{i=1}^{c} m_{XCi} + \sum_{i=1}^{c} m_{ICi} = \sum_{i=1}^{c} n_{XCi} + \sum_{i=1}^{c} n_{ICi}$$

$$g_C = m_{XC} + m_{IC} = n_{XC} + n_{IC}.$$

Since $m_B > 0$ then the network has more than one external closed path. Since the external edges of the closed path are distinct, then the total number of external edges in the closed paths equals the total number of external edges in the network, i.e.

(1) $p_{GX} = m_{XC} = m_X = n_X$.

Every external node has two or more edges. If an external node has two edges then the two edges will be in the external closed path, and if the external node has more than two edges, then two edges will be in the closed path and the extra edges will be either internal edges or bridge edges or both. Those internal and bridge edges indicate that the type of their external nodes is either V_{XdI} or V_{XdBI} for $d = 2,4,\ldots$.

Therefore, the degree of a node of type V_{XdI} minus two gives the number of internal edge with respect to that node. Similarly, the degree of a node of type V_{XdB} or V_{XdBI} minus two gives the number of bridge edges and internal edges at that node. Let m_{IX} be the number of internal edges from the nodes of type V_{XdI}, then:

$$m_{IX} = \sum_{i=1}^{n_{X2I}} \text{degree}(v_{X2I}) - n_{X2I};$$

If this is applied over the set of external nodes V_X, then the number of bridge edges and m_{IX} internal edges can be obtained as follows:

Since D_X, the degree of the external nodes is given by equation (7.10), includes all edges connected to the set of external nodes, then

(2) $m_B = D_X - 2n_X - m_{XI}$.

Since every internal edge is counted twice, once in each cycle, then

$$m_{IC} = 2m_I$$

Thus g_C can be written as follows:

(3) $g_C = m_X + 2m_I = n_{XC} + n_{IC}$

■

7.6 The maximum circumference

In a network with c cycles the minimum circumference a cycle can have is three edges. Let $g_{min} = 3$ be the minimum circumference of a cycle. The circumferences of the c cycles may be equal or different. If the c circumferences are different then the cycle with maximum circumference will be termed g_{max}. The following theorem introduces the conditions that the maximum circumference in a network is $g_{max} = n - 1$.

Theorem (7.5)

In a network with only V_{X2}, V_{X2I}, $n_I > 0$ and with $g_i = 3$ for $i = 1,2,...,c-1$, $c > 1$, then the maximum circumference is:

$$g_c = g_{max} = n - 1. \qquad (7.47)$$

Proof:

Since $V = \{V_{X2} \oplus V_{X2I}\}$

then $n = n_X + n_I$;

and $n_X = n_{X2} + n_{X2I}$;

and $p_{GX} = n_X = m_X$.

If $c = 1$ then $g_1 = m = n$.

If $c > 1$, and $m_I \geq 1$ then $m > n$,

and $g_i \neq n \neq m$; otherwise there is a contradiction with $g_1 = m = n$.

Thus $g_i < n < m$.

Since $g_{\min} = 3$ then $g_{\max} > g_{\min}$ always.

The necessary conditions to have one of the c cycles with $g_{\max}$ are:

Each cycle of the remaining c-1 cycles, has a circumference equals $g_{\min}$, this condition necessitates that c>2, otherwise if $c = 1$ the two cycles can have $g_1 = g_2 = g_{\min}$.

Thus if c>2 and $g_i = g_{\min}$ for $i = 1,\ldots,c-1$ then:

The external edges of the c-1 cycles are $m_{X(c-1)} = 2$ edges.

The internal edges between the c cycles is $m_I = (c-2) + (c-1) = 2c-3$,

There are three nodes of type V_{X2I}. One node with degree c, and two nodes with degree three. Thus the total number of external nodes in the c-1 cycles is $n_{XC(c-1)} = 3$. The total number of internal nodes is $n_I = c-2$.

Then, the total circumferences of the c-1 cycles is

$$g_{(c-1)} = \sum_{i=1}^{c-1} g_{\min} = g_{\min} \text{ x } (c-1) = 3(c-1),$$

Thus the c th cycle has $n_{XCc} = n_X - 1$ and $n_{ICc} = n_I = c-2$.

Thus
$$\begin{aligned} g_{\max} = g_c &= n_{XCc} + n_{ICc}, \\ &= n_X - 1 + n_I, \end{aligned}$$

$$g_{\max} = n - 1$$

■

7.6.1 The network and the maximum circumference

Any network with $g_{max} = n - 1$ will be termed the basic configuration of G, and it has the following properties:

It has one node with degree = c.

It has c nodes with degree = 3.

It has two nodes of V_{X2I} type.

It has c-2 internal nodes.

It has n-(c+1) nodes with degree = 2;

It has $m_I = c + 1$;

It has $g_C = 3c + n - 4$;

It has one cycle with $g_{max} = n - 1$,

It has c-1 cycles with $g = g_{min}$

7.7 Sub-classes of membership

The number of edges of each type can be obtained by using the previous equation. An edge of any type has two end-nodes. The two end-nodes may be of the same type as the edge type, or one of the end-nodes might be of different type. For example, the two end-nodes of an internal edge may be internal nodes, or one of them might be an internal node and the other an external node. Furthermore, the degrees of the two end-nodes, of the same type or different type, may be the same or different. For example, an external edge may have two end-nodes of V_{Xd} type, and d may have the same value, or it may have different values. Also an external edge may have one end-node of V_{Xd} type and the other end-node of V_{XdI} type. Therefore, the set of edges of each type will be classified further based on the type of the end-nodes of that edge, i.e. if the two end-nodes have exactly the same type, then the edge will be termed by the same type of the end-nodes, and if the two end-nodes have different types, then the edge will be

termed by the type of the two end-nodes. If d of the two end-nodes is the same value, then the value of d will be used, and if d has two different values, then the two values will be used.
Therefore, the following sub-classes are defined:

(1) The sub-classes of the one-degree edges are:

(i) SX edge

A one degree edge is termed SX if:

(1) One end node is of one-degree type.

(2) The other end-node is of V_X type.

(ii) SB edge

A one degree edge is termed SB if:

(1) One end-node is of one-degree type.

(2) The other end-node is of V_B type.

(iii) SI edge

A one degree edge is termed SI if:

(1) One end-node is of one-degree type.

(2) The other end-node is of V_I type.

(2) The sub-classes of the bridge edges are:

(i) BB edge

A bridge edge is termed BB if

(1) The two end-nodes of the edge are of V_B type.

(ii) BX edge

A bridge edge is termed BX if:

(1) One end-node is of V_{XdB} type.

(2) The other end-node is of V_B type.

(iii) BXX edge

A bridge edge is termed BXX if:

(1) The two end-nodes are of V_{XdB} type.

(iv) BXI edge

A bridge edge is termed BXI if:

(1) One of the end-nodes is of V_{XdB} type.

(2) The other is of V_{XdBI} type.

(v) BXXI edge

A bridge edge is termed BXXI if:

(1) The two end-nodes are of V_{XdBI} type.

$E_B = \{E_{BB}, E_{BX}, E_{BXI}, E_{IX}\}$

(3) The sub-classes of the internal edges are:

(i) IX edge

An internal edge is termed IX if:

(1) One end-node is of V_X type.

(2) The other end-node is of V_I type.

(ii) II edge

An internal edge is termed II if:

(1) The two end-nodes are of V_I type.

Thus, $E_I = \{E_{II}, E_{IX}\}$

(4) The sub-classes of the external edges are:

(i) XX edge

An external edge is termed XX if:

(1) The two end-nodes are of V_{Xd} type.

(2) The two end nodes have the same value of d.

(3) It belongs to only one cycle.

(ii) X_{ij} edge

An external edge is termed X_{ij} if:

(1) The two end-nodes are of V_{Xd} type.

(2) The values of d are i and j and $i \neq j$.

(iii) XI_i edge

An external edge is termed XI_i if:

(1) One end-node is of V_{Xd} type.

(2) The other is of V_{XdI} type with i internal edges.

(iv) XI_{ij} edge

An external edge is termed XI_{ij} if:

(1) The two end-nodes are of V_{XdI} type.

(2) The i and j are the number of internal edges in the first and the second end-nodes.

(v) XB edge

An external edge is termed XB if:

(1) One end-node is of V_{Xd} type.

(2) The other end-node is of V_{XdB} type.

(vi) XBI_i edge

An external edge is termed XBI_i if:

(1) One end-node is of V_{Xd} type or V_{XdB} type.

(2) The other end-node is of V_{XdBI} or V_{XdI} type.

(vii) XBIij edge

An external edge is termed XBIij if:

(1) One end-node is of V_{XdI} type with i internal edges.

(2) The other end-node of V_{XdBI} with j internal edges.

Thus $E_X = \{E_{XX}, E_{Xij}, E_{XIi}, E_{XIij}, E_{XB}, E_{XBIi}, E_{XBIij}\}$

7.7.1 Connections between classes

The nodes and edges classification and sub-classification have been introduced in Section 6.2 and 6.3. An external node of subclass V_{X2} is always connected, by an edge of external type, to an external node of subclass V_{X2}, V_{XdI} or V_{XdB}. Similarly, a bridge node of subclass V_{BB} is always connected, by an edge of bridge type, to a node of subclass V_{BB}, V_{XdB} or V_{XdIB}. Also an internal node of sub class V_{II} is always connected to a node of subclass V_{II} or V_{XI}. A nodes of a class are connected by edges of the same class, and nodes at the boundary, i.e. between two classes have a mixed entity of the two classes. Figure 7.11 shows the different connections between classes.

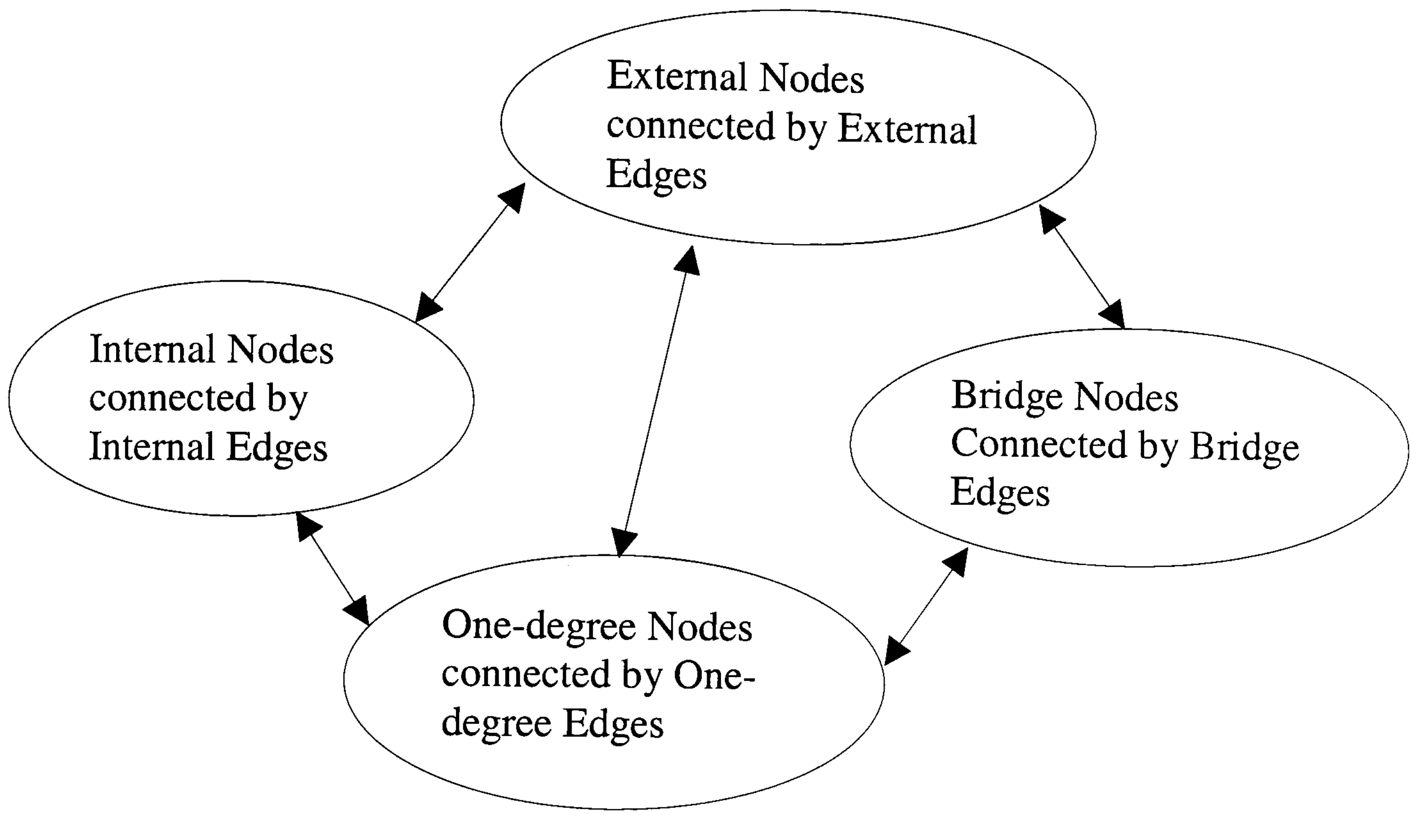

External Nodes connected by External Edges
Internal Nodes connected by Internal Edges
Bridge Nodes Connected by Bridge Edges
One-degree Nodes connected by One-degree Edges

Figure 7.11 Connection between classes

7.7.2 The sub-classes of the mixed-cycles

A mixed cycle, c_{XI}, is defined in Section 7.4.3 as a cycle with mixed external edges and internal edges. Furthermore, a mixed-cycle may have zero, one or two edges of XI_{ij} type. Accordingly, a mixed-cycle is classified into the following three types:

(i) c_{XI0} mixed-cycle

A mixed-cycle, c_{XI}, is termed c_{XI0} if:

(1) It has zero edge of type XI_{ij}.

(2) It has one or more nodes of type V_{X2}.

(3) It has exactly two nodes of type V_{X2I}.

(ii) c_{XI1} mixed-cycle

A mixed-cycle, c_{XI}, is termed c_{XI1} if:

(1) It has one edge of XI_{ij} type.

(2) It has zero node of type V_{X2}.

(3) It has exactly two nodes of type V_{X2I}.

(iii) c_{XI2} mixed-cycle

A mixed-cycle, c_{XI}, is termed c_{XI2} if:

(1) It has two edges of XI_{ij} type.

(2) It has zero, one or more nodes of type V_{X2}.

(3) It has exactly four nodes of type V_{X2I}.

An adjacent cycle:
If two cycles have one or more share internal edges, then the two cycles are said to be adjacent.

Classifying the mixed-cycles into these types, introduces another natural property, namely "**the location property**" of a mixed cycle with respect to other mixed-cycles in the network external closed path.

Since a mixed-cycle of class c_{XI2} has two independent edges of XI_{ij} type, then c_{XI2} have at least two independent internal edges. Therefore, a c_{XI2} a mixed-cycle must lie between at least two independent mixed-cycles. Therefore a c_{XI2} mixed-cycle will be termed c_{XI2M}, **a two sided-middle mixed-cycle**. Also a mixed-cycle of class c_{XI1} has two internal edges, not necessary dependent, and therefore lies between at least two mixed-cycles, not necessarily dependent. A c_{XI1} mixed-cycle will be termed c_{XI1M}, **a one-sided middle mixed-cycle.**

A mixed-cycle of class c_{XI0} has one or more dependent internal edges, and it has one or more nodes of V_{X2} type. The V_{X2} type nodes belong to one and only one cycle. Therefore, a mixed-cycle of class c_{XI0} has one or more adjacent mixed-cycles, and will be termed c_{XI0T}, **a terminal mixed-cycle.**

7.8 The network open paths

The definition of an open path is given in Section 7.4.1. Since there are different types of edges in the network, and since an open path starts and ends at two different nodes, then the edges of a path can be used to classify the open path type. Thus, if all edges of an open path are external, then the path is termed **an external open path**. If all edges of an open path are internal edges, then the open path is termed **an open internal path**. If the edges of an open path consist of external edges and internal edges, then the open path is termed **an open mixed path**.

7.8.1 The open external paths

A path will be termed P_{OX}, **open external path if:**

(1) All its edges are of external type.

(2) Its end-nodes are external nodes.

(3) Its end-nodes belong to P_{GX}.

(4) Its length is the minimum number of external edges between the two external end-nodes.

7.8.2 The open internal paths

A path will be termed P_{OI}, **open internal path** if:

(1) Its edges are of internal type.

(2) Its end-nodes are always external nodes of type V_{X2I}.

(3) Its end-nodes belong to P_{GX}.

(4) Its length is the number of internal edges between the two external end-nodes.

(5) The end-nodes may have more than one internal path.

The ith P_{OI} may be written with its end-nodes as $P_{OI}(v_{X1},\ v_{X2})$. Let n_{OI} be the length of the ith P_{OI}, i.e. the number of edges between the external end-nodes. Since the end-nodes of P_{OI} may have one or more open internal paths, then $P_{OI\text{-}j}(v_{X1},\ v_{X2})$ for $j = 1,2,....$, is used to define and to distinguish each P_{OI} between the two external end-nodes. For example, in Figure 7.11, between v_2 and v_4 there is only one open internal path, i.e.

$P_{OI}(v_2,\ v_4) = \{e_5, e_6, e_7, e_8\}$ with $n_{OI}(v_2,\ v_4) = 4$.

While between v_1 and v_2 there are two open internal paths, i.e.:

$P_{OI\text{-}1}(v_1,\ v_2) = \{e_9, e_5\}$, with $n_{OI\text{-}1}(v_1,\ v_2) = 2$; and

$P_{OI\text{-}2}(v_1,\ v_2) = \{e_{10}, e_6, e_5\}$,

with $n_{OI\text{-}2}(v_1,\ v_2) = 3$.

Since the end-nodes of P_{OI} are external nodes of type V_{X2I}, then:

An external node of degree two does not have an internal path.

An external node of degree three has one internal path.

An external node of degree four has two internal paths.

Let $n_{P_{OI}}$ be the total number of different P_{OI} in the network, then there is a direct relationship between the external nodes and $n_{P_{OI}}$, i.e. in a network with $p_{GX} = n_X = m_X$, then:

$$n_{P_{OI}} = (n_X - 1) + (n_X - 2) + \cdots + 2 + 1 \,. \qquad (7.48)$$

Since the end-nodes are of type V_{X2I}, then the number of open internal paths is:

$$n_{P_{OI}} = \sum_{i=1}^{n_{X2I}-1}(n_{X2I} - i) = (n_{X2I} - 1) + (n_{X2I} - 2) + \cdots + 2 + 1. \tag{7.49}$$

and let S_{GOI} be the sum of the lengths of the $n_{P_{OI}}$ paths, then

$$S_{GOI} = \sum_{i=1}^{n_{P_{OI}}} n_{OI}(v_{X1}, v_{X2}); \tag{7.50}$$

Let P_{GOI} be the set of open internal paths in a network, then

$$P_{GI} = \{P_{OI}(v_i, v_j)\}, \text{ with } i = 1 : n_X - 1; \text{ and } j = i + 1 : n_X. \tag{7.51}$$

Example (7.1): The open internal paths of Figure 7.12 are derived

.

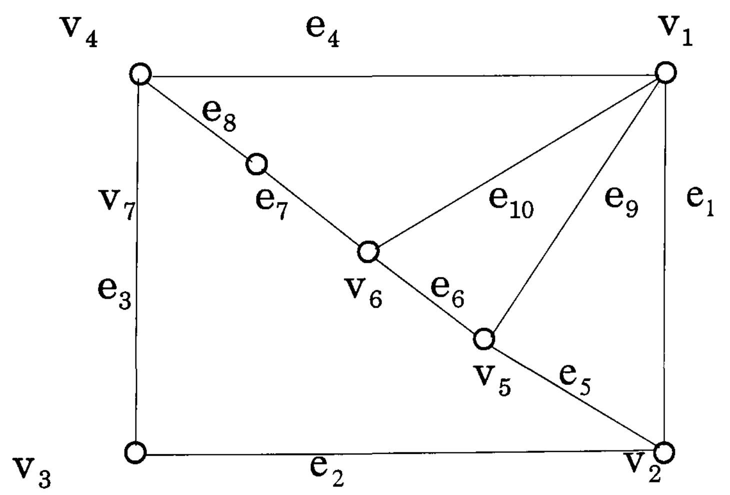


Figure 7.12 A network to illustrate the open internal paths

$P_{OI}(v_1, v_2) = \{e_9, e_5\}$; and $n_{OI\text{-}1} = 2$

$P_{OI}(v_1, v_3) = \{\Phi\}$;

$P_{OI}(v_1, v_4) = \{e_{10}, e_8, e_7\}$; $n_{OI\text{-}2} = 3$;

$P_{OI}(v_2, v_3) = \{\Phi\}$;

$P_{OI}(v_2, v_4) = \{e_3, e_6, e_7, e_8\}; n_{OI\text{-}3} = 4$;

$P_{OI}(v_3, v_4) = \{\Phi\}$;

thus

$n_{P_{OI}} = 3$;

$$n_{POI} = \sum_{i=1}^{3} n_{OI\text{-}i} = 2 + 3 + 4 = 9;$$

$P_{GI} = \{P_{OI\text{-}i}; i = 1 : n_{P_{OI}}\}$.

7.8.3 The open mixed paths

A path will be termed P_{OXI}, **open mixed path** if:

(1) Its edges consist of both external edges and internal edges.

(2) Its end-nodes are external nodes.

(3) Its end-nodes belong to P_{GX}.

(4) Its length is the minimum number of edges between the two external end-nodes.

(5) The end-nodes may have more than one mixed open paths.

7.9 The Network dimensions

In a connected network, any two nodes have at least one open path between them. **Define distance** to be the length of an open path between the two nodes in the connected network. Therefore, an isolated node has a zero distance. It has no edges connected to any other nodes. The distance between two adjacent nodes is one edge. **The minimum distance** between two, non-adjacent, nodes, is then the minimum length of the open path between the two nodes, and **the longest distance** between them is the maximum length of the open path between them.

7.9.1 The network Diameter

The network external closed path, P_{GX}, has been introduced in Section 7.4.4. If $(n_X/2) > n_I$ then the maximum length of an open internal path in the network is $n_I - 1$. The end-nodes of the open internal path are of external type. **Define** the network diameter to be the maximum value of the minimum distance between two nodes on P_{GX}. The following theorem determines the value of the network diameter:

Theorem (7.6) (The network Diameter)

If a network has an external closed path and $(n_X/2) > n_I$, then the network diameter is:

$$\text{Diameter} = \left\lceil \frac{m_X}{2} \right\rceil. \qquad (7.52)$$

Proof:

Let G be a network with m edges and n nodes, and let m>n.
Since $(n_X/2) > n_I$ then the maximum distance of an open internal path in the network is $n_I - 1$, and the maximum value of the minimum distance of an open external path between any two external nodes is $\geq (n_X/2) \geq n_I - 1$.
Since $p_{GX} = m_X = n_X$, as given by equation (7.34), then if m_X is even then the diameter, i.e. the maximum value of the minimum distance, between any two external nodes, is $m_X / 2$. If m_X is odd, then the diameter is $(m_X + 1)/2$.

■

7.10 The range of P_{GX}

Consider the set of networks having n nodes and m edges but in which the end-nodes for each edge are not specified. Such a network with unspecified edge locations will be termed G_u.

Since $p_{GX} = m_X = n_X$, as given by equation (7.34), then P_{GX} includes all the external edges of G_u. Since the minimum length a closed path can have is three edges, then the minimum length of P_{GX} is three edges. Let p_{GXmin} denote the minimum length of P_{GX}, i.e.

$$P_{GXmin} = 3.$$

Remark (7.3):

Every network with P_{GX} and with $n_I > 0$ can have a P_{GXmin}.

Let p_{GXmax} denote the maximum value that the P_{GX} can be.

In G_u, if $n = m$, then $c = 1$ and $g = n = m$, i.e. the maximum number of nodes or edges a cycle can have is $n = m$. Thus, if $n = m$, then $c = 1$ and the maximum length P_{GX} can have is: $p_{GXmax} = m_X = n_X = n$.

If $m > n$ then $c > 1$ and $g_{max} = n - 1$, i.e. $p_{GXmax} = m_X = n_X = n - 1$.

The set of many give different networks. They may give a network with $p_{GX} = p_{GXmin}$, or a network with $p_{GX} = p_{GXmin} + 1$ or more up to $p_{GXmax} = m_X = n_X = n$. Thus for each network p_{GX} varies from p_{GXmin} to p_{GXmax}, i.e.

$$p_{GXmin} \le p_{GX} \le p_{GXmax}. \tag{7.53}$$

7.11 The relation between P_{GX} and g_C

From Theorem (7.2), $g_C = 2m - n_X$; and from theorem (7.1), $p_{GX} = m_X = n_X$. Substituting p_{GX} instead of n_X yields an important relationship between the network external path, the circumference of the c cycles and the m, i.e.

$$g_C + p_{GX} = 2m. \qquad (7.54)$$

Since the value of 2m is constant for a given network, then

1.There is an indirect relationship between P_{GX} and g_C, i.e. as p_{GX} increases g_C decreases, and the opposite is true.

2.For each one value of p_{GX} the network has one and only one value of g_C. Thus, if $p_{GX} = p_{GXmin} = 3$, then $g_C = 2m - 3$. Let g_{Cmax} the maximum value g_C can be, then $g_{Cmax} = 2m - 3$. (7.55)

As p_{GX} is increasing by one edge, g_C is decreasing by one edge. Thus, if $p_{GX} = p_{GXmax} = m_X = n_X = n$, there will be no internal nodes, i.e. $n_I = 0$, then g_C has the minimum value it can be, i.e. $g_C = g_{Cmin} = 2m - n$. Let g_{Cmin} be the minimum value of g_C.

Then $g_{Cmin} = 2m - n$. (7.56)

Thus g_{Cmin} and g_{Cmax} define the range of g_C for a given network, i.e.

$$g_{Cmax} \geq g_C \geq g_{Cmin}. \qquad (7.57)$$

3. For each value of p_{GX} and g_C the network has several possible connections, and these connections are countable as given in the next section.

7.12 The relation between p_{GX} and g_{max}

In a network with $m_I \geq 1$, the maximum circumference of a cycle is $g_{max} = n - 1$, as given by equation (7.41). The conditions, at which a network has a cycle with $g_{max} = n - 1$, have been introduced in Section (7.6).

Since $p_{GXmax} = n$ and since $g_{max} = n - 1$ then a network can have only one cycle with $g_{max} = n - 1$. The network has many different connections, some of the connections may have one cycle with $g_{max} = n - 1$ and some connections may not have a cycle with $g_{max} = n - 1$, i.e. if $p_{GX} = p_{GXmax} = n$ then a network may have a cycle with $g_{max} = n - 1$, and all connections with $p_{GX} < p_{GXmax} = n$ have to have a cycle with $g_{max} = n - 1$. The opposite is not necessarily true, i.e. if $p_{GX} = p_{GXmin} = 3$ and has a cycle with $g_{max} = n - 1$, then not necessarily all connections with $p_{GX} > p_{GXmin} = 3$ have a cycle with $g_{max} = n - 1$. The basic configuration of a network introduced in Section 7.6 will be used to introduce the relation between p_{GX} and g_{max}.

Let n_{I2} be the number of internal nodes of degree two, and let n_{I3} be the number of internal nodes of degree three, then $n_I = n_{I2} + n_{I3}$.

If G_u has

$p_{GX} = m_X = n_X = c_{XI} = 3$,

One cycle with $g_{max} = n - 1$,

and $g_i = g_{min} = 3$ of the remaining (c-1) cycles then the network has only one of two possible configurations as shown in Figures 7.10 and 7.11.

The first possible configuration:

The first possible basic configuration of G_u is shown in Figure 7.10. It has:

(1) $p_{GX} = m_X = n_X = c_{XI} = 3$, i.e.

One external node of degree c; and

Two external nodes of degree three.

(2) It has $n_{I3} = c - 2$ nodes of degree three.

(3) It has $n_{I2} = 0$ nodes of degree two.

(4) One cycle with $g_{max} = n - 1$.

(5) Each cycle has three edges except one cycle,

i.e. $g_i = 3$ for $i = 1, 2, \cdots, c-1$.

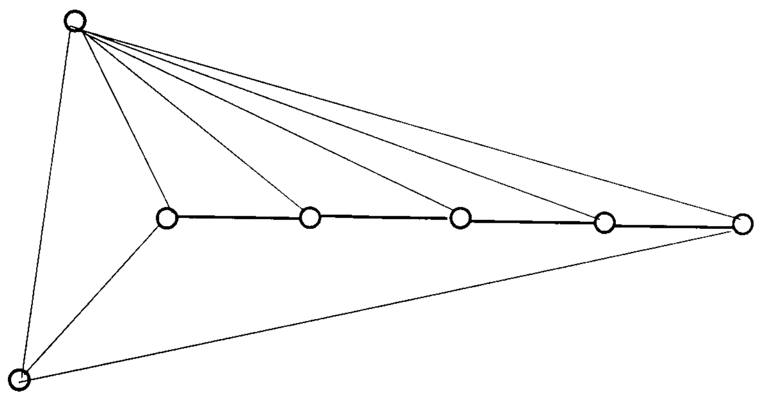


Figure 7.13 The first basic configuration of G_u

The number of internal cycles can be obtained, i.e.

$c_I = c - c_{XI}$.

For example, in Figure 7.13, $c_I = c - c_{XI} = 6 - 3 = 3$.

If $g_i = g_{min} = 3$ kept unchanged for $i = 1, 2, \cdots, c-1$ and p_{GX} of G_u is changed, then $g_c = g_{max} = n_I + 2 = n - 1$ will be destroyed, i.e. G_u will not have a cycle with $g_{max} = n - 1$, it may has a cycle with a circumference $\geq$ $g_{min} = 3$. The relationship between g_C and g_i is direct.

$g_C = g_1 + g_2 + \cdots + g_c$;

Let $g_i = g_{min} = 3$ for $i = 1,...,c-1$, and

let $g_c = g_{max}$

then the sum of the c circumferences is $g_C = 3(c-1) + g_{max}$.

Since $g_C + p_{GX} = 2m$, as given by equation (7.46), and since p_{GX} of G_u varies from $p_{GXmin} \le p_{GX} \le p_{GXmax}$, then

$$g_{max} = g_C - 3(c-1);$$

$$g_{max} = (2m - p_{GX}) - 3(c-1);$$

thus the relation between p_{GX} and the maximum circumference of a cycle in G_u is:

$$g_{max} = 3n - m - p_{GX}. \tag{7.58}$$

The second possible basic configuration:

The second possible basic configuration of G_u is shown in Figure 7.14. It has:

(1) $p_{GX} = m_X = n_X = c_{XI} = 3$, with

One external node of degree c; and

Two external nodes of degree three.

(2) Number of nodes of degree three is $n_{I3} = c - 2$.

(3) Number of nodes of degree two is $n_{I2} = n - n_X - n_{I3} = n - c - 1$.

(4) One cycle with $g_{max} = n - 1$.

(5) One cycle with $g_{c-1} = n_{I2} + 3$.

(6) $g_i = 3$ for $i = 1, 2, \cdots, c-2$.

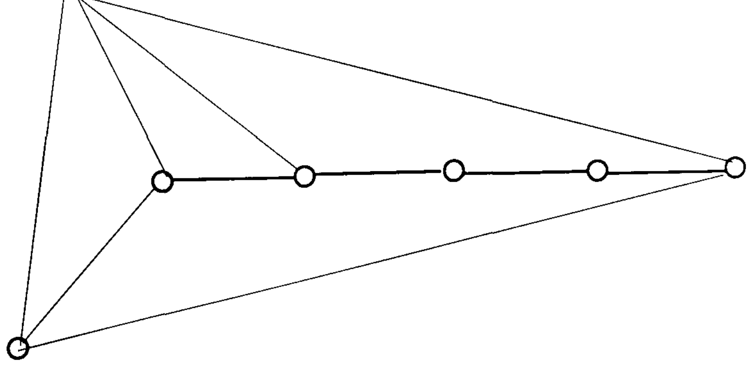


Figure 7.14
The second basic configuration of G_u

The number of internal cycles can be obtained, i.e.

$$c_I = c - c_{XI}.$$

For example, in Figure 7.14, $c_I = c - c_{XI} = 4 - 3 = 1$.

Let each internal cycle has three edges only, then the internal nodes of each internal cycle are of degree three, furthermore:

One internal cycle has two internal nodes of degree three,

Two internal cycles have three internal nodes of degree,

Thus there is a relationship between the numbers of internal cycles and internal nodes, i.e.:

$$c_I = n_{I3} - 1;$$

$$n_{I2} = n_I - n_{I3}.$$

Applying these relationships on Figure 7.14, then;

$$n_{I3} = c_I + 1 = 1 + 1 = 2;$$

$$n_I = n - n_X = 7 - 3 = 4;$$

$$n_{I2} = n_I - n_{I3} = 4 - 2 = 2;$$

$$m_I = m - m_X;$$

Thus $g_i = 3$ for $i = 1,...,c-2$, and

$$g_{c-1} = n_{I2} + 3 = 2 + 3 = 5$$

$$g_c = g_{max} = n - 1 = 7 - 1 = 6.$$

If $g_i = g_{min} = 3$ kept unchanged for $i = 1, 2, \cdots, c-2$ and p_{GX} of G_u is changed, then $g_c = g_{max} = n_I + 2 = n - 1$ will exist as long as $n_{I2} > 0$, but g_{c-1} will change, i.e. it will decrease, with the increase p_{GX} of G_u. Figures 7.15 to 7.18 show the variation in g_{c-1} and $g_c = g_{max}$ as p_{GX} is increasing.

If G_u has $p_{GX} = m_X = n_X = 4$, as shown in Figure 7.15

then

$$n_I = n - n_X = 3;$$

$$n_{I2} = n_{I2} - 1 = 2 - 1 = 1;$$

Thus

$$g_i = 3 \text{ for } i = 1,\ldots,c-2, \text{ and}$$

$$g_{c-1} = n_{I2} + 3 = 1 + 3 = 4$$

$$g_c = g_{max} = n - 1.$$

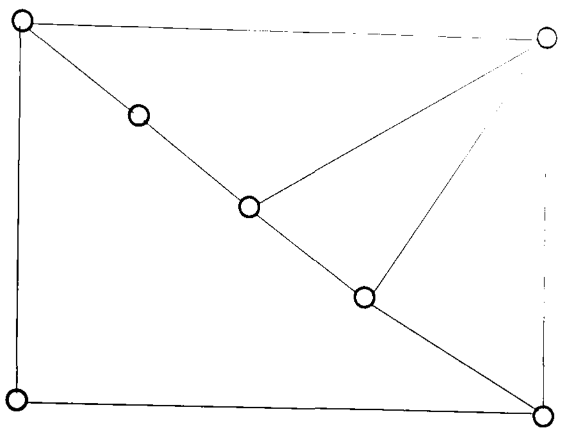


Figure 7.15 A network with $p_{GX} = 4$

If G_u has $p_{GX} = m_X = n_X = 5$, as shown in Figure 7.16,

then

$$n_I = n - n_X = 2;$$

$$n_{I2} = n_{I2} - 1 = 1 - 1 = 0;$$

Thus

$$g_{c-1} = n_{I2} + 3 = 0 + 3 = 3;$$

i.e.

$$g_i = 3 \quad \text{for } i = 1,\ldots,c-1;$$

and $g_c = g_{max} = n - 1$.

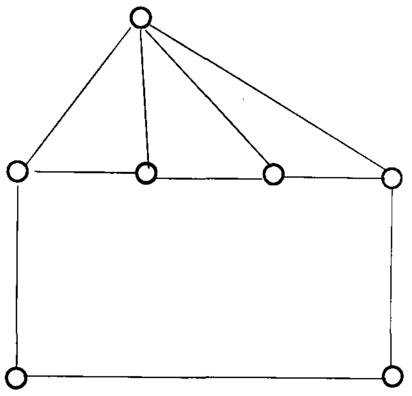


Figure 7.16 A network with $p_{GX} = 5$

If G_u has $p_{GX} = 6$ and $g_i = 3$ for $i = 1,\ldots,c-1$; and one of the internal nodes of degree three is an external node as shown in Figure 7.17, i.e.

If $\quad p_{GX} = m_X = n_X = 6,$

and $\quad g_i = 3$ for $i = 1,\ldots,c-1,$

then

$$n_I = n - n_X = 1;$$

$$n_{I2} = 0$$

$$n_{I3} = n_I - n_{I2} = 1 - 0 = 1;$$

$$g_c = g_{max} = n - 2.$$

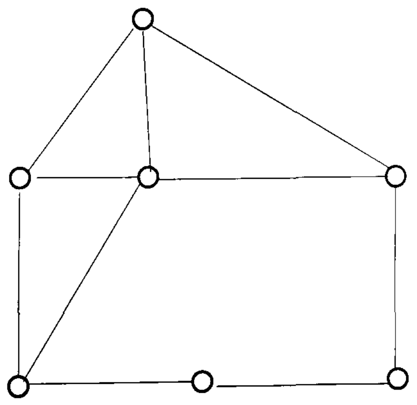


Figure 7.17 A network with $p_{GX} = 6$

If G_u has $p_{GX} = n$ then the last internal node of degree three will be an external node as shown in Figure 7.18, i.e.

If $p_{GX} = m_X = n_X = n = 7$

then

$n_I = 0$

$g_i = 3$ for $i = 1,\ldots,c-1$;

$g_c = g_{max} = n - 3$.

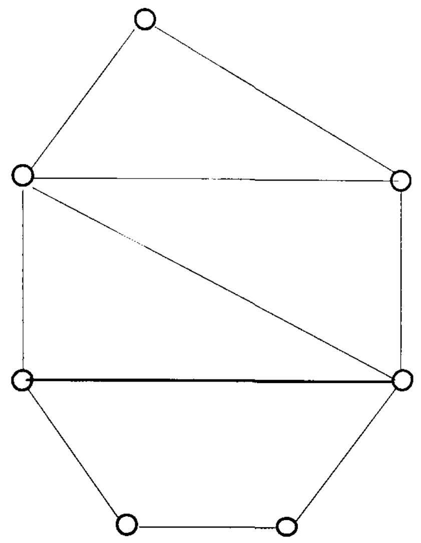


Figure 7.18 A network with $p_{GX} = 7$

Thus for the second basic configuration of a network with $n_{I2} > 0$

$g_C = 2m - p_{GX}$;

$g_{max} = n - 1$;

and

$g_{c-1} = g_C - g_{max} - 3(c-2)$;

or

$$g_{c-1} = 2n - m + 4 - p_{GX}. \qquad (7.58)$$

The difference between the first and the second basic configuration is n_{I2}. If $n_{I2} = 0$, then G_u has the first configuration, and if $n_{I2} > 0$ then G_u has the second configuration.

7.13 Network classifications

The network external closed path P_{GX} is a natural property exists in every connected network. The relation between p_{GX}, m and g_C has been given by equation (7.46), in Section (7.11) and the relation between p_{GX} and g_{max} has also been described in Section (7.12). Every connected network has one and only one P_{GX} as described in Section 7.4.1 and proved by Theorem (7.1). Since P_{GX} has a range of values as given by equation (7.45). Then the P_{GX} property can be used to classify the different sets of networks that G_u may have. Each subset of networks of G_u that has only one value of P_{GX} represents one class. Then its possible to find the basic configuration of each P_{GX}.

Let $G_{p_{GX}}$ represents one subset of networks of G_u which has only one value of P_{GX}. Then $G_{p_{GX}}$ has the same range as the range of P_{GX}. For example, if $7 \geq p_{GX} \geq 3$ G_3 then $G_{p_{GX}}$ has (7-3+1)=5 classes from G_3 to G_7. Finding the a class of $G_{p_{GX}}$ for a certain value of P_{GX} depends on finding the basic configuration first.

Then the parameters of the basic configuration of $G_{p_{GX}}$ for every value of P_{GX} can be obtained.

For example, let $p_{GX} = 3$ then the parameters of G_3 are:

$p_{GX} = m_X = n_X = 3$;

$g_C = 2m - 3$

$n_I = n - n_X$;

$m_I = m - m_X$;

$c_{XI} = 3$;

$c_I = c - 3$;

$g_{max} = n - 1$.

Let n= 14 nodes and m = 20 edges, and let $p_{GX} = 3$ then the parameters of G_3 are:

$p_{GX} = m_X = n_X = 3$.

$g_C = 2m - 3 = (2).(20) - 3 = 37$.

$n_I = n - n_X = 14 - 3 = 11$.

$m_I = m - m_X = 20 - 3 = 17$.

$c = m - n + 1 = 20 - 14 + 1 = 7$.

$n_{I3} = c - 2 = 5$.

$n_{I2} = n_I - n_{I3} = 11 - 5 = 6$.

$c_{XI} = 3$;

$c_I = c - 3 = 7 - 3 = 4$.

Since $n_{I2} > 0$ then G_3 has the second configuration, then

$$g_{c\text{-}1} = 2n - m + 4 - p_{GX}. \qquad (7.59)$$

$g_{c\text{-}1} = (2).(14) - 20 + 4 - 3 = 9$.

$g_c = g_{max} = n - 1 = 14 - 1 = 13$.

$g_i = g_{min} = 3$ for $i = 1, 2, \cdots, 5$.

Knowing parameters of the basic configuration of G_3, it is possible to find the parameters of the basic configuration of any value of P_{GX} in the possible range of p_{GX}.

7.14 Identifying the different possible connections

If P_{GX} of a network is known, then it is easy to identify all possible connections that this network may have. Each P_{GX} identifies a set of connections. The identification is based on using P_{GX} and the basic configuration of the network as introduced in Section 7.6.

If the network has the first basic configuration, then the number of possible connections is:

$$z = g_{max} - g_{min}. \qquad (7.60)$$

The z possible connections can be obtained as follows: let $C_{P_{GX},z}$ be the set of c circumferences of the basic configuration of the network for p_{GX} where $p_{GXmin} \geq p_{GX} \geq p_{GXmin}$, then,

$$C_{P_{GX},z} = \{g_i = g_{min},\ g_j = g_{min} + 1, g_c = g_{max} - z + 1\}, \qquad (7.61)$$

for

$i = 1, \cdots, c - z;$

$j = c - z, \cdots, c - 1;$

with $g_{max} = 3n - m - p_{GX}. \qquad (7.62)$

Each z may have w different possible connections.

Thus, if $p_{GX} = 3$, then the z the number of possible connections are:

$C_{3,1} = \{g_i = g_{min}, g_{max}\}$; for $i = 1, \cdots, c - 1$

$C_{3,2} = \{g_i = g_{min}, g_{c-1} + 1, g_{max} - 1\}$ for $i = 1, \cdots, c - 2$

$C_{3,3} = \{g_i = g_{min}, g_{c-2} + 1, g_{c-1} + 1, g_{max} - 2\}$ for $i = 1, \cdots, c - 3$

.

.

$C_{3,z} = \{g_i = g_{min}, g_j = g_{min} + 1, g_{max} - z\}$ for $i = 1, \cdots, c - z$, and $j = z + 1, \cdots, c - 1$

if $p_{GX} = 4$

$C_{4,1} = \{g_i = g_{min}, g_{max}\}$; for $i = 1, \cdots, c - 1$

If a network has the second basic configuration, then the z possible connections can be obtained by using the following procedure:

let $C_{p_{GX},z}$ be the set of c circumferences of the second basic configuration of the network for p_{GX} where $p_{GXmin} \geq p_{GX} \geq p_{GXmin}$, then,

$$C_{p_{GX},1} = \{g_i = g_{min},\ g_{c-1}, g_c = g_{max}\}; \tag{7.63}$$

for

$$i = 1, \cdots, c-2$$

with $g_{c-1} = 2n - m + 4 - p_{GX}$; (7.64)

and $g_{max} = 3n - m - p_{GX}$. (7.65)

Procedure:

```
        $p_{GX} = 3$;
        $z = 1$;
        $C_{p_{GX},z} = [g_1, g_2, \cdots, g_{c-2}, g_{c-1}, g_c]$;
let     $g = [g_1, g_2, \cdots, g_{c-2}, g_{c-1}, g_c]$;
        $z_1 = g_{c-1} - g_{c-2}$;
        if $z_1 > 1$
10              for $j = c-2:-1:1$
                        $z_1 = g_{c-1} - g_j$;
                                if $z_1 > 1$
                                $g_{c-1} = g_{c-1} - 1$;
                                $g_j = g_j + 1$;
                                $z = z + 1$;
                        $C_{p_{GX},z} = [g_1, g_2, \cdots, g_{c-2}, g_{c-1}, g_c]$;
                end;
        end;
$z_2 = g_c - g_{c-1}$;
if $z_2 > 1$
        $g_c = g_c - 1$;
        $g_{c-1} = g_{c-1} + 1$;
        $z = z + 1$;
        $C_{P_{GX},z} = [g_1, g_2, \cdots, g_{c-2}, g_{c-1}, g_c]$;
Go to 10;
end.
```

Each $C_{P_{GX},z}$ connection has different possible configuration, i.e. if P_{GX} of a network is known, and g_i for $i = 1,2,...,c$, is also known, then the c cycles may have different possible configurations.

7.15 The relation between P_{GX} and the mixed-cycles

In a network as the one described in Theorem (7.1) and (7.2), c, the number of cycles is given by:

$$c = c_{XI} + c_I;$$

Each c_{XI}, mixed-cycle, as introduced in Section 7.7, has at least one external edge in P_{GX}. Thus if each c_{XI} cycle has exactly one external edge in the network, then

$c_{XI} = p_{GX}$; this is the maximum number of c_{XI} in the network.

and $c_I = 0$;

and $n_I > 0$;

and all the n_X nodes are of type V_{X2I}, i.e. of degree three;

and all the c_{XI} are of type c_{XI1M}.

If $c = 1$, then c is of external type.

If $c = 2$, then they are mixed-cycles, i.e. the minimum number of mixed-cycles in a network is 2 cycles. Thus, $p_{GX} \geq c_{XI} \geq 2$.

Since $c = m - n + 1 = c_{XI} + c_I$, is constant, then if c_{XI} increases then c_I must decrease to give c.

A mixed-cycle has three types as given in Section 6.7.1, these types are c_{XI0T}, c_{XI1M} and c_{XI2M}.

The first basic configuration has $c_{XI} = p_{GX}$ always.

If the second basic configuration has one cycle with $g_i = g_{max} = n - 1$, then $c_{XI} = 3$. If $g_i = g_{max} = n - 1$ decreases by one, c_{XI} increases by one mixed-cycle until $c_{XI} = p_{GX}$. A P_{GX} with one cycle has $p_{GX} \le g_i < g_{max} = n - 1$ may have $c_{XI} \le p_{GX}$. Thus the c cycles can have different position in the network.

7.16 Number of possible connections

There are many different possible connections of a network with n nodes and m edges. For example, if m=n, then there is only one cycle.

There are $\sum_{i=1}^{n}(n - i)$ possible ways to have one cycle with m=n.

If c=2, then the first cycle has $\sum_{i=1}^{n}(n - i)$ possible ways, and the second cycle has $(n - 3)$ possible ways. If c=3 then there are $\sum_{i=1}^{n}(n - i) + 2(n - 3)$.

In general if c>1, then

$$\text{the number of possible connection} = \sum_{i=1}^{n}(n - i) + (c - 1)(n - 3).$$

7.17 The network entity

Having introduced the edge state phenomenon and the different parameters and relations associated with a graph of a network, this chapter concludes by defining the network entity, which describes and identifies the minimum set of connections having the same descriptions.

An entity is a set of parameters describing an object such that the object or a class of the object can be identified. An example of an entity is a home address. The home address consists of the following parameters:

The house/apartment number;

The street name;

The city name;

The county name;

The zip code;

The country name.

If any one of these parameters is changed the mail may not be delivered correctly.

A connected graph is an object. It has a set of parameters which define its entity. In this section, the analysis of defining the network parameters proceeds from the most general case of a network without restrictions, and then applies the network natural properties as natural restrictions that have to be used as parameters to define the network entity:

(1) V is the set of nodes with n the number of elements in V.

(2) E is the set of edges with m the number of elements in E.

(3) If V is considered alone, i.e. without considering the edges between the n nodes, then V form an isolated set of nodes, every node has no edges.

(4) If the n nodes are connected by the m edges without restrictions, then there are many possible ways of different connections as given in Section 7.14.

(5) If one restriction is applied on the connection, then the number of possible ways of connection will reduce. It will be less than the number of possible ways without restrictions. The more restrictions applied, the more the number of possible ways of connections will be

reduced.

(6) **The mechanism of connection:** connecting an edge to a node defines not only the edge and the node types, but also it defines the induced relationship, such as the degree of the node and the cycle relationship. For example, if an isolated node is connected by an edge coming from an internal node, then the isolated node is changed to be a one-degree node, and the edge is classified as an edge of SI type. Furthermore, if another edge coming from an external node is connected to the previous one-degree node, then, the attribute of the one-degree node changes to be an internal node. From the degree side, the connection of the first edge then the second edge changes the node degree from zero to one to two, and it changes the node type from an isolated node to a one-degree node to an internal node, and finally it changes the network structure by adding a new cycle of a certain class. Thus, adding (or deleting) an element (an edge or a node) to the network changes the entity of the element accordingly, and this change, consequently, changes the network parameters and relations. So that, every element, in each set, has a very well defined class or sub-class.

(7) The restrictions that will be applied on the connections are the edge and node natural types that induced through mechanism of connection and the induced relationships.

(8) **The node and edge restriction**: V is the set of nodes, and E is the set of edges. Each set is classified, as given in Sections 7.2 and 7.3, through the mechanism of connection into four types, i.e. $V = \{V_X \oplus V_I \oplus V_B \oplus V_S\}$ and $E = \{E_X \oplus E_I \oplus E_B \oplus E_S\}$ with $n = |V|$ the number of elements of V, and $m = |E|$ and the number of E. If a

network has the same description as the one described by Theorems (7.1) and (7.2), and if $E_B = E_S = V_B = V_S = \{\Phi\}$ then V reduces to $V = \{V_X \oplus V_I\}$ and E reduces to $E = \{E_X \oplus E_I\}$, which in turn must reduce the number of possible connections. For example, if a network has n nodes, unclassified, and there is one edge to be connected between any two nodes, then there are n.(n-1) possible ways of connecting the edge. If n and m are classified as internal and external, and if the one edge is of external type then the external edge has to be connected between external nodes only. Thus, the edge/node type restriction reduces the number of possible ways of connecting the one external edge to n_X. If there are m_X edges to be connected between n_X nodes, then the number of possible ways of connection reduces to one.

(9) **The network external path restriction**: If a network has P_{GX}, then $p_{GX} = n_X = m_X$. Many other relations, related to P_{GX}, can be obtained such as $n_I = n - n_X$ and $m_I = m - m_X$. The range of P_{GX}, of a network, is $3 \le p_{GX} \le n$. Since range of P_{GX} is deterministic, then the related functions are deterministic, such as $g_{c\text{-}1} = 2n - m + 4 - p_{GX}$ and $g_C = 2m - p_{GX}$. Determining one P_{GX} of a network out of the range of P_{GX}, not only a reduction from n to n_I and from the m to m_I. Thus, reducing the number of different possible connections to the internal edges and nodes. There are m_I edges to be connected internally between the n_X nodes and n_I nodes such that the network has c cycles. There is $n_I(n_X + 1)$ possible ways to connect one internal edge, and there are $n_I(n_X + 1) - 1$ possible ways to connect two internal edges. Thus there are $n_I(n_X + 1) - m_I$ possible ways to connect m_I internal edges.

(10) **The circumference restriction**: The number of cycles, in all different possible connections, with or without restrictions, is constant. But the c circumferences of the c cycles are not equal in all different possible connections. As explained in Section 7.10, g_C has a direct relation with P_{GX}, each P_{GX} has one and only one g_C. Each combination of P_{GX} and g_C has z different possible connections as given in Section 7.13. At each z^th connection the c circumferences are fixed. But, the internal edges of the c cycles can be connected in different away so that the sum of the c circumferences is always g_C, i.e. $g_C = g_1 + g_2 + \cdots + g_c = 2m - p_{GX}$. The circumference restriction reduces the number of different possible connections to one connection with one P_{GX} and one g_C and one z configuration, which has fixed value of g_i for $i = 1, 2, ..., c$.

(11) **The open internal paths restriction**: The internal edges has to be change such that $g_C = g_1 + g_2 + \cdots + g_c = 2m - p_{GX}$. If the connection of one internal edge is changed from (v_{I1}, v_{I2}) to (v_{I1}, v_{I3}) the length of one or more of the open internal paths will change. Thus, if a connection has one P_{GX} and one g_C and one z configuration, i.e. with fixed value of g_i for $i = 1, 2, ..., c$, then if one or more internal edges have been changed, then the length of one or more of the internal paths will change. Thus, L_P, the total length of the $n_{P_{OI}}$ open internal paths will change. If after the change, L_P did not change, then the two connections are similar to each other. Thus, applying $n_{P_{OI}}$ and L_P restrictions reduced the different possible connections to small number of similar connections.

(12) Thus, these parameters, the natural restrictions, are used to form the network entity, i.e. $G = \{V, E, P_{GX}, C_{P_{GX},z}, P_{GI}\}$.

(13) The network entity can describe one or a set of similar connections of a network.

Example (7.2): To identify the network entity

The network in Figure 7.19 has $m_X = n_X = 6$; $n_I = 3$; $m_I = 7$.

Thus $c = (m_X + m_I) - (n_X + n)_I + 1 = (6+7) - (6+3) - 1 = 5$.

The $C = \{g_1, g_2, g_3, g_4, g_5\}$ ={3, 3, 4, 4, 6}

There are few different connections that can give a network with these data. The open internal paths of Figure 7.19 are:

$P_{OI}(v_{X1}, v_{X2}) = \{e_{I2}, e_{I1}\} = 2$;

$P_{OI}(v_{X1}, v_{X3}) = \{e_{I2}, e_{I5}\} = 2$;

$P_{OI}(v_{X1}, v_{X5}) = \{e_{I2}, e_{I3}, e_{I6}, e_{I7}\} = 4$;

$P_{OI}(v_{X1}, v_{X6}) = \{e_{I2}, e_{I3}, e_{I4}\} = 3$;

$P_{OI}(v_{X2}, v_{X3}) = \{e_{I1}, e_{I5}\} = 2$;

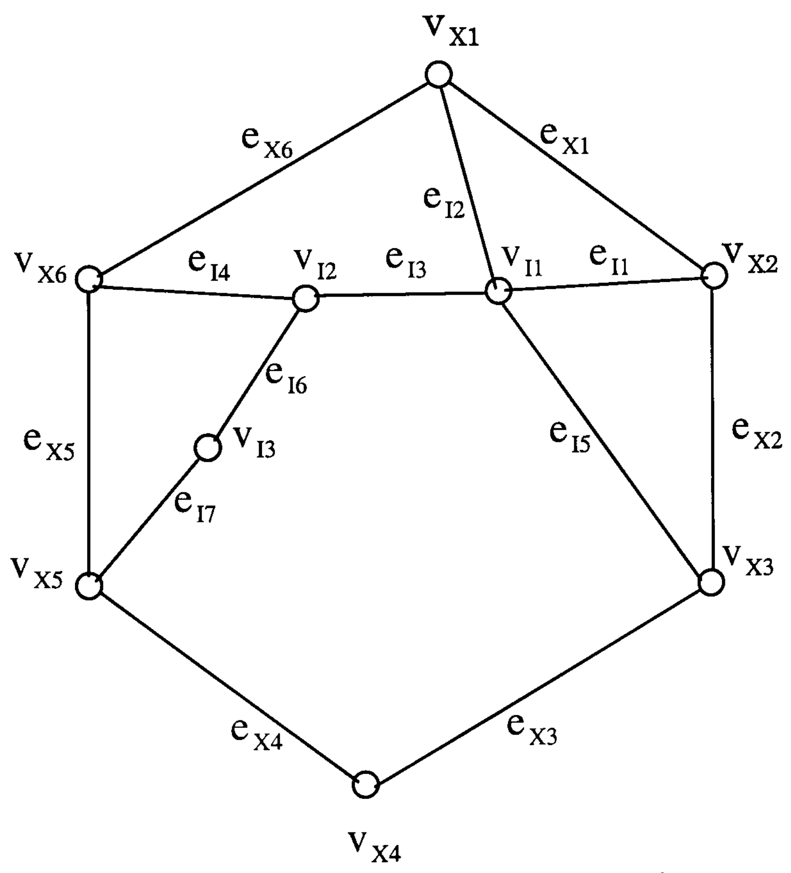


Figure 7.19 A network used to define the entity

$P_{OI}(v_{X2}, v_{X5}) = \{e_{I1}, e_{I3}, e_{I6} e_{I7}\} = 4$

$P_{OI}(v_{X2}, v_{X6}) = \{e_{I1}, e_{I3}, e_{I4}\} = 3$

$P_{OI}(v_{X3}, v_{X5}) = \{e_{I5}, e_{I3}, e_{I6}, e_{I7}\} = 4$

$P_{OI}(v_{X3}, v_{X6}) = \{e_{I5}, e_{I3}, e_{I4}\} = 3$

$P_{OI}(v_{X5}, v_{X6}) = \{e_{I7}, e_{I6}, e_{I4}\} = 3$

Thus the total number of open internal paths can be written as follows:

$$3(2) + 4(3) + 3(4) = 30$$

if P_{GX}, C are kept constant and one of the internal edges change its position, then set of open internal paths will change to describe the new connection. Thus the set of {V, E, P_{GX}, $\{P_{OGI}\}$} describe one and only one connection.

Chapter 8

Using the edge state phenomenon to partition the network

8.1 General

The edge state phenomenon has been introduced in Chapter 7 in order to define some new properties of the network, which are useful in the theoretical development. In this chapter, the edge phenomena and some of the induced network properties are used to introduce a new approach for solution of the network-partitioning problem. The new approach is introduced for a class of network connections without bridge edges. The problems of: (i) establishing the existence of a balanced k-partitioning of a network; and (ii) obtaining such a partitioning; are known [25] to be NP-hard problems. The method to be described offers a new heuristic approach to compact computational solution of (ii), and gives insight into (i). Partitioning a network is an operation in which a set of the network edges is cut. Those edges are termed **cut edges**, the end nodes of the cut edges are termed **the boundary nodes**, and the remaining nodes in each part consists of internal nodes and external nodes are termed **the internal part nodes**. The goal of partitioning a network into k parts is to balance the number of boundary nodes with the number of the part nodes in the k partitions.

A cut-line is a line used to cut a set of edges of the network. The cut-line concept is introduced in Section 8.2. The concept is introduced gradually from partitioning a network with one cycle by one cut line, in Section 8.3, to partitioning a network with c>1 cycles by k-1 cut lines, in Section 8.4. Cut lines are not allowed to cross each other, thus the relationship between cut lines are introduced.

The external close path of a network and the open internal paths are all data obtainable from the network.

In a network with an external close path, **the starting edge, the ending edge** and the **route** of each cut-line are defined in Section 8.5. The starting edge and the ending edge of a cut-line are always of external type. Thus the end nodes of the starting edge and the ending edge are classified as **external boundary nodes**. Two open paths determine the cut line route. While the cut line is proceeding from one internal edge to another internal edge, it defines the **internal boundary nodes**.

The **I-I cut-line**, which partitions the external close path into k equal or balanced parts, is defined in Section 8.7.

The open internal paths are used not only to specify the cut line route but also to find the internal nodes in each part.

8.2 The cut-line concept

Let P be an open path consisting of n nodes and n-1 edges as defined in Section 7.4.1 and let e be one of the edges in P. Let a cut operation is performed on e by a line so that the cut operation partitions e by disconnecting e from its end nodes. Then it can be said that the line partitions e into two separate parts, each part has one node and it partitions P into two open paths P1 and P2, as shown figure (8.1a). The line is termed **the cut-line**, the edge that has been cut is termed **the cut-edge** and the end nodes of the cut-edge are termed **the boundary nodes**.

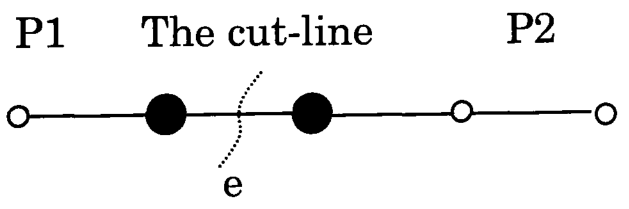


Figure (8.1a) One cut-line

○ A node

● A boundary node

.... A cut-line

The two end nodes of P and the two boundary nodes of the cut-edge are the four end nodes of the new two open paths P1 and P2, two end nodes for each open path.

Thus, one cut-line partitions an open path into two open paths by cutting only one edge from P. Accordingly, two cut-lines partition P into three open paths by cutting two edges only as shown in Figure 8.1b.

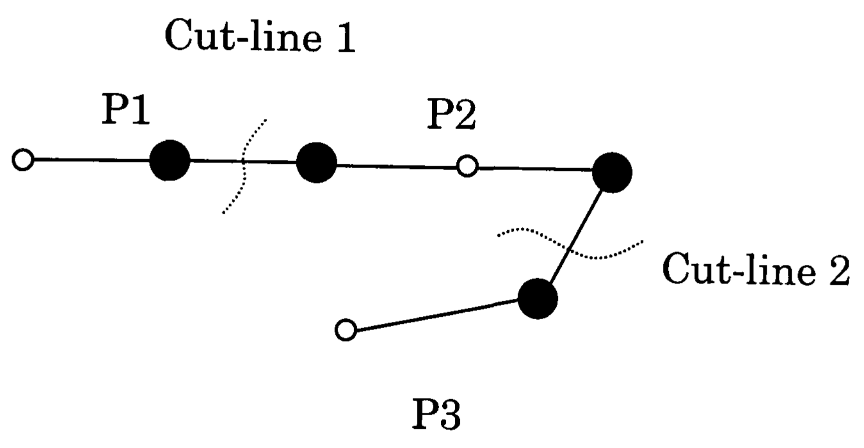


Figure 8.1b Two cut-lines

Remark (8.1)

If a cut-line cuts k edges from an open path, then the open path is partitioned into k+1 open paths

8.3 Partitioning a cycle

A cycle in a network, as defined in Section 7.4.2, is a closed path with minimum number of edges. Partitioning a cycle into two or more open paths involves using one or more cut-lines. The necessary conditions of partitioning a cycle into two or more open paths are introduced in Section 8.3.1.

8.3.1 Partitioning one cycle into two open paths

If a cut-line cuts one edge from a cycle, then the cycle becomes an open path. The end nodes of the open path are the boundary nodes of the cut-edge. If the cut-line cuts another edge in the cycle, then the cycle is partitioned into two separate open paths. Each open path starts and ends at a boundary node, i.e. the boundary nodes of the two cut-edges are the ending nodes of the two open paths. Thus, if one cut-line cuts exactly two edges from a cycle, then the cut-line partitions the cycle into exactly two open paths, as shown in Figure 8.2.

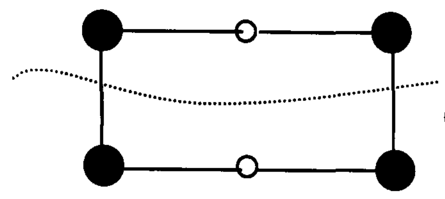


Figure 8.2 One cut line cuts two independent cut edges
There are four boundary nodes.

The two edges are cut-edges, and the partitioned cycle is termed the **cut-cycle**. The two cut-edges are either dependent edges or independent edges. If the two cut-edges are independent, then there are four boundary nodes, as shown in Figure 8.2, and if the two cut-edges are dependent, then there are three boundary nodes as shown in Figure 8.3.

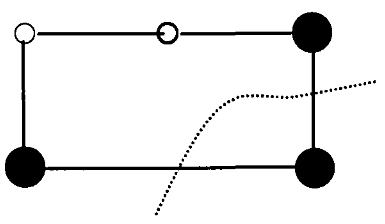


Figure 8.3
One cut line cuts two dependent cut edges
There are three boundary nodes.

Remark (8.2)
If one cut-line cuts only two edges from a cycle, then cut-line partitions the cycle into two open paths.

8.3.2 Dependent and independent cut-lines

Since one cut-line partitions a cycle into two separate open paths by cutting only two edges from the cycle, then one of the cut-edges is termed **the starting-edge of the cut-line** and the other cut-edge is termed **the ending-edge of the cut-line.**

If a cycle is partitioned by two or more cut-lines such that each cut-lines cuts only two edges, then some of the cut-lines have the same starting-edge or the same ending-edge and some have different starting edges and different ending edges.

Definition:

If the two cut-lines have the same starting edge or the same ending edge then **the two cut-lines are said to be dependent**.
If the two cut-lines have different starting edges and different ending edges, then **the two cut-lines are said to be independent**.

The dependent and independent cut-lines principle is used to partition a cycle into k>2 open paths as explained in Section 8.3.2.

8.3.3 Partitioning one cycle into three open paths

Since one cut-line partitions a cycle into two open paths by cutting only two edges from the cycle, then one cut-line is not sufficient to partition the cycle into three open paths. Therefore, it is necessary to have two cut-lines and the two cut-lines must be dependent. One cut line partitions the cycle into two open paths, say P1 and P2, by cutting only two edges. The second dependent cut-line partitions only one of the two open paths, say P2, into two open paths, say P21 and P22, by cutting only one edge from P2. Thus, the two dependent cut-lines have only three cut-edges. The three cut-edges have three different types of connection, either all of them are dependent or all of them are independent or one cut-edge is independent and two cut-edges are dependent. Figures 8.4a, 8.4b and 8.4c show all possible connections of the cut-edges. Figure 8.4a shows the two dependent cut-lines and three cut-edges, and the three cut edges are independent. Therefore there are six boundary nodes, i.e. (2 boundary nodes per cut edge x 3 cut-edges).

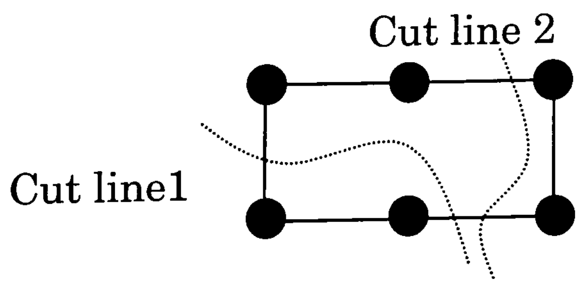


Figure 8.4a

Two dependent cut-lines cut three independent edges

Figure 8.4b shows two dependent cut-lines and three cut-edges. Two of the cut-edges are dependent and the third cut edge is independent. Therefore there are five boundary nodes, i.e. (2 nodes per cut-edge x 2 dependent cut-edges - 1) + (2 nodes per cut-edge x 1 independent cut-edge).

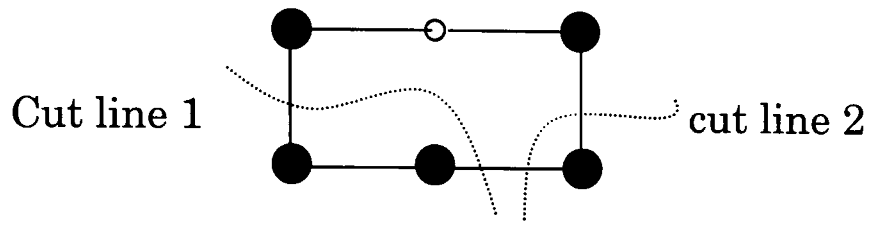


Figure 8.4b

Two dependent cut-lines and three cut-edges, two dependent and one independent

Figure 8.2c shows the two dependent cut-lines and three cut-edges. The three cut-edges are dependent, one cut edge is common between the other two cut edges. Therefore there are four boundary nodes, i.e. (2 nodes per cut edge x 3 dependent cut edges – 1 node from each two dependent cut-edges x 2 set of dependent cut-edges).

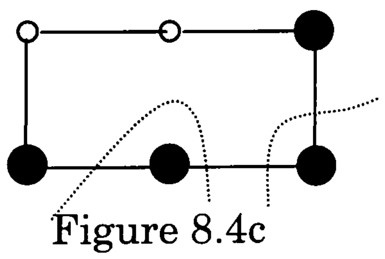


Figure 8.4c

Two dependent cut lines cut three dependent edges

Remark (8.3)

Every two dependent cut lines partition the one cycle into three parts by cutting only three cut-edges. The number of boundary nodes depends on the type of connection these edges have.

8.3.4 Partitioning one cycle into four open paths

If the circumference of a cycle is equal or greater than k (=4), then the cycle can be partition into four open baths by either dependent cut-lines or by independent cut-lines.

If the cut-lines are dependent, then three (i.e. k-1) dependent cut-lines are needed to partition the cycle into four open paths. The three cut-lines cut four edges. The four cut-edges can be dependent or independent as shown in figure 8.5.

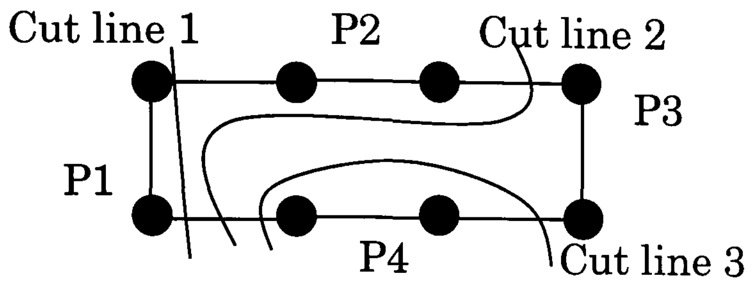


Figure 8.5

Three dependent cut-lines and four cut-edges

If the cut-lines are independent, then two cut-lines partition the cycle into four open paths by cutting four edges. The first cut-line partitions the cycle into two open paths, as described in Section 8.3.1. The second cut-line partitions one of the resultant two open paths into three open paths by cutting two edges as described in Section 8.2. Thus, the two independent cut-lines partition the cycle into four open paths by cutting four edges as shown in Figures 8.6a to 8.6d. The four cut-edges can be independent or dependent or combination of dependent and independent cut-edges.

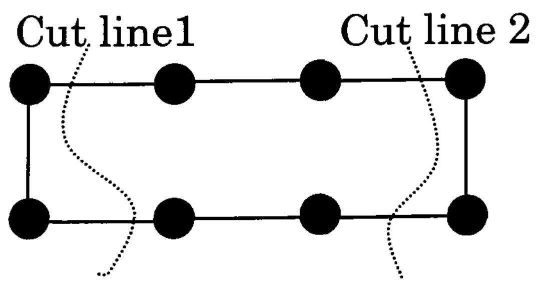


Figure 8.6a

Two independent cut-lines

cut four independent cut edges

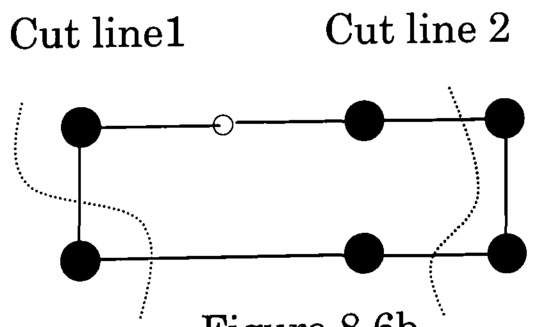


Figure 8.6b

Two independent cut-lines

cut three dependent cut-edges

and one independent edge

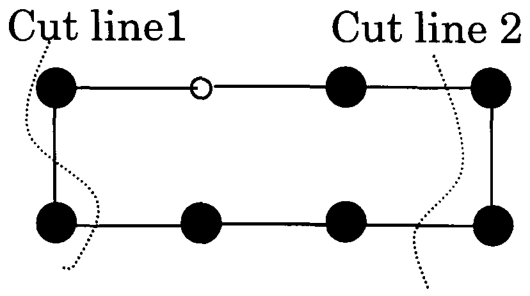


Figure 8.6c
Two independent cut lines
cut two dependent cut edges
and two independent cut edges

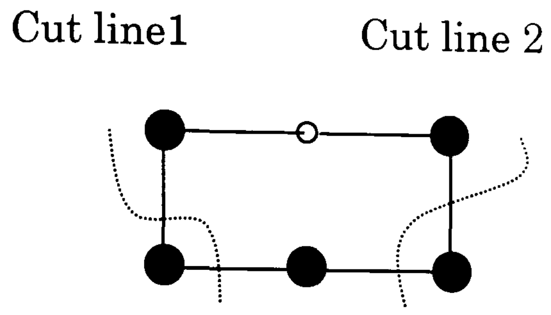


Figure 8.6d
Two independent cut lines
cut four dependent cut-edges

Remark (8.4)

A one cycle can be partitioned into four open paths by either three dependent cut-lines or two independent cut-lines.

8.3.5 Partitioning one cycle into k open paths

In general, a cycle can be partitioned into k open paths by dependent or independent cut lines, or by combination of them such that each cut line cuts two edges. If k is even, then, the cycle can be partition by k-1 dependent cut lines, or by k/2 independent cut-lines.
If k is odd and k>3, then the cycle can be partition by two dependent cut-lines and (k-2)/2 independent cut lines.

8.4 Partitioning more than one cycle

Two cycles may have one or more internal edges between them. Partitioning more than one cycle by one or more cut-line is the same as partitioning one cycle.

8.4.1 Partitioning two cycles with an internal edge

Two cycles have one or more internal edges. If a cut-line is to partition the two cycles into two parts, then, according to Remark (8.2), the cut-line has to cut two edges from each cycle. If the cut-line cuts only two external edges from any cycle, then the cut-line will not partition the second cycle. Thus, to partition the two cycles the cut-line must cut two edges from each cycle, i.e. the internal edge and one external edge from each cycle. The result is two open paths as shown in Figure 8.7.

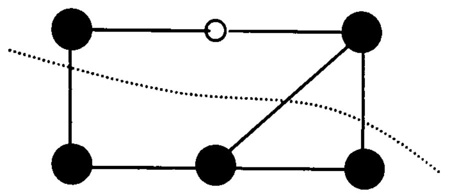


Figure 8.7 Partitioning two cycles by one cut-line

Therefore, each cut line has to cut three edges (two external edges and one internal edge) to partition two cycles into two open paths.
Also, one cut-line has to cut five edges (two external edges and three internal edges) to partition four cycles into two open paths.

Remark (8.5)
In general, if there are c cycles, and there is one internal edge between each two cycles then one cut-line partitions the c cycles into two open paths by cutting those internal edges and two external edges.

8.4.2 Using dependent cut-lines

Remark (8.3) establishes that two dependent cut-lines partition one cycle into three open paths. The same rule can be used to partition two or more cycles into three parts, each part is an open path.
Figure 8.8a shows two cycles with one internal edge between them, and Figure 8.8b shows two cycles with more than one internal edge. To partition the two cycles into three parts two dependent cut-lines must be used. If one of the cut-lines cuts two external edges and one internal edge, then the two cycles are partitioned into two parts. Since the two cut-lines are dependent, then the two cut-lines have one or more common edges.

The second
cut-line

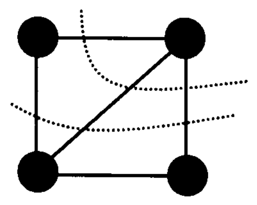


Figure 8.8a

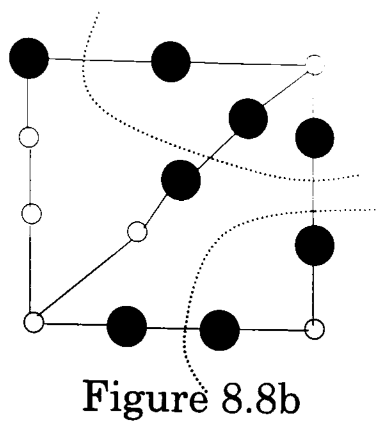


Figure 8.8b

Partitioning two cycles by two dependent cut lines

Thus the second cut-line has two dependent edges as shown in Figure 8.8a, and it has one common edge as shown in Figure 8.8b. The two cut lines cut four edges. The four cut-edges may be dependent as in Figure 8.8a, and they might be independent as shown in Figure 8.8b.

If the four cut edges are dependent, then there are four boundary nodes. If one cut edge is independent and three are dependent, then there are six boundary nodes. If two cut edges are independent and two are dependent, then there are seven boundary nodes. If all cut edges are independent, then there are eight boundary nodes.

Remark (8.6)
Two dependent cut lines partition two cycles into three parts by cutting only four edges.

8.4.3 Using independent cut-lines

Let the circumference of each cycle be $g_i = g_{\min} = 3$, $i = 1,2$, and let $m_I = 1$, then $m = 5$. If the two cut-lines do not cut the internal edge, then each cut line cuts two external edges from each cycle. Therefore the two cycles will be partitioned into three parts as shown in Figure 8.9.

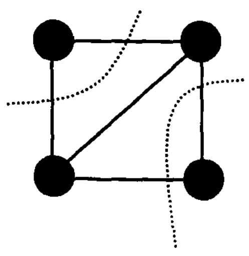


Figure 8.9 Partitioning two cycles by two independent cut-lines

If one cut-line partitions the two cycles it will cut three edges, the internal edge, and two external edges, one edge from each cycle. Since the other cut line is independent, it must cut different edges. This implies that the size of one of the cycles must be more than the three edges, to have another independent cut line as shown in Figures 8.10a and 8.10b. The second cut line cuts another set of edges entirely different from the cut edges that have been cut by the first cut-line. It cuts either another internal edge and two other external edges from the two cycles as shown in Figure 8.10a or it cuts two external edges from one cycle as shown in Figure 8.10b. If two internal edges have been cut, then the two cycles will be partitioned into five parts, if not, then the two cycles are partitioned into four parts.

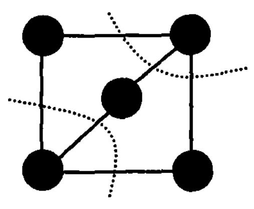


Figure 8.10a

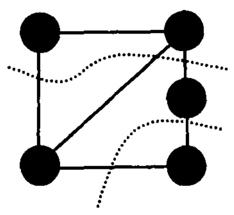


Figure 8.10b

Figure 8.10a and b Partitioning two cycles by two independent cut-lines

Therefore, two independent cut-lines can partition two cycles into three, four or five parts. The two independent cut-lines partition the two cycles into:

three parts if the two cut-lines did not cut any internal edge;

four parts if one cut-line cuts the internal edge and the other did not;

or five parts if the two cut-lines cut the internal edge independently.

Having illustrated the different routes one or more cut-lines have to follow to partition a cycle or a set of cycles, Sections 8.5 and 8.6 define the starting edge and the ending edge and the route of each cut line.

8.5 The class of the starting-edge and the ending-edge

A network can be partitioned into k parts by either k-1 dependent cut-lines or k/2 independent cut-lines or a combination of dependent and independent cut-lines and depending on the size of the network. Each cut-line partitions the network into two parts exactly. Let L represent a cut-line.

The starting-edge and the ending-edge of a cut-line with respect to a cycle have been introduced in Section 8.3.2.

In a network, the cut-line starts by cutting an edge of external type and then proceeds inside a cycle, in the network, looking to cut another edge. The next edge is either an internal edge or an external edge. If the cut-line cuts an internal edge, it will continue proceeding looking for another edge. If the next edge is of external type, the cut-line will terminate. **Thus, the starting-edge and the ending-edge of a cut-line, in a network, are always of external type.**

Furthermore, the **starting-edge** and **the ending-edge** of a cut-line may belong to the one cycle or they may belong to two different cycles. If they belong to one cycle, then the cycle is either an external cycle, so that there is no internal edge to be cut, or a mixed-cycle, and the cut-line does not cut any internal edge. If they belong to two different cycles, then the two cycles are of mixed-cycle type, and the cut-line has to cut one or more internal edges.

Therefore, each cut-line starts from an external edge of a mixed-cycle, i.e. it starts by cutting a mixed cycle, and it ends by cutting an external edge of a mixed-cycle, i.e. by cutting another mixed-cycle.

Each mixed-cycle has one or more internal edges with other cycles. These cycles are either internal cycles or mixed-cycles. If there is only one internal edge, then the cut-line will cut the internal edge, i.e. it will cut the new cycle. If there are more than one internal edge, and each internal edge belongs to a different cycle, then the cut-line selects a route to minimize the number of internal edges as will be explained in Section 8.5.4.

8.5.1 The sub-classes of the starting and the ending edges

Even though the starting-edge and ending-edges of the cut-line are of external type, the external type has many sub-classes as given in Section 7.5.4. For example, the X edges have nodes of type V_{X2}, and at least every two edges belong to one mixed-cycle, while each edge of sub-class XI_{ij} has two nodes of type V_{XdI}, therefore each XI_{ij} edge belongs to one and only one mixed-cycle. Furthermore, Section 7.6.1 gives the location of a mixed cycle in the external closed path. Therefore, a XI_{ij} edge of a mixed cycle of middle location is more suitable to be the start edge or the end edge than an X edge of a terminal mixed cycle.

8.5.2 The number of possible cut lines in p_{GX}

Since $P_{GX} = m_X = n_X$, as defined by equation (7.34), then the edges of p_{GX} can be numbered from 1 to m_X. Since a cut line starts and ends by cutting external edges, then if the ith external edge is selected to be the starting edge, then, then there will be $(m_X - 1)$ different cut lines to the remaining $(m_X - 1)$ external edges. If the cut lines are started from the $(i+1)^{th}$ external edge, there will be $(m_X - 2)$ different cut lines. If the cut lines started from $(i+2)^{th}$ external edge, there will be $(m_X - 3)$ different cut lines. The same case continues with other external edges until there is one cut line between the last external edge and the first external edge. Thus, the total number of cut lines is the sum of the number of cutting lines from external edge, i.e.

Total possible number of cut lines in p_{GX} of a network

$$= (m_X - 1) + (m_X - 2) + (m_X - 3) + \dots + 2 + 1. \qquad (8.1)$$

The required number of cutting lines is based on k the number of partitions.

8.5.3 The cut-line route

Each cut-line partitions the network into two parts by cutting a set of cut-edges, i.e. by cutting two external edges and zero, one or more internal edges. **The route of a cut-line** is defined to be the set of cut-edges.

If there is more than one cut-line, then each cut-line has its route. Cut-lines are not allowed to cross each other. If the two cut-lines are independent, then each cut-line has a different route, (i.e. different starting edge, ending edge and internal edges). If the two cut-lines are dependent, then the two cut-lines will share part of the route but not all the route. They may have the same starting edge or the same ending edge or they may share one or more internal edges, but each cut line has its route.

The starting-edge and the ending-edge, each, belong to cycles of mixed type. Thus, route of a cut-line starts from a mixed-cycle, and proceeds to either a mixed cycle or an internal cycle until it terminates by cutting an external edge of a mixed-cycle.

The open internal paths, P_{OI}, and the open mixed paths P_{OXI} introduced in Section 7.8 are used to determine the route of a cut-line from the starting-edge to the ending-edge, i.e. from one cycle to anther cycle. Since the end-nodes of the starting edge and the end-nodes of the ending edge are of external type and since the end-nodes of both the open internal path and the open mixed path are of external type, **then there are two open paths** between the end nodes of the starting-edge and the ending-edge. Those **two open paths** determine the **route** of one and only one **cut line**. The two paths both may be open internal paths or both open mixed paths or combination of an open internal path and an open mixed path. If one of the two paths is an open mixed path then the open a mixed path has minimum number of external edges. Those two open paths are termed $P_{b\text{-}1}$ and $P_{b\text{-}2}$, **the boundary-cut-lines**. Between $P_{b\text{-}1}$ and $P_{b\text{-}2}$, there are only two external edges, the starting-edge and the ending-edge, and there may be zero, one or more than one internal edges. The goal of the cut line is to cut the minimum number of internal edges between the starting edge and the ending edge.

The cut-line may cut zero, one, or more than one internal edge of P_{b-1} and P_{b-2} edges. The end nodes of the external edges, i.e. the starting edge and the ending edge, is termed **the external boundary nodes**, and the end nodes of the internal edges, that have been cut by the cut line, will be termed **the internal boundary nodes**, as explained more in section (8.6.1).

8.6 Partitioning the network nodes

Since the cut-line partitions the network into two parts, then V, the set of network nodes, is partitioned into three sets: the boundary set; the external set; and the internal set. Thus each part consists of these sets.

8.6.1 The boundary nodes

The **two open internal paths** may have internal edges between them, and they might not. The end nodes of the starting edge, the ending edge and the internal edges between P_{b-1} and P_{b-2} are the **boundary nodes**. Let

V_b be the set of **boundary nodes,**

n_b be the number of elements of V_b,

V_{Xb} be the set of boundary nodes from the external edges only,

n_{Xb} be the number of elements in V_{Xb},

V_{Ib} be the set of internal boundary nodes from only the internal edges, between P_{b-1} and P_{b-2}, and let n_{Ib} be the number of elements in V_{Ib}.

Then

$$V_b = \{V_{Xb} \oplus V_{Ib}\}; \qquad (8.2)$$

and

$$n_b = n_{Xb} + n_{Ib}. \qquad (8.3)$$

Since the cut-line cuts two edges, the starting-edge and the ending-edge, then the two edges are either independent or dependent. If they are independent then

$$n_{Xb} = 4; \qquad (8.4)$$

and if they are dependent, then

$$n_{Xb} = 3. \qquad (8.5)$$

From Theorem (6.1) the balance partitioning number of boundary nodes is:

$$n_{bu} = \left\lceil \frac{n}{k+1} \right\rceil$$

and

$$n_{bl} = n_{bu} - 1;$$

Thus, let

$$n_b = n_{bu};$$

or

$$n_b = n_{bl}.$$

Then the range of the number of internal boundary nodes is

$$0 \le n_{Ib} \le n_b - n_{Xb}. \qquad (8.6)$$

8.6.2 The remaining internal nodes

Since V_I is the set of internal nodes in the network, and V_{Ib} is the set of **internal boundary nodes** in the network, then the difference between V_I and V_{Ib} gives **the set of remaining internal nodes** in the network.

Let

V_{IP}	be **the set of remaining internal nodes,**
n_{IP}	be the number of elements of V_{IP},

then

$$V_I = \{V_{IP} \oplus V_{Ib}\}; \tag{8.7}$$

and

$$n_{IP} = n_I - n_{Ib}. \tag{8.8}$$

8.6.3 The internal and external nodes of the two parts

The nodes of each part (or subsystem) consist of external nodes and internal nodes. The following discussion defines the numbers of external nodes and boundary nodes in each part.

Since V_X is the set of external nodes in the network as described by equation (7.6), and since V_{Xb} is the set of external boundary nodes in the network as defined by equation (8.2), then let V_{XP} be the set of remaining external nodes in the network such that:

$$V_X = \{V_{Xb} \oplus V_{XP}\} \tag{8.9}$$

and let n_{XP} be the number of elements of V_{XP}, then

$$n_{XP} = n_X - n_{Xb} \tag{8.10}$$

Let

$V_{XP\text{-}i}$ be the set of external nodes of the i^{th} part

$n_{XP\text{-}i}$ be the number of elements in V_{XPi},

then

$$n_{XP\text{-}i} = \frac{n_{XP}}{2} = \frac{n_X}{2} - 2. \tag{8.11}$$

If $n_{XP\text{-}1} = n_{XP\text{-}2}$ and $n_{Xb} = 4$

then

$$n_{XP\text{-}1} = n_{XP\text{-}2} = \frac{n_X - 4}{2} = \frac{n_X}{2} - 2. \tag{8.12}$$

Let

$V_{IP\text{-}i}$ be the set of remaining internal nodes in the i^{th} part for $i = 1,2$,

$n_{IP\text{-}i}$ be the number of elements of $V_{IP\text{-}i}$,

then

$$V_{IP} = \{V_{IP\text{-}1} \oplus V_{IP\text{-}2}\}; \tag{8.13}$$

and

$$n_{IP} = n_{IP\text{-}1} + n_{IP\text{-}2}. \tag{8.14}$$

Thus, from equation (8.8) and (8.14)

$$n_{IP} = n_{IP\text{-}1} + n_{IP\text{-}2} = n_I - n_{Ib};$$

If $n_{IP\text{-}1} = n_{IP\text{-}2}$

then

$$n_{IP\text{-}i} = \frac{n_{IP}}{2} = \frac{n_I - n_{Ib}}{2} \tag{8.15}$$

Let

$V_{P\text{-}i}$ be the set of nodes of the i^{th} part

$n_{P\text{-}i}$ be the number of elements in $V_{P\text{-}i}$,

then

$$V_{P\text{-}i} = \{V_{XP\text{-}i} \oplus V_{IP\text{-}i}\}; \tag{8.16}$$

and

$$n_{P\text{-}i} = n_{XP\text{-}i} + n_{IP\text{-}i}. \tag{8.17}$$

8.6.4 Using the network open paths to obtain the $V_{IP\text{-}i}$

The set of nodes of $V_{IP\text{-}i}$ can be obtained from the set of internal paths between V_{XPi}, the i^{th} part external and $P_{b\text{-}i}$ the i^{th} boundary path.
Since the end nodes of any open internal path between V_{XPi} and $P_{b\text{-}i}$ are either external nodes or internal nodes or internal boundary nodes,

$$V_{IP\text{-}i} = \bigcup(P_{OI\text{-}Pi})$$
$$n_{IP\text{-}i} = |V_{IP\text{-}i}|. \tag{8.18}$$

8.7 The I - I and I - J cut lines

Since the **starting edge** and the **ending edge** of a cut line are of external type, then they belong to P_{GX} the network external path. Since

$p_{GX} = m_X = n_X$, as given by equation (7.34), then one cut-line partitions P_{GX} into two parts.

Let each part equals the network diameter as given by equation (7.45). Let the edges of each part be numbered from one to the network diameter.

If p_{GX} is even, then the length of each part equals the network diameter, and if p_{GX} is odd, then the length of one part equals the network diameter, and the length of the second part equals the network diameter minus one. Figure 8.11 shows a network, with even p_{GX}, partitioned by an I-I cut line.

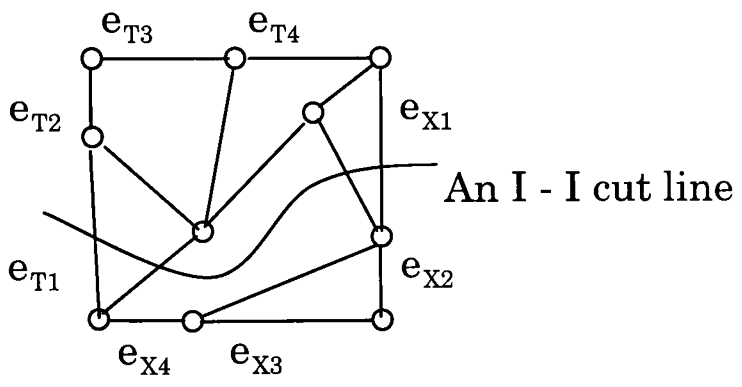


Figure 8.11 Using the I-I cut line
to partition a network with even p_{GX}

The starting-edge of a cut line can be any external edge of part one, i.e. $\{e_{X1}, e_{X2}, e_{X3}, e_{X4}\}$, and the ending-edge can be any external edge of part two, i.e. $\{e_{T1}, e_{T2}, e_{T3}, e_{T4}\}$. If both the starting-edge and the ending-edge have the same edge number, then the cut line is termed an **I-I cut line**, else it is termed an **I-J cut line.**

Let $L(e_{Xi}, e_{Tj})$ be a cut line from the ith external edge of part one to the jth external edge of part two.

Let $j = \text{Diameter} + i$; (8.19)

then if $i = j$

then $L(e_{Xi}, e_{Tj})$ is an **I-I cut line;**

else $L(e_{Xi}, e_{Tj})$ is an **I-J cut line.**

The end-nodes of the starting edge and the ending edge determine the route of the cut line. Thus if the network diameter is even there are $(\frac{m_X}{2})$ **I-I cut lines**, and if the network diameter is odd there are $(\frac{m_X}{2}-1)$ **I-I cut lines**.

8.7.1 The new balanced partitioning conditions

An **I-I cut line** partitions the network nodes into two three parts. One part represents the boundary part and two parts each of them represents one subsystem. Those three parts may be equal, balanced or unbalanced. The necessary balancing conditions of these three parts are discussed in this section.

The number of external nodes, in each the two parts, is given by equation (8.11), and the number of the internal nodes, in each of the two parts, is given be equation (8.15).

If p_{GX} is even, then an **I-I cut line** partitions P_{GX} into two equal external parts such that $n_{XP\text{-}1} = n_{XP\text{-}2} = \frac{n_X}{2} - 2$. Then the two parts have equal number of external nodes.

If the p_{GX} is odd, then the **I-I cut line** partitions P_{GX} into two balanced external parts such that $n_{XP\text{-}1} = \frac{n_X}{2} - 2$, and $n_{XP\text{-}2} = n_{XP\text{-}1} - 1$. Then the two parts have balanced number of external nodes.

If $n_{IP\text{-}1} = n_{IP\text{-}2} = \frac{n_I - n_{Ib}}{2}$ then the **I-I cut-line** partitions V_{IP} into two equal internal parts.

Since the nodes of each part is defined by equation (8.17), i.e. $n_{P\text{-}i} = n_{XP\text{-}i} + n_{IP\text{-}i}$, then the two parts are equal if they have equal numbers of external nodes and the internal nodes, i.e.

If $(n_{XP\text{-}1} + n_{IP\text{-}1}) = (n_{XP\text{-}2} + n_{IP\text{-}2})$ then the two parts are equal.

If $n_b = n_{P\text{-}1} = n_{P\text{-}2}$ then the **I-I cut line** partitions the network nodes into two equal parts, in this case the partition is called **equal partitions**.
If $|n_b - n_{P\text{-}i}| = 1$ then the partition is called **balanced partitions**.

8.8 Partitioning a network into k-balanced parts

Partitioning a network, with c>1 cycles and with $E_B = \{\Phi\}$, $V_S = \{\Phi\}$ and $V_{XdB} = \{\Phi\}$ into k subsystems, has the same principle as partitioning a cycle into k parts.

The first step: determining the number of **cut-lines** and weather these cut

lines are dependent or independent.

The second step: partitioning P_{GX}, the network external closed path

into k-balanced parts.

The third step: partitioning n_I, the total number of internal nodes

into (k+1) balanced parts.

The goal of the following theorem is to establish an approach to test the existence of balanced partitioning in a given network. The approach is based on using the cut-line principle introduced in Section 8.5.

Theorem 8.2 (The balanced partitioning theorem)

In a network with c>1 and $V_B = \{\Phi\}$, $V_{XdB} = \{\Phi\}$ and $V_{XdBI} = \{\Phi\}$, if the network diameter is even and if L is an I-I cut-line such that

$$n_{XP\text{-}1} \cong n_{XP\text{-}2}$$

$$n_{IP\text{-}1} \cong n_{IP\text{-}2}$$

then

$$n_{P\text{-}1} \cong n_{P\text{-}2}.$$

If $n_{P\text{-}i} \cong \left\lceil \frac{n}{k+1} \right\rceil$ for $i = 1,2$,

and if $n_{Ib} \leq \left\lceil \frac{n}{k+1} \right\rceil - n_{Xb}$

Then $n_b \cong n_{P\text{-}i} \cong \frac{n_X}{2} + n_{IP\text{-}i} - 2$ for $i = 1, \ldots, k$.

Proof

Since the network diameter is even, then from equations (8.11)

$$n_{XP\text{-}i} = \frac{n_X}{2} - 2 \text{ for } i = 1,2.$$

Since $n_{IP\text{-}1} \cong n_{IP\text{-}2}$

Then

$$n_{P\text{-}i} = n_{XP\text{-}i} + n_{IP\text{-}i} \quad \text{for } i = 1,2.$$

Thus $n_{P\text{-}1} \cong n_{P\text{-}2}$.

since $n_{P\text{-}i} \cong \left\lceil \frac{n}{k+1} \right\rceil$ for $i = 1,2$,

$$n_{Ib} \leq \left\lceil \frac{n}{k+1} \right\rceil - n_{Xb}$$

and since

$$n_b = n_{Xb} + n_{Ib};$$

then

$$n_b \leq n_{Xb} + \left\lceil \frac{n}{k+1} \right\rceil - n_{Xb}$$

thus $n_b \leq \left\lceil \frac{n}{k+1} \right\rceil$

$$n_b \cong n_{P\text{-}i} \text{ for } i = 1,2$$

•

8.9 Applying the new balanced partitioning conditions

The new balanced partitioning conditions are applied on the IEEE-14 network, shown in figure 8.12. The goal is to partition the network with using a partitioning technique.

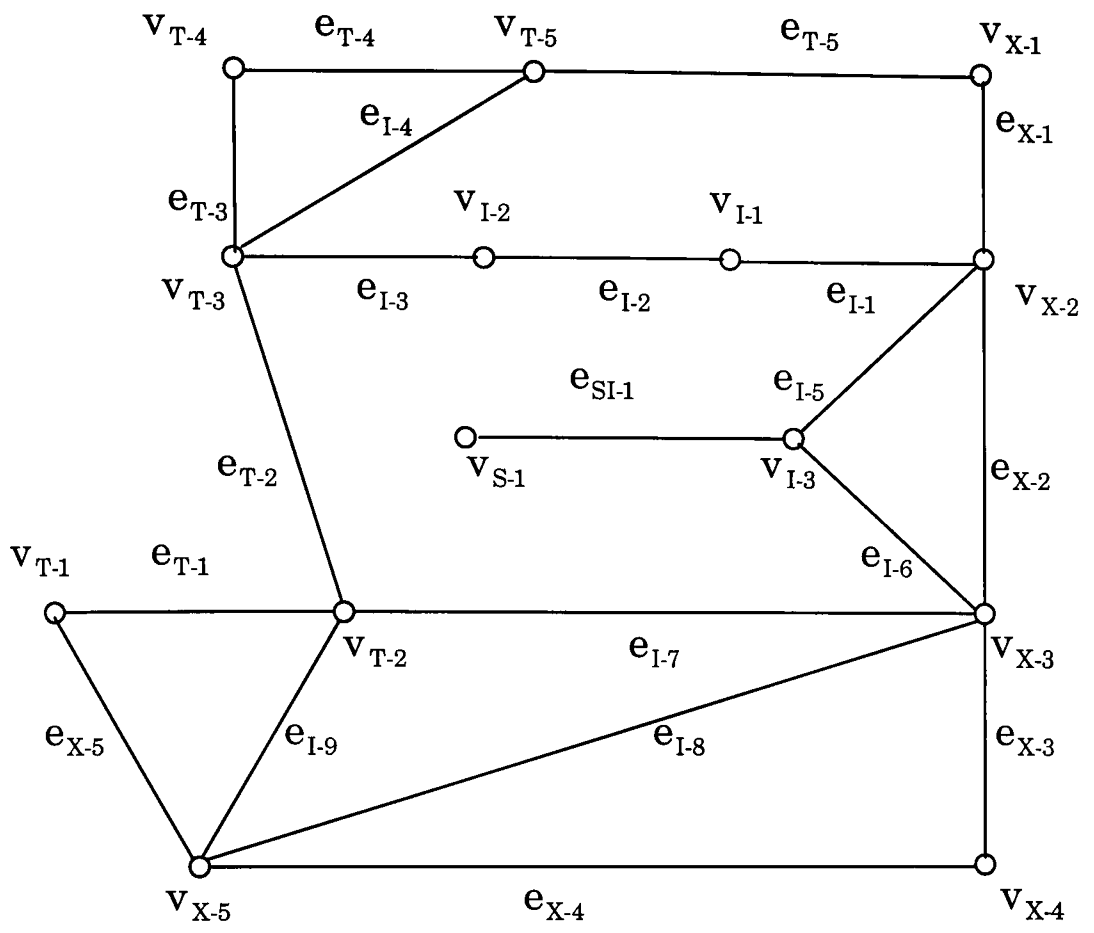


Figure 8.12 Identifying the parameters of the IEEE-14 network

Example (8.1) Applying the new balancing conditions

Step 1: Identifying the network parameters

$E_X = \{e_{X-1}, e_{X-2}, e_{X-3}, e_{X-4}, e_{X-5}, e_{T-1}, e_{T-2}, e_{T-3}, e_{T-4}, e_{T-5}\}$; and $m_X = |E_X| = 10$;

$V_X = \{v_{X-1}, v_{X-2}, v_{X-3}, v_{X-4}, v_{X-5}, v_{T-1}, v_{T-2}, v_{T-3}, v_{T-4}, v_{T-5}\}$; and $n_X = |V_X| = 10$;

$E_I = \{e_{I-1}, e_{I-2}, e_{I-3}, e_{I-4}, e_{I-5}, e_{I-6}, e_{I-7}, e_{I-8}, e_{I-9}\}$; and $m_I = |E_I| = 9$;

$V_I = \{v_{I-1}, v_{I-2}, v_{I-3}\}$; and $n_I = |V_I| = 3$;

$V_S = \{v_{S-1}\}$; and $n_S = |V_S| = 1$.

Thus m and n given by equations (7.5) and (7.9) are

$m = m_X + m_I + m_B + m_S = 10 + 9 + 0 + 1 = 20$;

$n = n_X + n_I + n_B + n_S = 10 + 3 + 0 + 1 = 14$;

and $p_{GX} = m_X = n_X = 10$

Step 2: Calculating the k-balanced partitioning

Let k=2, and by applying Theorem (8.1), the following results are obtained:

Since p_{GX} is even, then from equation (8.11) $n_{XP\text{-}i} = \frac{n_X}{2} - 2 = 3$; for $i = 1,2$

Since $p_{GX} > 4$, then from equation (8.4) $n_{Xb} = 4$;

Since $n_I = 3$, then from equation (8.15) $n_{IP\text{-}i} = \frac{n_I}{k+1} = \frac{3}{3} = 1$;

Thus, from equation (8.6) $1 \geq n_{Ib} \geq 0$

If $n_{Ib} = 1$,

then from equation (8.3)

$$n_b = n_{Xb} + n_{Ib} = 4 + 1 = 5;$$

and from equation (8.17)

$$n_{P\text{-}i} = n_{XP\text{-}i} + n_{IP\text{-}i} = 3 + 1 = 4 \text{ for } i = 1,\ldots,k;$$

and since $n_S = |V_S| = 1$, then the one-degree node is an internal node and it belongs to one of the two parts.

Thus the two balanced parts are: {4, 5} and the balanced boundary nodes is {5}.

If $n_{Ib} = 0$,

then

$$n_b = n_{Xb} + n_{Ib} = 4 + 0 = 4;$$

since $n_{Ib} = 0$, i.e. no internal boundary nodes, then the internal nodes belong to the two parts, i.e.

$$n_{IP} = n_I.$$

Balancing is achieved by dividing the internal nodes equally between the two pars, and equation (8.14) can be used as follows:

let $$n_{Ik} = \frac{n_I}{k} = \left\lceil \frac{n_I}{k} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2;$$

then

$$n_{IP\text{-}1} = n_{Ik} = 2,$$

$$n_{IP\text{-}2} = n_{IP} - n_{IP\text{-}1} = 3 - 2 = 1;$$

and from equation (8.17)

$$n_{P\text{-}1} = n_{XP\text{-}1} + n_{IP\text{-}1} = 3 + 2 = 5\,.$$

$$n_{P\text{-}2} = n_{XP\text{-}2} + n_{IP\text{-}2} = 3 + 1 = 4\,.$$

Since $n_S = |V_S| = 1$, then the one-degree node will belong to subsystem number two.

Thus the two balanced parts are: {5, 5} and the balanced boundary nodes is {4}.

Knowing that the network has these balanced values, section 7. describes a way of finding the I-I cut line, i.e. to check the given network has these values.

8.10 Determining the I-I cut-line in a network

Because the network has many different connections, now the role of the partitioning algorithm is to verify that the given configuration of the network has the balanced properties.

Since $n_{XId} = 6$, then the number of P_{OI} is given in equation (7.48), i.e.

$$n_{P_{OI}} = \sum_{i=1}^{n_{XId}-1} i\,;$$

thus $n_{P_{OI}} = 5 + 4 + 3 + 2 + 1 = 15$. Instead of generating the 15 open internal paths of the network, only P_{OI}'s of the I-I cut lines are generated.

Since the diameter = $\frac{m_X}{2} = 5$, then there are 5 possible I-I cut-lines.

These 5 I-I cut-lines are: $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$, $L_{2,2}(e_{X\text{-}2}, e_{T\text{-}2})$, $L_{3,3}(e_{X\text{-}3}, e_{T\text{-}3})$, $L_{4,4}(e_{X\text{-}4}, e_{T\text{-}4})$ and $L_{5,5}(e_{X\text{-}5}, e_{T\text{-}5})$.

Finding which of these I-I cut lines is based on Procedure (8.1)

Procedure (8.1)

```
        i=1;
        While (i < the number of I-I cut lines)
Step 1: Find from the network the data of the I-I cut line;
Step 2: Find the set of external boundary nodes of the I-I cut line;
Step 3: Find the set of internal connections between the two open paths;
Step 4: Find from the set of connections the internal boundary nodes;
Step 5: If the number of internal boundary nodes is in the acceptable range
                Then the network has the balanced partitioning values.
                Terminate.
        Else
                The I-I cut-line is not suitable;
        End
        i = i +1;
        End;
```

Applying Procedure (8.1) on the IEEE-14 network given in Figure (8.12) gives the following results:

Step 1: The data of $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ are:

$n_{OI\text{-}1}(v_{X\text{-}1}, v_{T\text{-}2}) = 3$;

$n_{OI\text{-}2}(v_{X\text{-}2}, v_{T\text{-}1}) = 4$;

$E_{P_{OI\text{-}1}}(v_{X\text{-}1}, v_{T\text{-}2}) = \{e_{T\text{-}5}, e_{I\text{-}4}, e_{T\text{-}2}\}$;

$E_{P_{OI\text{-}2}}(v_{X\text{-}2}, v_{X\text{-}5}) = \{e_{I\text{-}5}, e_{I\text{-}6}, e_{I\text{-}8}, e_{X\text{-}5}\}$;

$V_{P_{OI\text{-}1}}(v_{X\text{-}1}, v_{T\text{-}2}) = \{V_{X\text{-}1}, V_{T\text{-}5}, V_{T\text{-}3}, V_{T\text{-}2}\}$;

$V_{P_{OI\text{-}2}}(v_{X\text{-}2}, v_{T\text{-}2}) = \{V_{X\text{-}2}, V_{I\text{-}3}, V_{X\text{-}3}, V_{X\text{-}5}, V_{T\text{-}1}\}$;

Step 2: Finding the external boundary nodes:

$V_{Xb\text{-}1} = \{V_{X\text{-}1}, V_{T\text{-}2}\}$; and $n_{Xb\text{-}1} = 2$;

$V_{Xb\text{-}2} = \{V_{X\text{-}2}, V_{T\text{-}1}\}$; and $n_{Xb\text{-}2} = 2$;

Thus $n_{Xb} = n_{Xb\text{-}1} + n_{Xb\text{-}2} = 4$.

Step 3: Finding the set of connection edges between $P_{OI\text{-}1}$ and $P_{OI\text{-}2}$:

$\{e_{I\text{-}1}, e_{I\text{-}2}, e_{I\text{-}3}\}$;

$\{e_{I\text{-}7}\}$;

$\{e_{I\text{-}8}\}$.

Step 4: Finding the internal boundary nodes:

The $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ cut line starts by cutting $e_{X\text{-}1}$, then it proceeds to the first set of internal edges $\{e_{I\text{-}1}, e_{I\text{-}2}, e_{I\text{-}3}\}$ as follows:

if $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ cuts $e_{I\text{-}1} = (v_{X\text{-}2}, v_{I\text{-}1})$, then $v_{I\text{-}1}$ will be an internal boundary node, i.e. $n_{Ib} = 1$.

if $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ cuts $e_{I\text{-}2} = (v_{I\text{-}1}, v_{I\text{-}2})\, e_{I\text{-}2}$, then $v_{I\text{-}1}$ and $v_{I\text{-}2}$ will be internal boundary nodes, i.e. $n_{Ib} = 2$.

if $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ cuts $e_{I\text{-}3} = (v_{T\text{-}3}, v_{I\text{-}2})$, then $v_{I\text{-}2}$ will be an internal boundary node and $v_{T\text{-}3}$ a new unacceptable external boundary node.

Since $1 \geq n_{Ib} \geq 0$, then $e_{I\text{-}1}$ is the only suitable edge to be cut by $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$.

The $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ cut line, is then, proceed to the second internal set of edges, i.e. $\{e_{I\text{-}7}\}$.

If $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ cuts $e_{I\text{-}7}$, then a new external node becomes a boundary node. Thus $e_{I\text{-}7}$ is not suitable to be cut by $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$.

Step 5:

Since the route of $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ has to cut $e_{I\text{-}7}$ to reach to the ending edge, and since cutting $e_{I\text{-}7}$ adds an external node to the boundary set, then the $L_{1,1}(e_{X\text{-}1}, e_{T\text{-}1})$ cut line is not suitable I-I cut line.

Step 1: The data of $L_{2,2}(e_{X\text{-}2}, e_{T\text{-}2})$

$$n_{OI\text{-}1}(v_{X\text{-}2}, v_{T\text{-}3}) = 3;$$

$$P_{OI\text{-}2}(v_{X\text{-}3}, v_{T\text{-}2}) = 1;$$

$$E_{P_{OI\text{-}1}}(v_{X\text{-}2}, v_{T\text{-}3}) = \{e_{I\text{-}1}, e_{I\text{-}2}, e_{I\text{-}3}\};$$

$$E_{P_{OI\text{-}2}}(v_{X\text{-}3}, v_{T\text{-}2}) = \{e_{I\text{-}7}\};$$

$$V_{P_{OI\text{-}1}}(v_{X\text{-}2}, v_{T\text{-}3}) = \{v_{X\text{-}2}, v_{I\text{-}1}, v_{I\text{-}2}, v_{T\text{-}3}\};$$

$$V_{P_{OI\text{-}2}}(v_{X\text{-}3}, v_{T\text{-}2}) = \{v_{X\text{-}3}, v_{T\text{-}2}\}.$$

Step 2: Finding the external boundary nodes of $L_{2,2}(e_{X\text{-}2}, e_{T\text{-}2})$:

The $L_{2,2}(e_{X-2}, e_{T-2})$ cut line starts by cutting e_{X-2} and ends by cutting e_{T-2}. Thus the external boundary nodes are:

$$V_{Xb-1} = \{V_{X-2}, V_{T-3}\};\ n_{Xb-1} = 2;$$

$$V_{Xb-2} = \{V_{X-3}, V_{T-2}\};\ n_{Xb-2} = 2;$$

Thus $n_{Xb} = n_{Xb-1} + n_{Xb-2} = 4$.

Step 3: The set of connection edges between P_{OI-1} and P_{OI-2} are:

$$\{e_{I-5}, e_{I-6}\};$$

Step 4: The internal boundary nodes are:

The $L_{2,2}(e_{X-2}, e_{T-2})$ cut line starts by cutting e_{X-2}, then it proceeds to the set of internal edges between P_{OI-1} and P_{OI-2}, i.e. $\{e_{I-5}, e_{I-6}\}$ as follows:

Step 5:

if $L_{2,2}(e_{X-2}, e_{T-2})$ cuts e_{I-6}, then v_{I-3} will be an internal boundary node and v_{SI} will be a node belongs to part 1. Thus

$$n_{Ib} = 1;$$

therefore $n_b = n_{Xb} + n_{Ib} = 4 + 1 = 5;$

and $n_{P-1} = n_{XP-1} + n_{IP-1} + n_{SI} = 3 + 2 + 1 = 6;$

$n_{P-2} = n_{XP-2} + n_{IP-2} + n_{SI} = 3 + 0 + 0 = 3;$

Thus, if e_{I-6} is cut it will not give balanced partitions.

If $L_{2,2}(e_{X-2}, e_{T-2})$ cuts e_{I-5}, then v_{I-3} will be an internal boundary node and v_{SI} will be a node belongs to part 2.

Thus,

$$n_{Ib} = 1;$$

therefore $n_b = n_{Xb} + n_{Ib} = 4 + 1 = 5;$

and $n_{P-1} = n_{XP-1} + n_{IP-1} + n_{SI} = 3 + 2 + 0 = 5;$

$n_{P-2} = n_{XP-2} + n_{IP-2} + n_{SI} = 3 + 0 + 1 = 4;$

Thus, if e_{I-5} is cut balanced partitioning is achieved as shown in Figure 8.

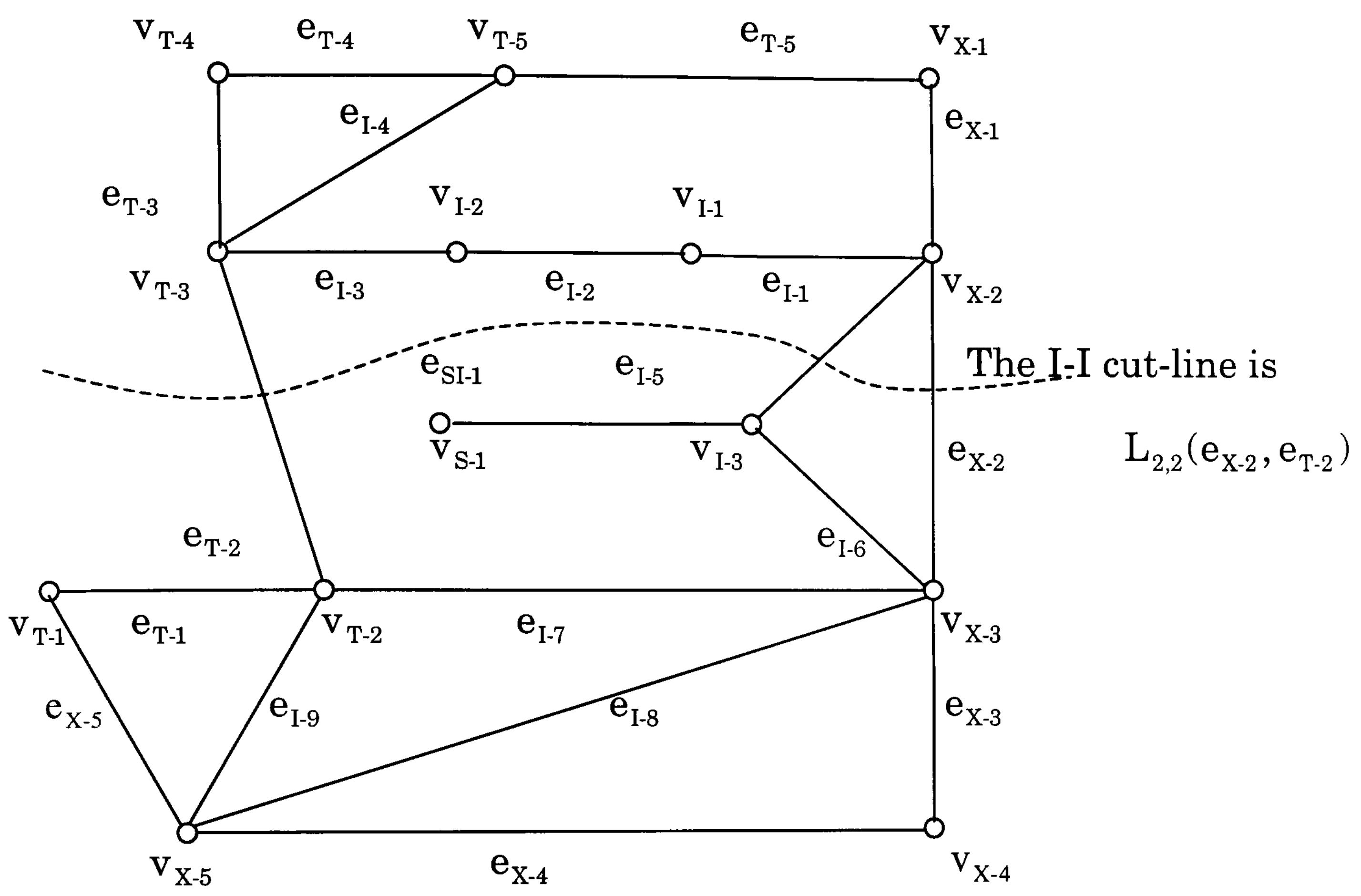


Figure 8.13 The balancing result of using the I-I cut-line on the IEEE-14 network

Step 1: The data of $L_{3,3}(e_{X-3}, e_{T-3})$

$n_{OI-1}(v_{X-3}, v_{T-4}) = 7$;

$P_{OI-2}(v_{X-4}, v_{T-3}) = 3$;

$E_{P_{OI-1}}(v_{X-3}, v_{T-4}) = \{e_{I-6}, e_{I-5}, e_{I-1}, e_{I-2}, e_{I-3}, e_{I-4}, e_{T-4}\}$

$E_{P_{OI-2}}(v_{X-4}, v_{T-3}) = \{e_{X-4}, e_{I-9}, e_{T-2}\}$;

$V_{P_{OI-1}}(v_{X-3}, v_{T-4}) = \{v_{X-3}, v_{I-3}, v_{X-2}, v_{I-1}, v_{I-2}, v_{T-3}, v_{T-5} v_{T-4}\}$;

$V_{P_{OI-2}}(v_{X-4}, v_{T-3}) = \{v_{X-4}, v_{X-5}, v_{T-2}, v_{T-3}\}$;

Step 2: Finding the external boundary nodes:

$V_{Xb-1} = \{v_{X-3}, v_{T-4}\}$; $n_{Xb-1} = 2$;

$V_{Xb-2} = \{V_{X-4}, V_{T-3}\}$; $n_{Xb-2} = 2$;

Thus $n_{Xb} = n_{Xb-1} + n_{Xb-2} = 4$.

Step 3: The set of connection edges between P_{OI-1} and P_{OI-2} are:

$\{e_{I-8}\}$;

$\{e_{I-7}\}$;

$\{e_{I-4}\}$;

$\{e_{I-1}, e_{I-2}, e_{I-3}\}$.

Step 4: The internal boundary nodes are:

The $L_{3,3}(e_{X-3}, e_{T-3})$ cut line starts by cutting e_{X-3}, then it proceeds to the set of internal edges between P_{OI-1} and P_{OI-2} as follows:

If the $L_{2,2}(e_{X-2}, e_{T-2})$ cut line cuts $e_{I-8} = (v_{X-5}, v_{X-3})$, then v_{X-5} will be a new external boundary node. Thus the $L_{2,2}(e_{X-2}, e_{T-2})$ cut line terminates.

Step 5: The $L_{2,2}(e_{X-2}, e_{T-2})$ cut line does not give balance partitions.

Step 1 : the data of $L_{4,4}(e_{X-4}, e_{T-4})$

$P_{OI-1}(v_{X-4}, v_{T-5}) = 5$;

$P_{OI-2}(v_{X-5}, v_{T-4}) = 3$;

$E_{P_{OI-1}}(v_{X-4}, v_{T-5}) = \{e_{X-3}, e_{I-6}, e_{I-5}, e_{X-1}, e_{T-2}\}$;

$E_{P_{OI-2}}(v_{X-5}, v_{T-4}) = \{e_{I-8}, e_{T-2}, e_{T-3}\}$;

$V_{P_{OI-1}}(v_{X-4}, v_{T-5}) = \{v_{X-4}, v_{X-3}, v_{I-3}, v_{X-2}, v_{X-1}, v_{T-5}\}$;

$V_{P_{OI-2}}(v_{X-5}, v_{T-4}) = \{v_{X-5}, v_{T-2}, v_{T-3}, v_{T-4}\}$;

Step 2: Finding the external boundary nodes:

$V_{Xb-1} = \{v_{X-4}, v_{T-5}\}$; $n_{Xb-1} = 2$;

$V_{Xb-2} = \{V_{X-5}, V_{T-4}\}$; $n_{Xb-2} = 2$;

Thus $n_{Xb} = n_{Xb-1} + n_{Xb-2} = 4$;

Step 3: The set of connection edges between P_{OI-1} and P_{OI-2} are:

$\{e_{I-8}\}$;

$\{e_{I\text{-}7}\}$;

$\{e_{I\text{-}1}, e_{I\text{-}2}, e_{I\text{-}3}\}$;

$\{e_{I\text{-}4}\}$.

Step 4: The internal boundary nodes are:

The $L_{4,4}(e_{X\text{-}4}, e_{T\text{-}4})$ cut line starts by cutting $e_{X\text{-}4}$, then it proceeds to the set of internal edges between $P_{OI\text{-}1}$ and $P_{OI\text{-}2}$ as follows:

If the $L_{4,4}(e_{X\text{-}4}, e_{T\text{-}4})$ cut line cuts $e_{I\text{-}8} = (v_{X\text{-}5}, v_{X\text{-}3})$, then $v_{X\text{-}3}$ will be a new external boundary node. Thus the $L_{4,4}(e_{X\text{-}4}, e_{T\text{-}4})$ cut line terminates.

Step 5: The $L_{4,4}(e_{X\text{-}4}, e_{T\text{-}4})$ cut line does not give balance partitions.

Step 1: The data of $L_{5,5}(e_{X\text{-}5}, e_{T\text{-}5})$

$n_{OI\text{-}1}(v_{X\text{-}5}, v_{X\text{-}1}) = 4$;

$P_{OI\text{-}2}(v_{T\text{-}1}, v_{T\text{-}5}) = 3$;

$E_{P_{OI\text{-}1}}(v_{X\text{-}5}, v_{X\text{-}1}) = \{e_{I\text{-}8}, e_{I\text{-}6}, e_{I\text{-}5}, e_{X\text{-}1}\}$;

$E_{P_{OI\text{-}2}}(v_{T\text{-}1}, v_{T\text{-}5}) = \{e_{T\text{-}1}, e_{T\text{-}2}, e_{I\text{-}4}\}$;

$V_{P_{OI\text{-}1}}(v_{X\text{-}5}, v_{X\text{-}1}) = \{v_{X\text{-}5}, v_{X\text{-}3}, v_{I\text{-}3}, v_{X\text{-}2}, v_{X\text{-}1}\}$;

$V_{P_{OI21}}(v_{T\text{-}1}, v_{T\text{-}5}) = \{v_{T\text{-}1}, v_{T\text{-}2}, v_{T\text{-}3}, v_{T\text{-}5}\}$.

Step 2: Finding the external boundary nodes:

$V_{Xb\text{-}1} = \{v_{X\text{-}1}, v_{X\text{-}5}\}$; $n_{Xb\text{-}1} = 2$;

$V_{Xb\text{-}2} = \{V_{T\text{-}5}, V_{T\text{-}1}\}$; $n_{Xb\text{-}2} = 2$;

Thus $n_{Xb} = n_{Xb\text{-}1} + n_{Xb\text{-}2} = 4$.

Step 3: The set of connection edges between $P_{OI\text{-}1}$ and $P_{OI\text{-}2}$ are:

$\{e_{I\text{-}9}\}$;

$\{e_{I\text{-}7}\}$;

$\{e_{I\text{-}1}, e_{I\text{-}2}, e_{I\text{-}3}\}$.

Step 4: The internal boundary nodes are:

The $L_{5,5}(e_{X-5}, e_{T-5})$ cut line starts by cutting e_{X-5}, then it proceeds to the set of internal edges between P_{OI-1} and P_{OI-2} as follows:

If the $L_{5,5}(e_{X-5}, e_{T-5})$ cut line cuts $e_{I-9} = (v_{X-5}, v_{T-2})$, then v_{T-2} will be a new external boundary node. Thus the $L_{5,5}(e_{X-5}, e_{T-5})$ cut line terminates.

Step 5: The $L_{5,5}(e_{X-5}, e_{T-5})$ cut line does not give balance partitions.

Thus applying such a procedure to test the existence of the balanced partitioning values in a given network is possible and achievable.

8.11 Chapter review

The cut line concept is very simple but important. It defines the boundary and the subsystems areas. Cut lines have relationships between them, either dependent or independent. Partitioning a network by using the network external path property and the I-I cut-line is possible and the balance partitioning vales are calculable, i.e. there is no need to use partitioning techniques to obtain these values. Testing the existence of the balanced partitioning values in a given network is possible and achievable by a procedure such as procedure (8.1).

Chapter 9

Concluding Remarks

9.1 General

In this thesis, the network-partitioning problem has been discussed. The basic requirement has been to produce a balanced partitioning of a given electrical network, subject to the constraints imposed by engineering application. That application has been chosen to be a decomposed solution to the state-estimation problem, with particular reference to an algorithm known as DSE. This algorithm gives a particularly attractive computational balanced partitioning of a global network.

The objective of research has been to devise new partitioning algorithms which satisfy the requirements of DSE, but which follow new directions in the use which is made of information provided by the global network, and in particular, by its graph-theoretical properties.

It is well known that the network-partitioning problem is classified as NP-hard. Previously proposed solutions to this problem have usually been based on largely heuristic principles, and the use of graph-theoretical properties in the partitioning process is often limited. An important aim of this research is to explore the use of graph-theoretic properties, some of which are newly defined for the purpose, in order to simplify computational solution to the partitioning problem.

The broad objectives of partitioning a network are to divide the network into k subsystems. The partitioning operation is based on 'cuts'. Cuts produce cut-edges, which determine subsystems and boundary nodes for each subsystem. The remaining nodes in each subsystem are denoted internal nodes. The effect of constraints arising from DSE is that partitioning should provide a balanced result, in which the number of internal nodes in each subsystem are approximately equal and not less than the number of global boundary nodes, as described in Chapter 5, i.e.

$$n_b \le \left\lfloor \frac{n}{k+1} \right\rfloor;$$

$$n_r = n - n_b;$$

$$n_p = \left\lfloor \frac{n_r}{k} \right\rfloor;$$

then

$$n_{ir} = n_p \text{ for i=1,2, ..., k;}$$

$$n_{ir} = n_p;$$

The value of k is not pre-determined by DSE. Instead, k is determined as the 'best' value, in the sense of a balanced result, by the partitioning process, and used as a defining parameter in specifying the processor arrangement for DSE.

9.2 Developments of the research

Following the definition of the DSE requirements, the goal was to develop partitioning techniques to satisfy the DSE restrictions. Several concepts from graph theory were investigated, such as the spanning tree method introduced in Chapter 4, the covering set concept and the contraction concept. Partitioning algorithms may be developed based on these and other graphical concepts.

Chronologically, the spanning tree approach was developed first, followed by the maximum degree technique and then by using the newly defined properties.

Due to the limited information on which it is based the spanning tree optimal partitioning method suffers from the disadvantage that, whilst it may give satisfactory results for some spanning trees of a given network, results may be unacceptable for other spanning trees, and its effectiveness varies from one to another.

The limitations of the first algorithm, the non-availability of helpful relations from graph theory and the desire to find a general solution were the motivation behind making better use of the DSE restrictions and examining graph theory for suitable properties which can help to obtain the boundary nodes and the internal nodes as early as possible in the partitioning process. This leads to the ideal balanced partitioning conditions, introduced in Chapter 5, and to use the covering set property introduced in Chapter 6. An investigation to find the relationship between the boundary nodes and boundary cycles leads to the edge state phenomenon discussed in Chapter 7.

Knowing that the network-partitioning problem is an NP-hard problem and knowing the un-completeness of the existing graph theory have increased the motivation to simplify the network-partitioning problem. The simplifications are discussed in the sequence of the thesis chapters. The following sections summarize the conclusions and the simplifications:

9.3 Partitioning by using the spanning tree property

One of the network properties is the spanning tree and its branching property. The network has many different spanning trees. The spanning tree and its branching property have been used to partition the network as described in Chapter 4. Obtaining a spanning tree by using the row reduction method is simple and fast. The obtained spanning tree matrix has a staircase structure. Balancing the number of edges of the k sub-spanning trees balances the k subsystems. The optimal number of cut-edges is (k-1) edges.

The optimal partitioning technique is designed to use the spanning tree branches property. Every spanning tree has branches. Using the branches property made the technique very fast and flexible. The technique is fast in finding the number of edges of each sub-spanning tree and flexible in selecting the starting branch. The flexibility of the technique facilitates the technique to partition a wide range of spanning trees. The staircase structure of spanning tree matrix limited the flexibility of the technique. Balancing the k sub-spanning tree is obtained by balancing the number of edges. The set of cut-edges and the set of boundary nodes are obtained by using simple fast matrix addition operation.

Classifying the spanning tree nodes into bottom nodes (or one-degree nodes), branch nodes and junction nodes simplified using the branches, consequently partitioning the spanning tree.

Using a spanning tree matrix, other than the staircase structure, with some modification to the algorithm will make the technique more general and able to partition more spanning trees.

9.4 Determining the ideal balanced partitioning values

Using the network properties is the first step of simplifying the NP-hard problem. Determining the conditions of an ideal balanced partitioning is the second step of simplifications the NP-hard problem.

In chapter (5) the DSE restrictions and the conditions for ideal balanced partitioning are discussed and formalized in simple equations. The classification of the subsystem nodes into internal nodes and boundary nodes is a very important fact in balanced partitioning. The number of boundary nodes has a direct relationship with the number of cut-edges in the network and with k, the number of subsystems. The number of internal nodes in a subsystem also has an indirect relationship with k.

Ignoring the existence and the importance of such classification, in balanced partitioning, was one of the difficulties of developing a mathematical formula for the balanced partitioning and it was one of the reasons that lead to classify the network-partitioning problem as NP-complete problem.

The relationship between internal nodes, the boundary nodes and k has lead to the ideal balanced partitioning. The concept of balancing the numbers of internal nodes in the k subsystems is fundamental to achieve to a global balanced partitioning. The conditions of the global ideal balanced partitioning are discussed in Chapter 5. Theorem (5.1) defined new simple mathematical formulas to obtain the ideal balanced partitioning values of a network for a given k, i.e. n_b, n_{ir} for $i = 1, 2, \ldots, k$ and the possible range of m_b.

The new formulas lead to new directions of dealing with the network-partitioning problem. The balanced partitioning values of a given network can be calculated. The introduction of these values by Theorem (5.1) is a simplification step to calculate these values early in the partitioning process, comparing with the traditional partitioning methods that find these values late.

The second new direction is with respect to the role of the partitioning technique. The role now is to check if the given network configuration has these values or not. If the network has these values, then how to obtain these sets, i.e. the set of cut-edges, the set of boundary nodes and the k-sets of internal nodes of the k subsystems.

9.5 Partitioning by using the ideal balanced partitioning values

Knowing the balanced partitioning values of a given network prior of using a partitioning technique leads to (i) check that the given network configuration possess the balanced partitioning values and to (ii) find these sets, i.e. the set of cut-edges, the set of boundary nodes and the set of internal nodes of each subsystem.

It has been found that the covering set concept is one of the nearest graph concepts which can be used to obtain the global boundary nodes as early as possible. Thus the maximum degree technique, introduced in Chapter 6, is based on finding two minimum covering sets, one set from the network and the other set is from the spanning tree.

Finding the set of global boundary nodes by using the maximum degree technique is fast, easy. The technique can be applied to a wide range of spanning trees to obtain the global boundary nodes from. The spanning tree of the network is used to determine the set of cut-edges and the subsystems nodes, i.e. the global boundary nodes and the internal nodes. The partitioning technique is fast and the simulation results agree with ideal balanced partitioning values obtained by using theorem (5.1).

It can partitions a wide range of spanning trees and give the ideal balanced partitioning values. The maximum degree uses the sets operation more than the matrix operation.

9.6 Towards theoretical foundation for graph theory

Further investigation in graph theory to obtain relationships between the set of boundary nodes and the boundary cycles has shown that many graph properties can be derived if suitable definitions and classifications are applied to the state of the edge in the network. It has been noted that an edge in a network is either belongs to a cycle or it does not. This observation has been named the 'edge state phenomenon' as described in Chapter 7.

The edge state phenomenon is a natural phenomenon that does exist in every network. It has the ability to explore the graph natural properties. Some of these properties, which are useful to the partitioning problem, have been deduced in Chapter 7 such as the network external closed path, the open internal paths and the cycle circumferences relationships. Other properties are to be stated in the future work and many other properties can be deduced.

It has been shown that graph is a structured entity. The basic elements of this entity are the nodes, the edges the connection.

A network is defined as a set of nodes connected by a set of edges. The mechanism of connection is very important. It defines the state of the edges, the state of the nodes and the network type. The states of four unconnected nodes are isolated nodes. The connection of three edges between the four nodes, such that no cycle is formed, defines the edges states and changes the nodes state. Two edges will be of one-degree type and one edge will be of a bridge type. The states of two nodes are changed to the one-degree types and two nodes are changed to the bridge types. The connection of a new edge between the two one-degree nodes, introduces a cycle. The cycle changes the states of the four edges and the four nodes to the external type.

The beauty of the edge state phenomenon is that it covers every thing in the network, the edges, the nodes, the paths and the cycles. It also covers the network itself in all its different sizes and all possible connections and with or without cycles. The edge state phenomenon makes it easy to deduce new graph properties and relationships in the two dimensions plane. There is a perfect match between the deduced relations, which define the different graph properties. Such as the network external close path and its relations with m, g_C and g_{max}, the basic configuration and the network classifications, $G_{P_{GX}}$.
The edge state phenomenon is also capable to describe the network properties in three dimensions.

Another graph property can be deduced related to the conservation of the graph natural properties. A change in the connection of one edge in G_u does not destroy the original graph properties. While, an elimination or an addition of an edge or a node to the network destroys the original graph properties. Thus new graph operations, based on the edge state phenomenon, can be defined and some of the conventional graph operations, defined in graph theory textbooks, need to be redefined.

The graph uniqueness is another property that has been deduced by using the edge state phenomenon and its properties.

9.7 Towards a mathematical partitioning model

Knowing the ideal balance partitioning values, the target is to find a mathematical model for partitioning the network. The edge state phenomena has been discovered while trying to find the mathematical solution. The flexibility and exhaustibility of the edge state phenomena guarantee that a mathematical partitioning model is possible.

The future mathematical partitioning model is based on the ideal balanced partitioning values obtained by using theory (6.1) and on the network properties as defined by the edge state phenomena such as the network external closed path, the circumference of the c cycles.

The first part of the mathematical model has been described in Chapter 8. Partitioning, as a cut operation in the network, has to follow a specified route to give the ideal balanced partitioning values. The route of the cut-line starts from an edge and ends at another edge. The following points can be concluded from Chapter 8:

- The cut-lines have relation between them, i.e. dependent or independent cut-lines
- Each cut-line has to cut at least two edges of external type.
- The cut-line may or may not cut internal edges.
- The end-nodes of the cut-edges are the boundary nodes. Thus the boundary nodes are classified into external and internal boundary nodes.
- The external edges belong to the external closed path. Thus network external closed path property has to be partitioned.
- Theorem (8.1) states the conditions of balanced partitioning of a specified network.

9.8 Future work

The network-partitioning problem is a very interesting and challenging problem and the discovery of edge state phenomenon makes it more interesting and really hard to stop. The deduced network properties are few graph properties in the plane, still there are many important graph properties and relationships that can be derived by using the edge phenomena concept, such as: The three dimensional graph.

The subject of the thesis is very interesting to the author. Thus, the future work will be a continuation to complete the mathematical partitioning model and to prove that the NP-hard problem is a P problem. Some of the points that has to be completed are:

- Design a procedure to identify the types edges and the nodes of a given network as described by the edge state phenomena in Chapter 8.
- Defining the cross edges property and its effect on the network.
- The distribution of the boundary nodes.
- Defining the network balance state.
- Applying the phenomenon in the three dimensions. The concept of finding the balanced partitioning values can be developed to find the unbalanced partitioning values.
- Defining new graph operations based on the edge state phenomenon.

References

1. Adby, P.R. and Demster, M. A.: "Introduction To Optimization Methods"; Chapman and Hall, London, 1974.
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.; " Network Flows: Theory, Algorithms, and Applications"; Prentice Hall, Englewood Cliffs, New Jersey, 1993.
3. Azzam, M. H.: " Developments In Decomposition Methods For Power System State Estimation ", Ph.D. Thesis, Brunel University, 1985.
4. Barnes. E.R.: " An algorithm for partitioning the nodes of a graph", SIAM J. Algebr. Discrete methods 1982, 3, (4), pp.291-307.
5. Beineke, Lowell W. and Wilson, Robin J.: "Selected Topics in Graph Theory"; Academic Press, New York/London, 1983.
6. Bellman, Richard: "Introduction To Matrix Analysis", New York; Maidenhead: McGraw-Hill, 1970.
7. Berge, C.; " Graphes"; Gauthier-Villars, 1985.
8. Berge,C.; " Hypergraphes"; Gauthier-Villars, 1987.
9. Biggs, N.L., Lloyd, E.K.; and Wilson, R.J.; " Graph Theory 1736-1936"; Clarendon Press, Oxford, 1976.
10. Bollobás, B.; " Random Graphs"; Academic Press, 1985.
11. Bollobás, B. "Advances In Graph Theory"; Amsterdam; Oxford: North-Holland Publishing, 1978.
12. Bondy, J.A., Murty, U.S.R.; "Graph Theory with Applications"; North-Holland, 1981.
13. Bui, T. N., and Jones, C.; "Finding good approximate vertex and edge partitions is NP-hard,"; Inf. Process. Letters, 42, 1992, pp. 153-159.
14. Bui, T. N., and Jones, C.; "A heuristic for reducing fill-in in sparse matrix factorization"; in Proceedings of Sixth SIAM Conference on Parallel Processing for Scientific Computing, 1993, pp. 445-452.

15. Botafogo, R. A.; "Cluster analysis for hypertext systems", In proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information retrieval (Pittsburg, Pa., June 27-July 1). 1993, ACM, New York, pp. 116-125.

16. Chamberlain, B.L.; "Graph Partitioning Algorithms for Distributing Workloads of Parallel Computation"; www.washington.edu/homes/brad/cv/publ/degree/generals.pdf, 1998.

17. Clements, K. A. and Wollenberg, B.F.: " An Algorithm for observability Determination in Power System State Estimation ", IEEE PES Summer Meeting, San Francisco, CA, A 75 447-3, July 1975.

18. Christofides, Nicos: " Graph Theory: An Algorithmic Approach ", Academic Press, London, 1975.

19. Colbourn, C. J.: "The Combinatorics of Network Reliability", vol. 4 of The International Series of Monographs on Computer Science. Oxford University Press, 1987.

20. Cullum, J. and Willoughby, R.A.: "Large Scale Eigenvalue Problems"; North Holland, Amsterdam, The Netherlands, pp. 193-240, 1985

21. Diestel, Reinhard: "Graph Theory"; New York: Springer, 1997.

22. Fiduccia,C.M., and Mattheyses, R.M.: " A linear-time heuristic for improving network Partitioning", Proceedings of 19th Design automation workshop, 1982, pp.175-181.

23. Fiorini, S. and Wilson, R. J.: "Edge Colourings of Graphs", London, Pitman, 1978.

24. Fleischner, H.; "Eulerian Graphs and Related Topics (Annals of Discrete Mathematics 45), North-Holland, Amsterdam, 1990.

25. Garey, M. R. and Johnson, D. S.: "Computers and Intractability: " A Guide to NP-Completeness ", Freeman, San Francisco, page 209-210, 1979.

26. Gibbons, A.; "Algorithmic Graph Theory"; Cambridge University Press, 1988.

27. Golumbic, M.C.; "Algorithmic Graph Theory and Perfect Graphs"; Academic Press, 1980.

28. Gondran, M., and Minoux, M.;" Graphes et Algorithmes"; Eyrolles, Paris, 1985.

29. Graham, R. M., Grötschel, L. and Lovász, " Handbook of Combinatorics"; North-Holland, Amsterdam, 1995, 2 volumes.

30. Graybill, Franklin A.: "Matrices With Applications in Statistics"; 1983.

31. Hampel, Frank R. and Ronchetti, Elvezio M.: "Robust Statistics", New York, Chichester, Wiley, 1986.

32. Habiballah, I.O. Roy, R.G. and Irving M.R.: "Markov Chains For Multipartitioning Large Power System State Estimation Networks", Int. Journal og Electric Power Research, Vol. 45, 1998, pp. 135-140.

33. Habiballah, I.O. and Quintana, V.H.: " Integrated-Linear-programming eigenvector-based approach for multi-partitioning power system state-estimation networks ", IEE Proc.-Gener. Transm. Distrib. 141, pp. 11-18, 1994.

34. Harary, F.; "Graph Theory"; Addison-Wesley, Reading, Massachusetts, 1969.

35. Hogben, Leslie: " Elementary Linear Algebra"; St. Paul, MN, 1975.

36. Jensen, T.R. and Toft, B.; "Graph Coloring Problems"; Wiley, New York, 1995.

37. Karypis, G. and Kumar, V.;" Analysis of multilevel graph partitioning"; Tech. Report 95-037, Computer Science Department, University of Minnesota, 1995.

38. Karypis, G. and Kumar, V.;" Parallel multilevel k-way partitioning scheme for irregular graphs"; In Super-computing 96 Conference Proceedings. ACM/IEEE, Nov. 1996. (a more complete version is available at http://www-users.cs.umn.edu/karypis/metis/publications/main.html).

39. Karger, David R. and Stein, Clifford; " A new Approach to the Minimum Cut Problem"; Journal of the ACM, Vol. 43, No. 4, July 1996, pp.601-640.

40. Karger, D. R. "A randomized fully polynomial time approximation scheme for the all terminal network reliability problem", In Proceedings of the 27th annual ACM, New York, pp. 11-17, 1995.

41. Kernighan, B. W. and Lin, L.: " An effecient heuristic procedure for partitioning graphs", Bell Syst. Tech. J., 1970, 49, pp. 291-307.

42. Krishnamurthy, B.; "An Improved Min-Cut Algorithm for Partitioning VLSI Networks", IEEE Trans. on Computers, Vol. C-33, May 1984, pp.438-446.

43. Lawler, E. L., Levitt K. N. and Turner, J.; "Module Clustering to Minimize delay in Digital networks"; IEEE Trans. on Computers, vol. C-18, Jan. 1969, pp. 47-57.

44. Lo, K. L.; Salem, M. M.; McColl, R. D. and Moffatt, A. M.; "Two-level state estimation for large power system Part1: Algorithms; Part2: Computational experience"; IEE Proceedings, Vol.135, Pt. C, No. 4, July 1988, p299-318.

44a. Lo, K. L.; Salem, M. M.; McColl, R. D.; Moffatt, A. M.and Sulley, J.L.; "Multi level state estimation for electric power systems"; 19th Universities Power Engineering Conference (UPEC 84), April 1984, paper 14.2, Dundee, UK .

44b. Lo, K. L.; Salem, M. M.; McColl, R. D.; and Moffatt, A. M. : "Two level power system state estimation"; 20th Universities Power Engineering Conference (UPEC 85), April 1985, pp. 37-40, Huddersfield, UK .

45. Lovász, L.; "Combinatorial Problems and Exercises"; 2ème édition, Akadémiai Kiadó, Budapest, 1993.

46. Lovász, L., Plummer, M.D.; "Matching Theory"; Annals of Discrete Mathematics 29, North-Holland, 1986 - also at: Akadémia Kiadó, Budapest, 1986.

47. Marsh, J.F. and Azzam, M "A model co-ordination approach to hierarchical static state-estimation for power systems"; Proc. of IEE conf., Durham, July 1986; pp. 145-149.

48. Marsh, J.F. and Azzam, M.: "MCHSE: a verstile frame-work for the design of two-level power system estimators"; IEE Proc. Pt. C, vol. 135, No.4, July 1988, pp. 291-298.

49. Marsh, J.F, Zitouni, S. and Irving, M.R.; "Computational Aspects of Distributed State-estimators for Power Systems"; 12th world Congress; IFAC; vol. 8, pp. 17-20, Sydney, Australia; 1993.

50. Nishizeki, T., Chiba, N.; "Planar Graphs: Theory and Algorithms (Annals of Discrete Mathematics 32)"; North-Holland, Amsterdam, 1988.

51. Ortega, James M.: " Matrix Theory ", Plenum, London, New York, 1988.

52. Palmer, Edgar: "Graphical Evaluation – An Introduction To Random Graphs", Wiley, 1985.

53. Park, C., AND Park. Y.: "An efficient algorithm for VLSI network partitioning problem using a cost function with balancing factor", IEEE Transactions on Computer-Aided Design 12, 11 (Nov. 1993), 1686-1694.

54. Picard, J. C. and Ratlifff, H. D. "Minimum cuts and related problems", Networks 5, 357-370. 1975.

55. Picard, J. C. and Queyranne, M.:" Selected applications of minimum cuts in networks", I.N.F.O.R: Can. Oper. Res. Inf. Proc.20, Nov., 394-422, 1982.

56. Pothen, Alex; "Graph Partitioning Algorithms with Applications to Scientific Computing"; Department of Computer Science; Old Dominion University; Norfolk; VA; pothen@cs.odu.edu.

57. Quintana, V.H., Simoes-Costa, A. and Mandel, A.:" Power System Topological Observability Using a Direct Graph-Theoretical Approach ", IEEE Trans. Power App. Sys, PAS-101, pp. 617-626, March 1982.

58. Ramanathan, A., and Colbourn, C.;" Counting almost minimum cutsets with reliability applications"; Math. Prog. 39, 3 (Dec.), 1987; 253-261.

59. Rao, V. B. and Arun, K. S.; " Constructive heuristics and lower bounds for graph partitioning based on a principal components approximation"; SIAM J. Matrix Annual. Appl., 14, 1993, pp. 991-1015.

60. Reid, J.K; "Large sparse sets of linear equations"; Academic Press, London, 1971.

61. Rose, James R.: " Sparse Matrix Computations", London, Academic Press, 1976.

62. Serre, Jean-Pierre: "Trees", Belin, New York: Springer-Verlag, 1980.

63. Sims, Chrles C.: "Computation With Finitely Presented Groups"; Cambridge: Cambridge University Press, 1994.

64. Strang, Gilbert: "Linear Algebra and its Application"; San Digo: Hartcourt, Brace, Jovanovich, 1988.

65. Saaki, H.; Aoki, K. and Yokoyama, R.: " A Parallel Computation Algorithm for Static State Estimation by means of Matrix Inversion Lamma", IEEE Trans. On Power System, Vol. PWRS-2, pp. 624-632, Aug 1987.

66. Sanchis, L. A. "Multi-Way Network Partitioning", IEEE Trans. on Computers, Vol.38, No.1, Jan 1989, pp.62-81.

67. Schweikert D. G. and Kernighan, B. W.; "A Proper Model for the Partitioning of Electrical Circuits", 9th Design Automation Workshop, 1972, pp. 57-62.

68. Suzukivi, Michio: "Group Theory 1"; Berlin, New York: Springer-Verlag, 1986.

69. Taylor, A.J.E.: "Techniques for Power System Simulation Using Multiple Processor" Ph.D. Thesis, University of Durham; 1990.

70. Thulasiraman, K. and Swamy, M. N. S.: "Graphs: Theory And Algorithm", New York: Chichester, Wiley, 1992.

71. Van Lint, J.H., Wilson, R. M.; "A Course in Combinatorics"; Cambridge University Press, 1992.

72. White, Peter: "Optimal Control"; Chichester: John Wiley & Sons, 1996.

73. Y. Wei and C. K. Cheng, "Toward Efficient Hierarchical Designs by Ratio Cut Partitioning", Proc. Int. Conf. on Computer-Aided Design, 1989, pp.298-301.

74. Y. Wei and C. K. Cheng, "Two-way Two-level Partitioning Algorithm," to appear in IEEE Int. Conf. on Computer Aided Design, 1990.

75. Van Cutsem, TH., and Ribbens-Pavella, M;: " Critical survey of hierarchical methods for state estimation of electrical power systems", IEEE Trans., 1983. PAS-102, pp. 3415-3424

Appendix A

The network basic definitions and notations

A.1 The network basic variables and properties

A network is a very simple structure [69], consisting of V, **a non-empty set of nodes** and E, **a non-empty set** of lines or **edges**, each of which links a pair of nodes. The direction of linkage from one node to another may or may not be important; if direction is important, the edge is said to be **directed**; if not, **undirected**.

The number of nodes and the number of edges are the two basic variables of a network. The notation **G = (V, E)** is used to denote a graph of a given network with V and E represent the sets of nodes and E edges respectively and with $n = |V|$ be the number of elements of V and $m = |E|$ be the number of elements of E. The number of nodes in a network is termed the **size** of the network.

An edge **e** is associated with exactly two **end-nodes** v and w. The notation e(v, w) is used to denote an edge e and its end-nodes v and w. The edge e (v, w) and the edge e (w, v) are the same edge, but with opposite direction. If e (v, w) belongs to G, then v and w are **adjacent** or **neighbouring**, nodes of G. Two edges are adjacent if they share a common end-node. Two edges are said to be **dependent** if they share one common node between them, otherwise they are said to be **independent**.

The degree $d(v)$ of a node v is equal to the number of edges connected to node v. The network degree D_G is the sum of the degrees of all nodes, i.e.

$$D_G = \sum_{i=1}^{n} d(v_i) = 2m \qquad (A.1)$$

A node of degree zero is termed **an isolated node**.

It is assumed that if two nodes are connected, the connection is made by one and only one edge. Thus, $n = O(D_G)$ and $m = O(D_G)$.

Further details about the graph network theory and its application may be found in a wide range of texts, of which references [21, 34, 69] are typical.

A.2 A walk, a path and a cycle

A **walk** W in a network is an alternating sequence of nodes and edges, for example $\{v_0, e_1, v_1, e_2, \cdots, e_j v_j\}$. A **path** is a sequence of nodes connecting two nodes via edges. The set of nodes of a path P has the form $V(P) = \{v_0, v_1, \cdots, v_j\}$. The set of edges of a path P has the form $E(P) = \{e_0, e_1, \cdots, e_j\}$; with $e_i = (v_i, v_{i+1})$. The nodes v_0 and v_j are termed **the end-nodes of P**. The number of edges in a path is called the **length** of the path. A path is termed **an open path** if **the end-nodes of P are different.** A path is termed **a closed path** if **the end-nodes of P are the same.** A **cycle** is a closed path over a set of nodes, V (P), such that the length of the closed path is minimum, and there is no closed path between any proper subset of nodes of V(P). Thus each cycle is independent i.e. each cycle in the network is represented by a different set of edges. A **cycle** is sometimes called a **circuit**. A graph without cycles is termed **a tree**.

A.3 Connectivity in Networks

A basic property a network may possess is that of being connected. A network is **connected** if it contains no isolated nodes or isolated sub-networks [49]. The isolated sub-networks are termed **components** of the network. The smallest component is the **isolated node**.

The concept of graph connectivity is very important in the DSE. If a network is not connected then it is not observable, and therefore, the DSE algorithm cannot be used, and all the presented techniques are not applicable. Consequently, it is assumed that the given network is connected.

A.4 Network Types

Sometimes networks are classified into types, according to known network properties. For example, if the edges of a network are identified with directions or with ordered pairs of nodes, the network is called **a directed network**. Otherwise the network is called an **undirected network**. A matrix can represent the connection of either network type.

Another property that used to classify the network types is the **cyclic property**. A network is said to be **acyclic** if it has no closed loops, i.e. no cycles. **A tree** is a typical acyclic connected graph. **A spanning tree** of a network is a tree that touches every node of the graph, and in this sense is the largest possible tree. A spanning tree has very well defined properties. It is a tree with n nodes and m-1 edges, and it may always be obtained by eliminating (m-n+1) edges from the network. In this thesis, the spanning tree and its properties has been used to partition the given network. More about the spanning tree and its properties are introduced in Chapter 4.

A bipartite network is a network G whose node set V can be partitioned into two non empty sets V_1 and V_2 in such a way that every edge of G joins a node in V_1 to a node in V_2.

A complete network is a network in which every node is connected to every node in the network. A complete network has $\frac{n(n-1)}{2}$ edges.

Appendix B

The NP-complete problem

B.1 The NP-complete

Problems are divided into two categories: those for which there exists an algorithm to solve it with **polynomial time complexity**, and those for which there is no such algorithm. The former class of problems are denote by P. There are problems, for which no known algorithm exists that solves it in polynomial time, but there is also no proof that no such algorithm exists. Among these problems that are not known to be in P (or in ~P), there is a subclass of problems known as NP-complete: those for which either all are solvable in polynomial time, or none are. Formally, a problem is NP if there exists an algorithm with polynomial time complexity that can certify a solution. For example, it is not known whether there exists a polynomial algorithm to solve a system of Diophantine equations, Ax=b for x in Z^n (integer n-vectors). However, we can certify any trial x in polynomial time, just by checking that it is in Z^n, then multiplying by A to compare with b. A problem, p, is NP-complete if it is NP and for any problem in NP, there exists a polynomial time algorithm to reduce it to p. A fundamental member of the NP-complete class is the **satisfiability problem**. It is an open question whether P=NP, and most regard the NP-complete problems as having exponential time complexity.

B.2 NP-hard

An optimization problem that relies upon the solution of an NP-complete problem. In that sense, NP-hard problems are at least as hard as NP-complete problems. Here are some NP-hard problems:

1. Bin packing
2. Covering, Cutting stock
3. Knapsack
4. Packing, Partitioning and Pooling
5. Traveling Salesman
6. Vehicle routing

Appendix C

C.1 The IEEE standard networks

The schematic diagrams of the 14-node, 30-node and 57-nodes IEEE standard networks are given in Figures C1, C2 and C3 respectively.

Figure C1 The IEEE-14 network

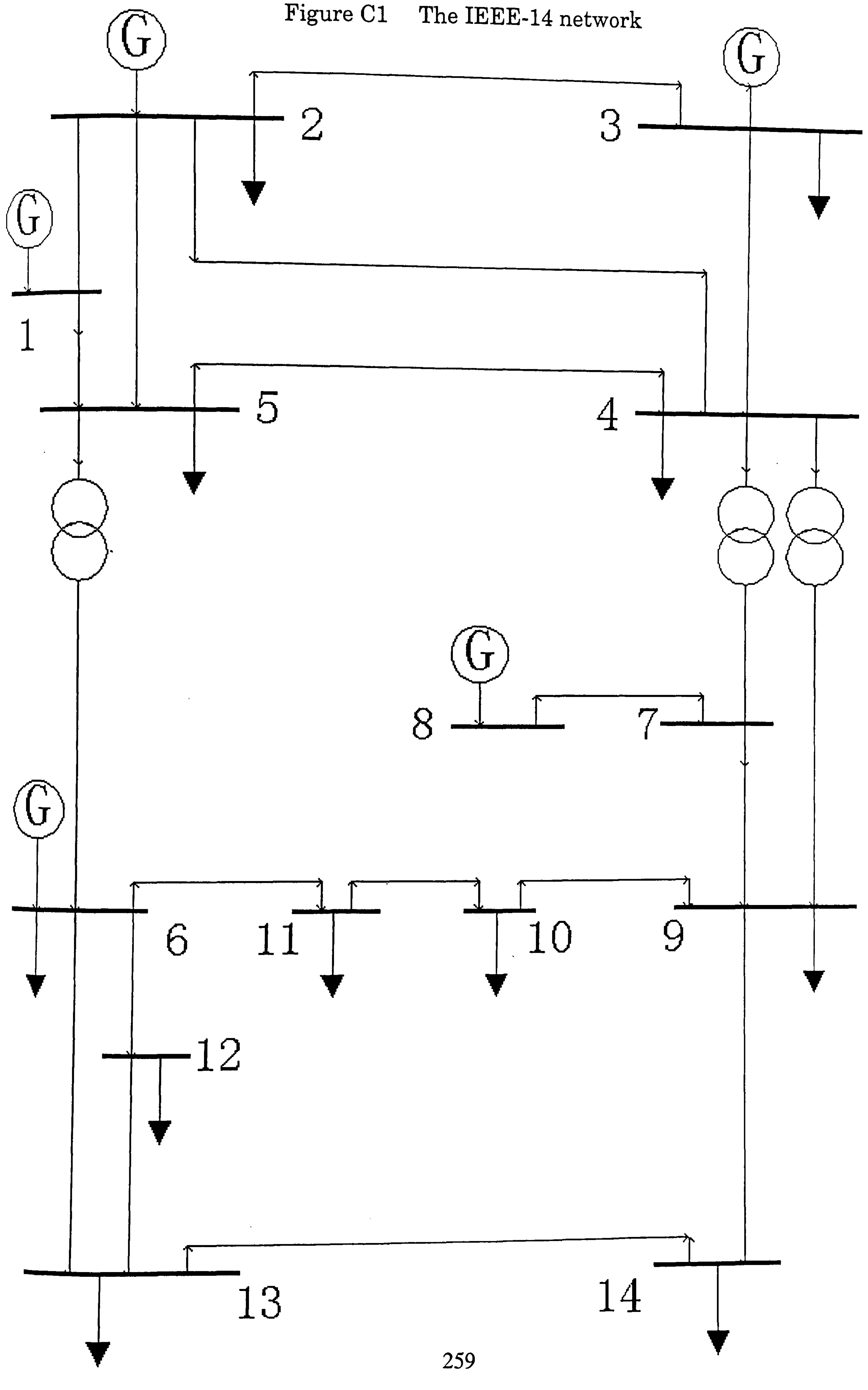


Figure C2 The IEEE-30 network

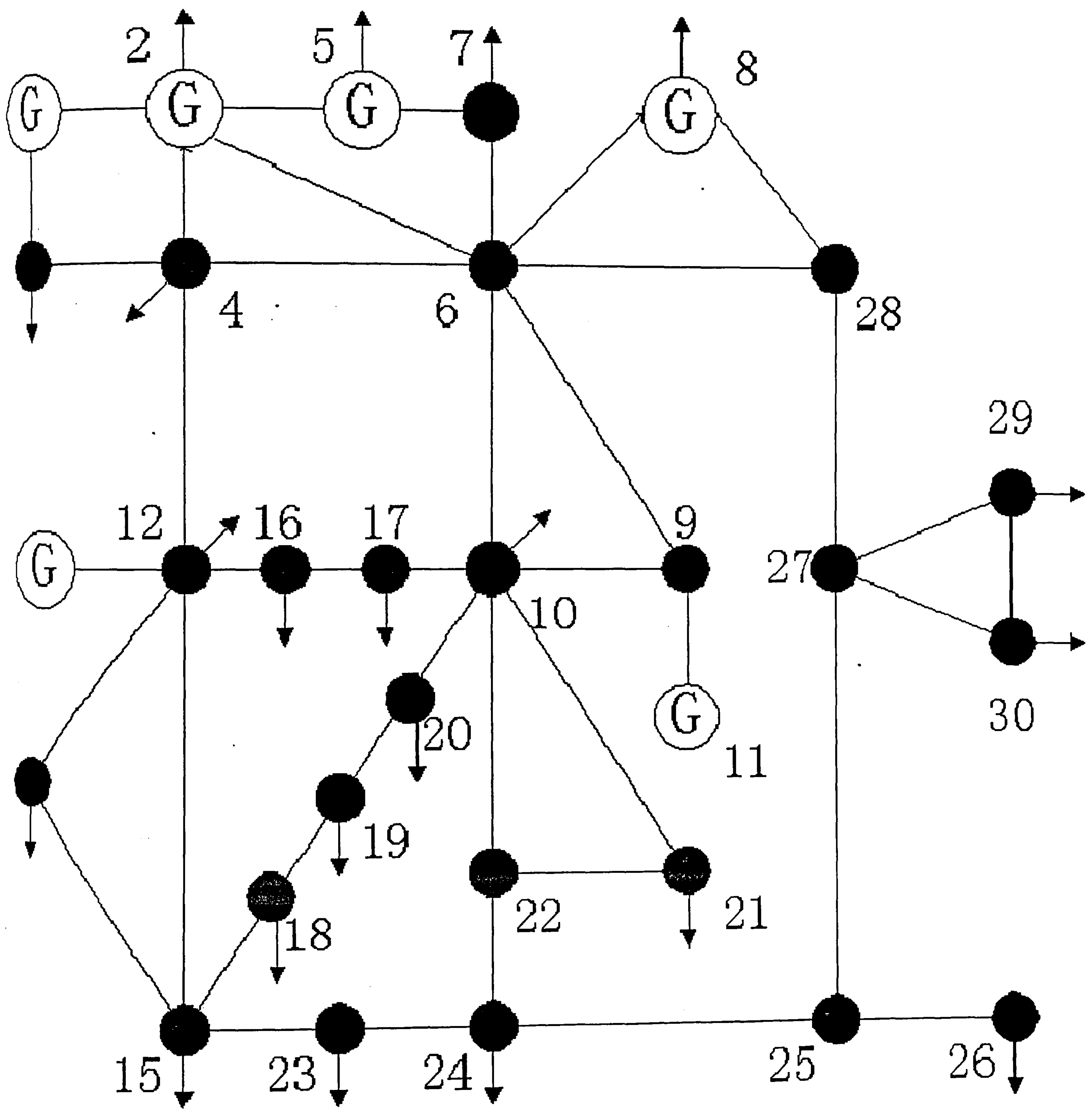
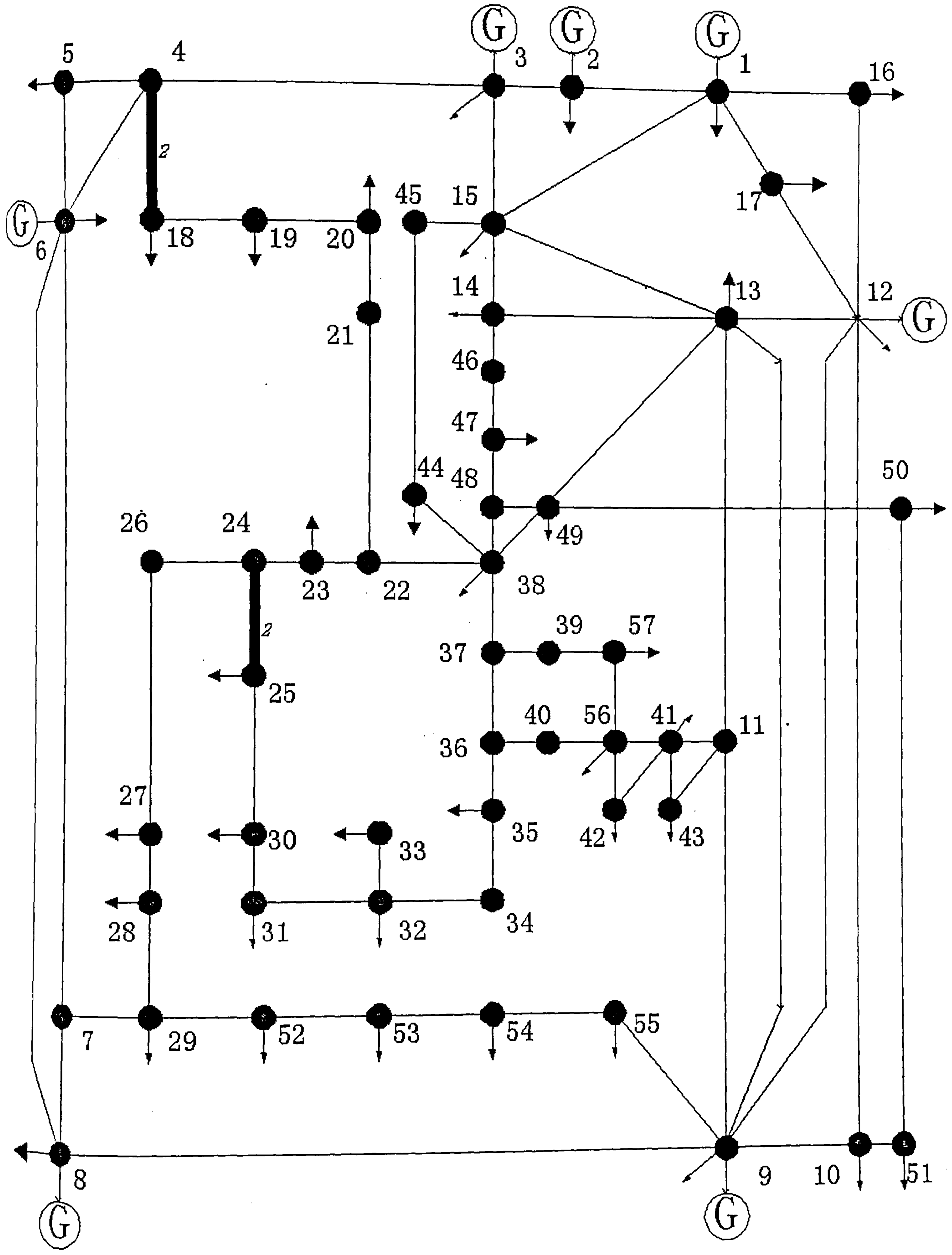


Figure C3 The IEEE-57 network


G
G
G
5
4
3
2
1
16
2
45
15
17
G
18
19
20
6
14
13
12
G
21
46
47
44
48
50
26
24
49
23
22
38
2
39
57
37
25
40
56
41
11
36
27
42
43
30
35
33
28
31
32
34
7
29
52
53
54
55
8
9
10
51
G
G