CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE

Reachability and Sequencing in Marked Graphs and State Graphs: Algorithms Based on Network Programming

Marc Andrew Comeau

A Thesis

in

The Department

of

Electrical Engineering

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

May 1986

ISBN 0-315-32244-6

ABSTRACT

Reachability and Sequencing in Marked Graphs and State Graphs: Algorithms Based on Network Programming

Marc Andrew Comeau, Ph.D.
Concordia University, 1986.

This thesis is concerned with an algorithmic study of reachability and sequencing problems in marked graphs and state graphs. Most of the results presented in the thesis are based on linear-programming formulations of relevant problems.

Our study of problems related to marked graphs is presented in Chapters 2 to 6. In Chapter 7, we discuss certain questions relating to state graphs.

In Chapter 2, we first give algorithmic proofs of reachability theorems for capacitated and uncapacitated marked graphs. We then define the concept of scatter of a firing sequence and consider the problem of designing minimum-scatter firing sequences executing a given firing-count vector. We conclude this chapter with a graph-theoretic characterization of the reachability problem for (0,1)-capacitated marked graphs.

In Chapter 3, we consider the problem of obtaining a maximum-weight marking reachable from a given initial marking. We formulate the problem as a linear program and give details of an algorithm based on the Simplex Method.

We offer interpretations, in terms of firing operations, to those which one encounters in the theory of network programming. Our study covers both live as well as nonlive capacitated marked graphs.

In Chapter 4, we study the submarking-reachability problem. We show that this problem reduces to an equivalent linear-programming problem defined on a smaller graph. We present an approach which unifies this study for both the capacitated and uncapacitated cases.

In Chapter 5, we prove the equivalence between the submarking-reachability problem and the problem of testing feasibility of the dual transshipment problem. We then give an algorithm, called Algorithm REACH, to solve this feasibility testing problem. Proof of correctness and complexity analysis of this algorithm are also presented.

In Chapter 6, we distribute Algorithm REACH. We present the several procedures constituting this distributed algorithm. The place of this distributed algorithm in the context of distributed algorithms for other graph problems as well as the possibilities it opens up for distributing general network optimization problems are discussed.

In Chapter 7, we study certain problems on state graphs dual to some of those discussed in the earlier chapters on marked graphs.

ACKNOWLEDGEMENTS

I wish to express my sincerest gratitude to my thesis supervisor, Dr. K. Thulasiraman for his superior guidance and insight throughout the span of this research. His support and interest were invaluable in the preparation of this work. It is my belief that it is a rare occasion for a graduate student to encounter a supervisor who so unselfishly devotes so much time and effort to a student's education as does Dr. Thulasiraman.

I also take pleasure in thanking Dr. S. Fleischer of the Technical University of Nova Scotia and Dr. J. F. Hayes of Concordia University for stimulating my interest in stochastic processes. Although I did not have an opportunity to pursue this line of research in my thesis, I feel that this is an avenue that I may follow.

I would especially like to thank Dr. J. C. Giguère for his patience and support during these seemingly never-ending final stages in the preparation of a thesis.

I would also like to thank Concordia University for its award of the University Graduate Fellowship Award for the period from September 1983 to August 1985.

Finally, I would like to thank all those persons, although not individually specified, who lent their support and helped make this thesis a reality.

DEDICATED TO MY LOVING PARENTS,

JEAN-MARIE AND MARGARET

AND TO FRANCES

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

A variety of models have been developed for, and applied to, the representation, analysis and specification of communications protocols and parallel computer programs. These models are largely language-theoretic and/or net-theoretic in nature. They include state-machine models, formal-language models, programming-language models, Petri-net models and mixed models [1], [2].

The central notion in the network-theoretic models is *concurrency* among discrete *events* and *conditions*. The conditions dictate which events are concurrently related and the events affect the conditions. Discrete events and conditions and their interactions are represented with a graph model such as the Petri net. It should be mentioned that a great deal of language-theoretic research has been centered around Petri nets. The net-theoretic or system-theoretic approach reflects the philosophy of Petri, Holt and other researchers involved in this area. Other graph models which are either generalizations, extensions, variants, or special cases of Petri nets include timed Petri nets, numerical Petri nets, complex bilogic or UCLA graphs, parallel-program schemata, computation graphs, state machines, and marked graphs. Reference to these and other models may be made through [1] and [2]. The inherent feature of these

graph models is that concurrency relations among events in the process being modeled evolve with the process as it executes. With the possible exception of timed Petri nets, the notion of time is absent in the net models. The net represents an interplay between states and activities and thus, the representation is independent of time. The passage of time only imposes a partial ordering on the occurence of events. Hence, all possible executions of a modeled system are represented in the net. This greatly facilitates the modeling of control in asynchronous discrete concurrent systems.

An important characteristic of these models, as with models in general, is the trade off between modeling power and analytic tractability. Questions of interest concerning Petri nets are, in general, exponentially hard to answer. Some important decision questions are even undecidable (an algorithm cannot be constructed to answer them) [1], [2]. Any significant extension or generalization of the Petri net tends to result in a Turing machine equivalence, and hence, intractability prevents easy analysis.

In the next four sections, we describe, briefly, the Petri net and three special cases of the net, namely, the state graph, the marked graph, and the computation graph.

1.1 Petri Nets

A Petri net is a directed graph G = (P,T,E) with a set

of vertices P, called places, a set of vertices T called transitions and a set of directed edges $E \subseteq (P \times T) \cup (T \times P)$ connecting places to transitions and vice-versa. In the terminology of graph theory, a Petri net is *bipartite* with *bipartition* (P,T) because the edge set E contains no edges from either P x P or T x T. The topology of a Petri net defines *input* and *output* sets for each place and transition as

$$I(p_j) \triangleq \{t_i \mid (t_i,p_j) \in E\},$$
$$O(p_j) \triangleq \{t_i \mid (p_j,t_i) \in E\},$$
$$I(t_j) \triangleq \{p_i \mid (p_i,t_j) \in E\},$$
$$O(t_j) \triangleq \{p_i \mid (t_j,p_i) \in E\}.$$

Figure 1.1 depicts a Petri net where places are represented with circles and transitions are represented with bars. Using the set notation, we have $P \triangleq \{p_1,p_2,p_3,p_4\}$, $T \triangleq \{t_1, t_2,t_3\}$, and $E \triangleq \{(p_1,t_1),(p_1,t_3),(p_2,t_2),(p_3,t_3),(t_1,p_2), (t_1,p_3),(t_2,p_1),(t_3,p_4)\}$ for the Petri net of Figure 1.1. By inspection, the input and output sets are

$$I(p_1) \triangleq \{t_2\}, \quad O(p_1) \triangleq \{t_1,t_3\},$$
$$I(p_2) \triangleq \{t_1\}, \quad O(p_2) \triangleq \{t_2\},$$
$$I(p_3) \triangleq \{t_1\}, \quad O(p_3) \triangleq \{t_3\},$$
$$I(p_4) \triangleq \{t_3\}, \quad O(p_4) \triangleq \emptyset,$$
$$I(t_1) \triangleq \{p_1\}, \quad O(t_1) \triangleq \{p_2,p_3\},$$
$$I(t_2) \triangleq \{p_2\}, \quad O(t_2) \triangleq \{p_1\},$$
$$I(t_3) \triangleq \{p_1,p_3\}, \quad O(t_3) \triangleq \{p_4\}.$$

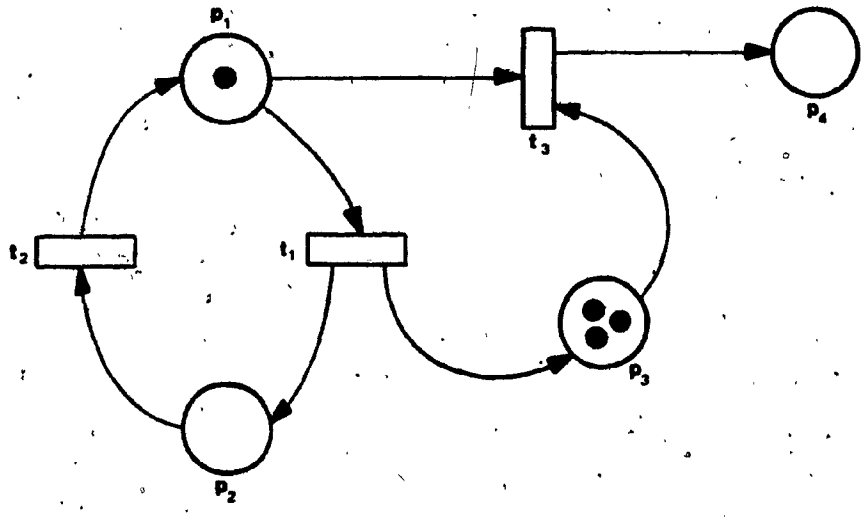


Figure 1.1

A Petri net with marking M

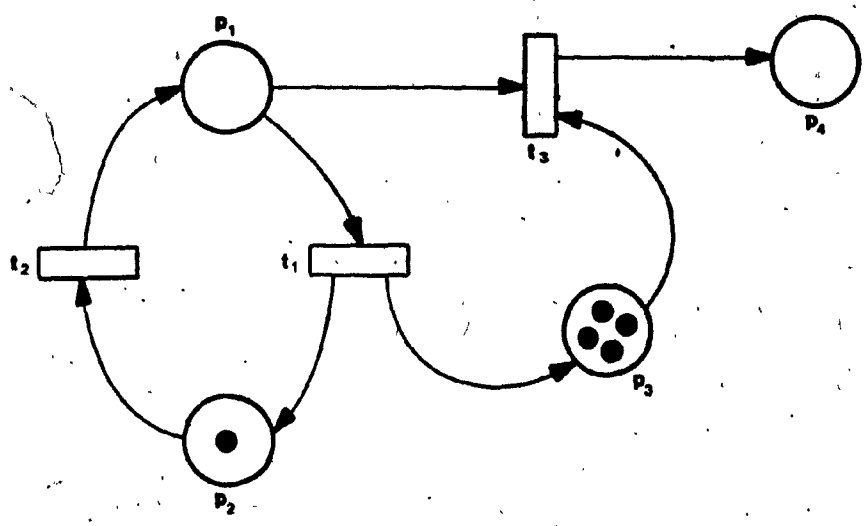


Figure 1.2

The marking after firing t_1 under M

Associated with each place $p_i \in P$ is a nonnegative integer $M(p_i)$ called the *marking* or *state* or *token-count* of place p_i. The distribution M of token counts is called the *marking* or *state* of the Petri net. In the Petri net of Figure 1.1, tokens are represented by the solid dots occupying the places. Thus, for this net, $M(p_1) = 1$, $M(p_2) = 0$, $M(p_3) = 3$ and $M(p_4) = 0$.

Associated with each transition $t_i \in T$ is a transition function $d_i(M)$ mapping M into a new marking M' resulting from the *firing* of transition t_i. The transition function $d_i(M)$ removes one token from each place in $I(t_i)$ and puts one token on each place in $O(t_i)$. Since M' must be nonnegative, a transition t_i can fire if and only if each place in $I(t_i)$ has at least one token under M. A transition $t_j \in T$ with $M(p_i) > 0$, $\forall\, p_i \in I(t_j)$ is said to be *enabled* in M and, hence, eligible for firing in M. Both transitions t_1 and t_3 are enabled in M in the Petri net of Figure 1.1. Note that transitions t_1 and t_3 are in conflict in the sense that although they are simultaneously enabled in M they cannot fire concurrently since the firing of either disables the other. The firing of t_1 under M results in the marking shown in Figure 1.2.

1.2 State Graphs

A Petri net $G = (P,T,E)$ is a *state graph* if

$$|I(t_j)| = |O(t_j)| = 1, \forall\, t_j \in T.$$

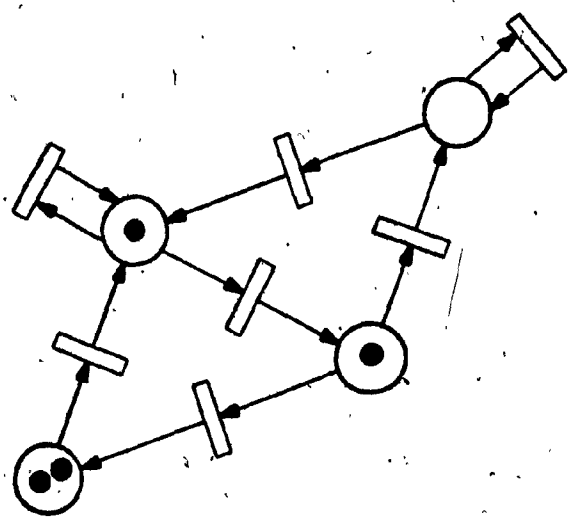


Figure 1.3

A state graph as a Petri net

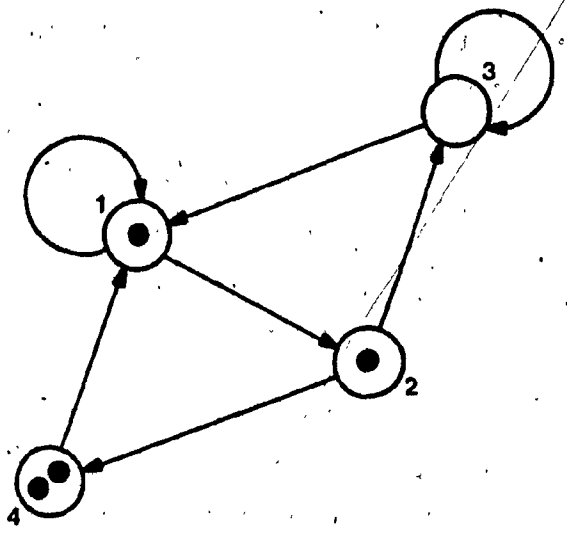


Figure 1.4

A state graph

Figure 1.3 illustrates a state graph. Since each transition has exactly one input place and one output place, then the bars may be dropped from the diagram and transitions may be represented simply with edges of a graph as shown in Figure 1.4. Thus, a state graph is a directed graph $G = (V,E)$ on vertex set V and edge set E, whose vertices represent places and whose edges represent transitions between places. The state graph derives its name from the *state machine* in which vertices represent *states* and edges represent possible transitions between states. The state graph is slightly more general in that vertices represent state variables. We think of the marking M (the values of the variables) as the state of G. A state machine is included in this definition as the special case of a state graph G containing exactly one token. The state graph illustrated in Figure 1.4, is defined by the sets

$$V \triangleq \{1,2,3,4\},$$

$$E \triangleq \{(1,1),(1,2),(2,3),(2,4),(3,1),(3,3),(4,1)\}$$

and has the marking $M(1) = 1$, $M(2) = 1$, $M(3) = 0$ and $M(4) = 2$. Transitions (1,1), (1,2), (2,3), (2,4) and (4,1) are enabled under M. The firing of a transition $e = (i,j) \in E$ simply moves one token from place i to place j. Intuitively, we see that the token movement induced by activity in a state graph behaves like a token flow. Indeed, a state graph models a closed network of queues with a single customer class.

1.3 Marked Graphs

A Petri net $G = (P,T,E)$ is a *marked graph* if

$$|I(p_j)| = |O(p_j)| = 1, \ \forall\ p_j \in P.$$

This definition is dual to that of a state graph. Figure 1.5 shows a marked graph. Since each place has exactly one input transition and one output transition, then the circles may be dropped from the diagram by representing the places with edges of a graph and the transitions with vertices as shown in Figure 1.6, where circles replace the bar representation of transitions. Thus, a marked graph is a directed graph $G = (V,E)$ on vertex set V and edge set E, whose vertices represent transitions and whose edges represent places interconnecting transitions. Tokens occupy the edges of a marked graph. The marked graph of Figure 1.6 is defined by the sets

$$V \triangleq \{1,2,3,4,5\},$$

$$E \triangleq \{(1,2),(1,3),(2,2),(2,3),(3,4),(4,4),(4,5),(5,2)\}$$

and has marking $M((1,2)) = 1$, $M((1,3)) = 0$, etc. The firing of a transition $v \in V$ in a marked graph G removes one token from each *input edge* of v and puts one token on each *output edge* of v. It should be clear from this definition that, in general, the token movement induced by activity in a marked graph differs radically in character from that in a state graph. In spite of this, the marked graph provides a model for a class of queueing networks in which every

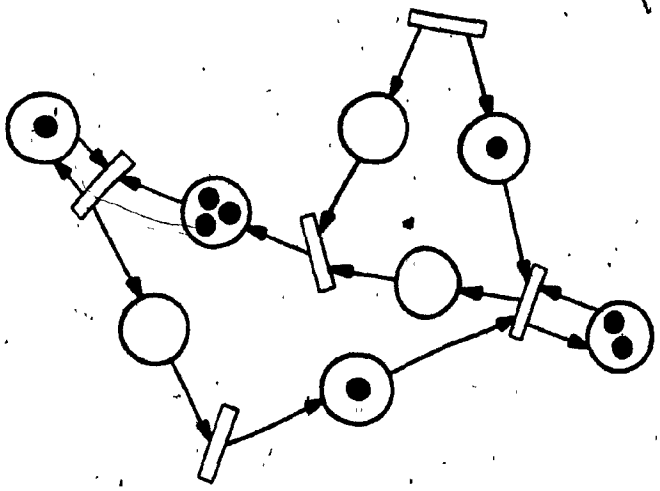


Figure 1.5

A marked graph as a Petri net

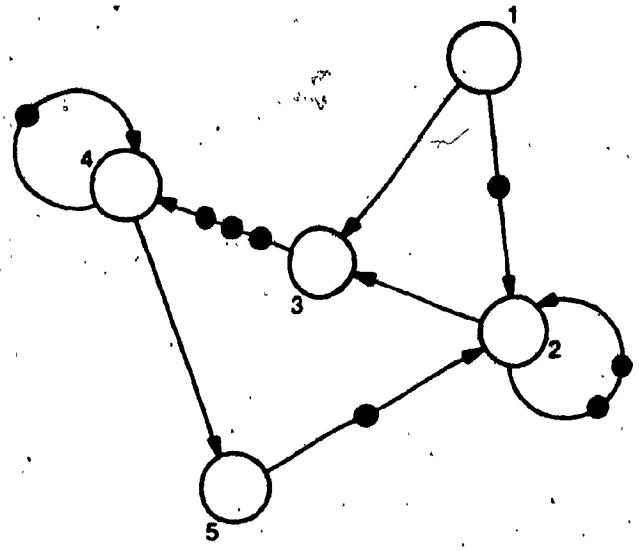


Figure 1.6

A marked graph

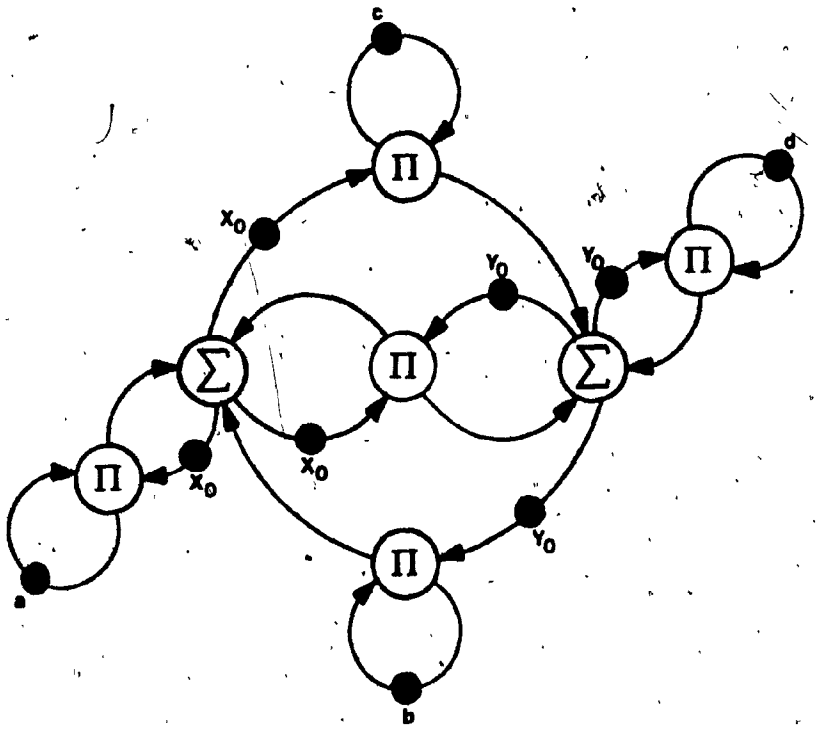
queue is fed from exactly one input process and feeds exactly one output process. Such a model arises as a restricted data-flow graph in which the flow of control has no data-dependent branches.

1.4 Computation Graphs

An extension of a marked graph called a *computation graph* was introduced by Karp and Miller [3] as a concurrent model for representing the flow of data-independent control inherent in a number of iterative type computations. A computation graph $G = (V,E)$ is a marked graph in which an *input quantum* $Q_I(e)$, an *output quantum* $Q_O(e)$ and a *threshold* $H(e)$ are associated with each edge $e \in E$. These quantities are defined as follows. If $e = (i,j) \in E$, then each firing of vertex i puts $Q_I(e)$ tokens on edge e. Similarly, $Q_O(e)$ tokens are removed from edge e each time vertex j fires. A vertex $v \in V$ is enabled under M if and only if $M(e) \geq H(e)$ on each input edge of vertex v. A marked graph is the special case of a computation graph when $Q_I(e) = Q_O(e) = H(e) = 1$, $\forall e \in E$. Figure 1.7 shows a computation graph representing the iterative solution to a set of difference equations.

1.5 Analysis of Marked Graphs

We now briefly introduce some important definitions and results which will be used in our presentations in the following chapters.



$$X_{k+1} = aX_k + bY_k + X_kY_k$$

$$Y_{k+1} = cX_k + dY_k + X_kY_k$$

Figure 1.7

A computation graph

The definitions for marked graphs that follow carry over to equivalent definitions for state graphs and Petri nets.

A *marked graph*, as defined in Section 1.3, is a directed graph $G = (V,E)$ with vertex set V, edge set E, a nonnegative-integer column vector M associated with E, called a *marking*, *state* or *token distribution* of G, and a *state-transition* function $\sigma_i(M)$ mapping M into a new marking M' resulting from firing vertex $i \in V$. The transition function subtracts one token from M on each edge incident into i and adds one token to M on each edge incident out of i, to obtain M'. Since M' must be nonnegative, $\sigma_i(M)$ can be applied only if M has a positive token count on each edge incident into vertex i. In other words, to be *legally fired*, a vertex must have at least one token on each one of its input edges. A vertex is said to be *enabled* under a marking M, if it is legally firable under M. The *enabling number* of vertex i is the minimum of the token counts on the edges incident into i.

A marking M' is *reachable* from a marking M if some sequence of legal firings will transform M into M'. The *reachability set* R(M) of a marking M is defined as the set of all markings reachable from M. Since the null sequence is trivially legal, $M \in R(M)$.

A marked graph G is *live* under a marking M if each

vertex $i \in G$ can be enabled through some legal firing sequence starting from M. The marking M is then called a *live marking*. Liveness is characterized in the following theorem [4], where the term *dead-subgraph* refers to a token-free directed circuit.

Theorem 1.1: A marked graph G is live under a marking M if and only if G contains no dead-subgraphs under M.■

A marked graph G is *bounded* under a marking M_o if the token count on each edge, $e \in G$, is finite in every marking in $R(M_o)$. Boundedness is guaranteed if and only if each edge of G belongs to a directed circuit with a finite token count. The marked graph is called k-*bounded* under M_o if $M(e) \leq k$, $\forall e \in G$, $\forall M \in R(M_o)$. A 1-bounded marked graph is called *safe*. The safeness condition is characterized in the following theorem [4], where the token count of a directed circuit refers to the sum of the tokens on the edges of the circuit.

Theorem 1.2: A marking of a marked graph is live and safe if and only if every edge in the graph belongs to a directed circuit with token count 1.■

Let G be a marked graph with an initial marking M_o and let $M \in R(M_o)$. Then, the *differential* marking $\Delta_M \triangleq M - M_o$ satisfies Kirchhoff's voltage law in G [5]. Thus, if B_f is a fundamental-circuit matrix of G, then

$$B_f \Delta_M = 0. \qquad (1.1)$$

This simple and elegant result, due to Murata, as profound as Tellegen's theorem, is the basis of almost all of the results in this thesis.

In view of Equation 1.1, we can consider the elements of Δ_M as *voltages* of the corresponding edges of G. Using well-known network-theoretic results, we can determine a set of *node voltages* $\{\sigma_1,\sigma_2,\cdots,\sigma_n\}$ of G such that $\sigma_i - \sigma_j = \Delta_M(e)$ where $e = (i,j) \in E$ is the edge directed from vertex i to vertex j and $\Delta_M(e)$ is the component of Δ_M corresponding to edge e. Let $\Sigma \triangleq [\sigma_1,\sigma_2,\cdots,\sigma_n]^t$ denote the column vector of σ_i's. Note that

$$A^t\Sigma = \Delta_M, \tag{1.2}$$

where A is the incidence matrix of G. If the smallest entry in Σ is not a zero, then we can obtain such a vector Σ_o by adding an appropriate number to all entries in Σ. It is easy to show that Σ_o also satisfies (1.2) and it is unique for a given value of Δ_M. The vector Σ_o is called the *minimum nonnegative-firing-count vector* and the σ_i's are called *firing numbers* or *firing counts*. The i^{th} element of Σ_o indicates the minimum number of times vertex i would fire in a firing sequence leading from M_o to M. The vertex with zero firing count will be referred to as a *datum*.

As an example, two markings M_o and M of a graph G are shown in Figure 1.8. The corresponding

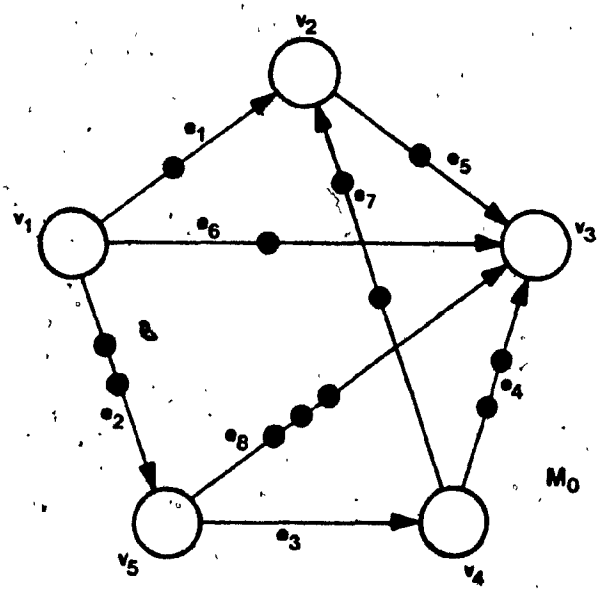


Figure 1.8(a)

Marking M_o

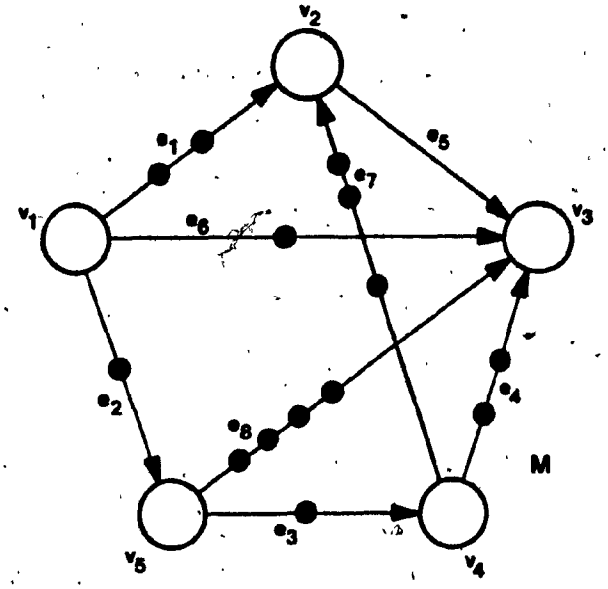


Figure 1.8(b)

Marking M

$$\Delta_M = [\ 1,-1,\ 1,\ 0,-1,\ 0,\ 1,\ 1]^t.$$

It can be verified that Δ_M satisfies Kirchhoff's voltage law in G. Furthermore,

$$\Sigma = [\ 0,-1,\ 0,\ 0,\ 1]^t,$$

satisfies Equation 1.2. Adding 1 to all of the elements in Σ, the minimum firing-count vector Σ_o is obtained as

$$\Sigma_o = [1,\ 0,\ 1,\ 1,\ 2]^t.$$

The following legal firing sequence executes Σ_o and takes M_o to M:

$$v_1^1 v_5^2 v_4^1 v_3^1,$$

where v_i^k refers to the operation of firing vertex v_i, k times, consecutively.

A nonnegative firing-count vector Σ is said to be *executable* from M_o if a legal firing sequence exists starting from M_o and its firing-count vector is Σ. Note that for any Δ_M satisfying Equation 1.1, existence of Σ satisfying Equation 1.2 is guaranteed. However, this alone does not guarantee executability of Σ. Executability of Σ_o, or, equivalently, reachability of M from M_o, is characterized in the following theorem [5]:

Theorem 1.3 (Reachability Theorem): Let M_o and M_r be two markings of a marked graph G. Let $\Delta_M = M_r - M_o$. M_r is

reachable from M_o if and only if

i) $B_r \Delta_M = 0$, and

ii) σ_u is zero for each vertex belonging to a dead-subgraph of G, where

$$\Sigma_o = [\sigma_1, \sigma_2, \cdots, \sigma_n]^t \geq 0$$

is the minimum nonnegative solution of

$$A^t \Sigma = \Delta_M . \blacksquare$$

A *capacitated marked graph* is a marked graph $G = (V,E)$ in which a lowerbound $L(e)$ and an upperbound $U(e)$ are specified on the token count $M(e)$ of each edge $e \in E$, for all markings of G.

A marking M of G is called *feasible* if and only if $L(e) \leq M(e) \leq U(e)$ $\forall e \in E$. We state this feasibility in vector notation as $L \leq M \leq U$. This definition reduces to the original definition when $L(e) = 0$ and $U(e) = \infty, \forall e \in E$. In a queueing network context, the upperbounds can represent storage capacities for the associated queues. Nonzero lowerbounds may seem meaningless in this context. However, we include them since they present no further complications and they represent the general case.

The *enabling number* of a vertex $v \in V$ under a marking M of a capacitated marked graph G is defined as

$$\mu_v = \min\{ \min_{e \in E_v^-} \{M(e) - L(e)\}, \min_{e \in E_v^+} \{U(e) - M(e)\}\}, \quad (1.3)$$

where E_v^- and E_v^+ are the input and output edge sets of vertex v, respectively. This reduces to the usual definition for uncapacitated graphs when $L(e) = 0$ and $U(e) = \infty$, $\forall e \in E$. A vertex v is *enabled* or *legally firable* under a marking M if its enabling number under M is greater than zero.

Let $C \subseteq E$ be a circuit of G and define an arbitrary orientation for C. Let C_+ and C_- denote the subsets of C consisting of all edges following and opposing that orientation, respectively. A *dead-subgraph* of a capacitated marked graph G under a marking M is either

i) an edge $e \in E$ with $L(e) = M(e) = U(e)$, or

ii) a circuit $C = C_+ \cup C_- \subseteq E$ such that either

$M(e) = L(e)$, $\forall e \in C_+$ and $M(e) = U(e)$, $\forall e \in C_-$,

or

$M(e) = U(e)$, $\forall e \in C_+$ and $M(e) = L(e)$, $\forall e \in C_-$

The first form of a dead-subgraph is trivial. An edge $e = (i,j) \in E$ with $L(e) = M(e) = U(e)$ forces $\mu_i = \mu_j = 0$ in every feasible marking of G.

A capacitated marked graph is *live* under a marking M if each vertex $i \in G$ can be enabled through some legal firing sequence starting at M. Liveness of a capacitated marked graph is characterized in the following theorem.

Theorem 1.4: A capacitated marked graph G is live under a marking M if and only if G has no dead-subgraphs under M.■

Necessary and sufficient conditions for reachability of capacitated marked graphs are given in the following theorem.

Theorem 1.5 (Capacitated-Reachability Theorem): Let M_o and M_r be two feasible markings of a marked graph G. Let $\Delta_M = M_r - M_o$. M_r is reachable from M_o if and only if

i) $B_f \Delta_M = 0$, and

ii) σ_k is zero for each vertex belonging to a dead-subgraph of G, where

$$\Sigma_o = [\sigma_1, \sigma_2, \cdots, \sigma_n]^t \geq 0$$

is the minimum nonnegative solution of

$$A^t \Sigma = \Delta_M . \blacksquare$$

We next give a brief review of the literature on the study of marked graphs. Broadly speaking, contributions to the theory of marked graphs and their extensions can be categorized into three groups: analysis, synthesis and applications.

As regards analysis, a number of interesting results have been reported [1] •• [35]. Of special relevance to the work presented in this thesis are [3] •• [7]. In [3], Karp and Miller have considered several problems related to the computation graphs which are a generalization of the marked graph. In [4], Commoner, Holt, Even and Pnueli have pre-

sented, among other things, an algorithmic approach to the reachability problem and have also studied the maximum-marking problem. As we have already mentioned, Murata [5] has presented a circuit-theoretic approach to the study of the reachability problem. Recently, Kumagai, Kodama and Kitagawa [6], [7] have introduced and studied the submarking reachability problem. Related to the problem of reachability is that of controllability which has been considered in [8] and [9]. Different classes of extended marked graphs have been proposed and investigated in [10] -- [12]. An *extended marked graph* is an augmented marked graph having specified arcs between some pairs of places and transitions to represent the permissive controlling function for firings of the corresponding transitions.

As regards synthesis problems related to marked graphs, in [5] has been considered the problem of finding a marked graph which possesses a given set of markings as set(s) of specified mutually-reachable markings. Certain reduction and expansion techniques useful in analysis and synthesis have been presented in [13] and [14]. In [36], Murata has presented a method for synthesizing or growing live and safe marked graph models of decision-free concurrent computations. In this synthesis, certain properties of the required marked graph such as liveness, safeness, the number of reachability classes etc., can be prescribed. In [37] has been considered the problem of

realizing the *synchronic distance matrix* of a marked graph. Suzuki and Murata have presented a refinement technique in [16] which can be used as a top-down approach for synthesizing Petri-net models of concurrent systems. Recently, Datta and Ghosh [38] have introduced a new class of Petri nets called *regular nets* and have presented a systematic method for synthesizing regular nets.

Several applications of Petri nets and their variants have been reported [39] -- [65]. These applications include performance evaluation, communications protocols, computer software, robotics, etc.

1.6 Scope of the Thesis

This thesis is concerned with an algorithmic study of reachability and sequencing problems in marked graphs and state graphs. Most of the results presented in the thesis are based on linear-programming formulations of relevant problems.

Our study of problems related to marked graphs is presented in Chapters 2 to 6. In Chapter 7, we discuss certain questions relating to state graphs.

In Chapter 2, we first give algorithmic proofs of reachability theorems for capacitated and uncapacitated marked graphs. We then define the concept of scatter of a firing sequence and consider the problem of designing

minimum-scatter firing sequences executing a given firing-count vector. We conclude this chapter with a graph-theoretic characterization of the reachability problem for (0,1)-capacitated marked graphs. This result has been motivated by results presented in [66] and [67].

In Chapter 3, we consider the problem of obtaining a maximum-weight marking reachable from a given initial marking. We formulate the problem as a linear program and give details of an algorithm based on the Simplex Method. We offer interpretations, in terms of firing operations, to those which one encounters in the theory of network programming. Our study covers both live as well as nonlive capacitated marked graphs.

In Chapter 4, we study the submarking-reachability problem. We show that this problem reduces to an equivalent linear-programming problem defined on a smaller graph. We present an approach which unifies this study for both the capacitated and uncapacitated cases.

In Chapter 5, we prove the equivalence between the submarking-reachability problem and the problem of testing feasibility of the dual transshipment problem. We then give an algorithm, called Algorithm REACH, to solve this feasibility testing problem. Proof of correctness and complexity analysis of this algorithm are also presented.

In Chapter 6, we distribute Algorithm REACH. We

present the several procedures constituting this distributed algorithm. The place of this distributed algorithm in the context of distributed algorithms for other graph problems as well as the possibilities it opens up for distributing general network optimization problems are discussed.

In Chapter 7, we study certain problems on state graphs dual to some of those discussed in the earlier chapters on marked graphs.

Finally, in Chapter 8, we summarize our contributions in this thesis and point out several open problems for future research.

Chapter 2

REACHABILITY AND SEQUENCING PROBLEMS ON MARKED GRAPHS

In this chapter, we study certain questions relating to the reachability problem on marked graphs. We first give an algorithmic proof of the reachability theorems (Theorems 1.3 and 1.5). We then define the concept of scatter of a firing sequence and discuss algorithms for constructing firing sequences with minimum scatter, executing a given executable minimum firing-count vector. Algorithms for different classes of graphs are developed. Finally, we study the structure of the reachability problem on (0,1)-capacitated marked graphs and give a graph-theoretic characterization of this problem. Our study here is motivated by similar results reported in [66] and [67].

2.1 Executability of a Firing-Count Vector

In this section, we give an algorithmic proof of the reachability theorem (Theorem 1.3) due to Murata [5]. The proof here is based on that of Theorem 5 in [4]. We then extend this proof to the capacitated case.

Theorem 2.1 (Reachability Theorem): Let M_o and M_r be two markings of a marked graph G. Let $\Delta_M = M_r - M_o$. M_r is reachable from M_o if and only if

i) $B_f \Delta_M = 0$, and (2.1)

ii) σ_k is zero for each vertex belonging to a dead-subgraph of G, where

$$\Sigma_o = [\sigma_1, \sigma_2, \cdots, \sigma_n]^t \geq 0$$

is the minumum nonnegative solution of

$$A^t \Sigma = \Delta_M. \tag{2.2}$$

Proof:

Necessity: Obvious.

Sufficiency: We may assume that $\Sigma_o \neq 0$. Otherwise, $M_o = M_f$ and the case is trivial. To prove the theorem, we shall first show that there exists a legally firable or enabled vertex with its firing count nonzero. Consider any $\sigma_{i_1} > 0$. If vertex i_1 is not enabled, then there exists an edge $e = (i_2, i_1)$, directed from some vertex i_2 to vertex i_1, with $M_o(e) = 0$. Clearly, $\sigma_{i_2} > 0$. If vertex i_2 is also not enabled, then repeat this process until a sequence of vertices $i_1, i_2, \cdots, i_{k-1}, i_k$ is located such that:

- $i_1, i_2, \cdots, i_{k-1}$ are all distinct;
- $\sigma_{i_j} > 0$, for $j = 1,2,\cdots,k$;
- for each edge $e = (i_{j+1}, i_j)$, $j = 1,2,\cdots,k-1$, directed from vertex i_{j+1} to vertex i_j, $M_o(e) = 0$.

Case 1: vertex i_k is legally firable or

Case 2: $i_k = i_j$, for some $j \in \{1,2,\cdots,k-2\}$.

One of the two cases above should occur because the graph G is assumed to be finite. If Case 1 occurs, then an enabled vertex with nonzero firing count has been found. Case 2 cannot occur for that would mean the existence of a token-free directed circuit (dead subgraph) $i_{\mu-1}, i_{\mu-2}, \cdots, i_j, i_{\mu-1}$ with each of the vertices on this circuit having a nonzero firing count, thereby contradicting the hypothesis of the theorem.

Now, fire the vertex i_μ. This would result in a new marking M_1 and a new firing-count vector Σ_1. The marking M_1 is obtained by increasing by one the token counts of all the edges directed away from i_μ, and by decreasing by one the token counts of all the edges directed into i_μ. Also, Σ_1 is obtained from Σ_0 by subtracting one from σ_{i_μ}. Clearly, M_1, Σ_1, and the vector $M_r - M_1$ satisfy the hypothesis of the theorem. If $M_1 = M_r$, then $\Sigma_1 = 0$ and we have established the reachability of M_r from M_0. Otherwise, locate an enabled vertex in M_1 with a nonzero firing count in Σ_1 and fire this vertex. Repeat this process until a marking equal to M_r is reached, with a firing-count vector Σ_r equal to zero. The hypothesis of the theorem guarantees the termination of this process and, hence, the executability of Σ_0 leading from M_0 to M_r.■

<u>Theorem 2.2</u> (Capacitated-Reachability Theorem): Let M_0 and M_r be two feasible markings of a capacitated marked graph G. Let $\Delta_M = M_r - M_0$. M_r is reachable from M_0 if and only if

i) $B_f \Delta_M = 0$, and

ii) σ_k is zero for each vertex belonging to a dead-subgraph of G, where

$$\Sigma_o \triangleq [\sigma_1, \sigma_2, \cdots, \sigma_n]^t \geq 0$$

is the minimum nonnegative solution of

$$A^t \Sigma = \Delta_M.$$

Proof:

Necessity: Obvious.

Sufficiency: An edge $e \in E$ is *depleted* whenever $M(e) = L(e)$ and is *saturated* whenever $M(e) = U(e)$. In these terms, we may say that a vertex of G is legally firable or enabled under a marking M if and only if it has no depleted input edges and no saturated output edges.

Proceeding along the same lines as in the proof of the uncapacitated reachability theorem (Theorem 2.1), we shall first demonstrate the existence of an enabled vertex under M_o with nonzero firing count. Consider any $\sigma_{i_1} > 0$. If vertex i_1 is not enabled in M_o, then there exists either a depleted edge $e_1 = (i_2, i_1)$ or a saturated edge $e_1 = (i_1, i_2)$. Note that $\sigma_{i_2} > 0$. If vertex i_2 is also not enabled under M_o, then there exists either a depleted edge $e_2 = (i_3, i_2)$ or a saturated edge $e_2 = (i_2, i_3)$. Repeating this process identifies a sequence of vertices $i_1, i_2, \cdots,$

i_{k-1}, i_k and edges $e_1, e_2, \cdots, e_{k-1}$ such that

- $i_1, i_2, \cdots, i_{k-1}$ are all distinct;
- $\sigma_{i_j} > 0$, for $j = 1,2,\cdots,k$;
- for each edge e_j, $j = 1,2,\cdots,k-1$
  either $e_j = (i_j, i_{j+1})$ and e_j is saturated
  or $e_j = (i_{j+1}, i_j)$ and e_j is depleted.

Two cases arise for finite graphs:

Case 1: vertex i_k is enabled or

Case 2: $i_k = i_j$, for some $j \in \{1,2,\cdots,k-2\}$

In Case 1, we have located an enabled vertex of G under M_o. Case 2 does not occur for that would imply the existence of a dead-subgraph, (the circuit $i_{k-1}, i_{k-2}, \cdots, i_j, i_{k-1}$) in which all the vertices have a nonzero firing count. The remainder of the proof follows exactly as in the uncapacitated case.■

2.2 Scatter in a Firing Sequence

A *firing traversal* of a marked graph is a traversal of the vertices of the graph which executes a given legal firing sequence. Thus, a firing traversal must visit vertices of a marked graph in the order dictated by the legal firing sequence and update the marking by firing the vertices. With this algorithmic definition of a firing traversal, equivalent firing sequences leading from a common initial marking to the same final making may be

studied. Consider executing the firing sequence $a^2b^3ac^4$. This representation is used to indicate that the vertex labeled a is visited and fired two times, consecutively, then vertex b is visited and fired three times, consecutively, etc. Among all possible legal firing sequences between two markings on a marked graph, which of these will require a minimum number of vertex visits in a firing traversal of the marked graph? This questions leads us to the concept of scatter in a firing sequence.

The *scatter* of a firing sequence F, which we denote by scatter(F), is the difference between the number of vertex visits needed to execute F, and the number of distinct vertices visited in a firing traversal. For example, if $F = a^2b^3ac^4$, then scatter(F) = 1 because a firing traversal with F would require four vertex visits (a,b,a,c) and three distinct vertices a, b and c would be visited. Thus, the scatter is a nonnegative integer.

Note that in this definition of scatter, consecutive firings of a vertex are considered to occur during the same visit of that vertex. An algorithm to perform a firing traversal might use a representation of F as an ordered list of pairs of the form (label, coefficient). The scatter in F would then be the number of pairs in a firing sequence minus the number of distinct labels. If the scatter of a firing sequence is zero, then a firing traversal executing it will visit each vertex at most once. For

a general graph, a zero-scatter firing sequence is not always present between an initial marking and a reachable final marking. This leads to the problem of finding a minimum-scatter firing sequence leading from some initial marking to a reachable final marking.

We now discuss an application which provides the motivation for introducing the concept of scatter. Consider modeling a general industrial environment with a marked graph as shown in Figure 2.1. Each vertex in this graph represents an activity. Vertices 1 to k represent *commodity sources* of possibly different commodities being introduced into the environment from the external world. Vertices 1' to p' represent *commodity sinks* or distribution centers to the external world. The token count on edge (i,j) incident into vertex j represents the number of units of some commodity made available for processing at vertex j. For an activity i to occur, there must be at least one unit of all the required commodities which, in fact, corresponds to having at least one token on each input edge of vertex i. Occurrence of an activity i, thus, corresponds to firing vertex i in the marked graph. Firing a vertex consumes one unit of commodity from each input edge and produces one unit of commodity on each output edge. If edge (i,j) is incident out of vertex i, then completion of activity i makes one unit of some commodity available for processing at vertex j. It may be noted that initially

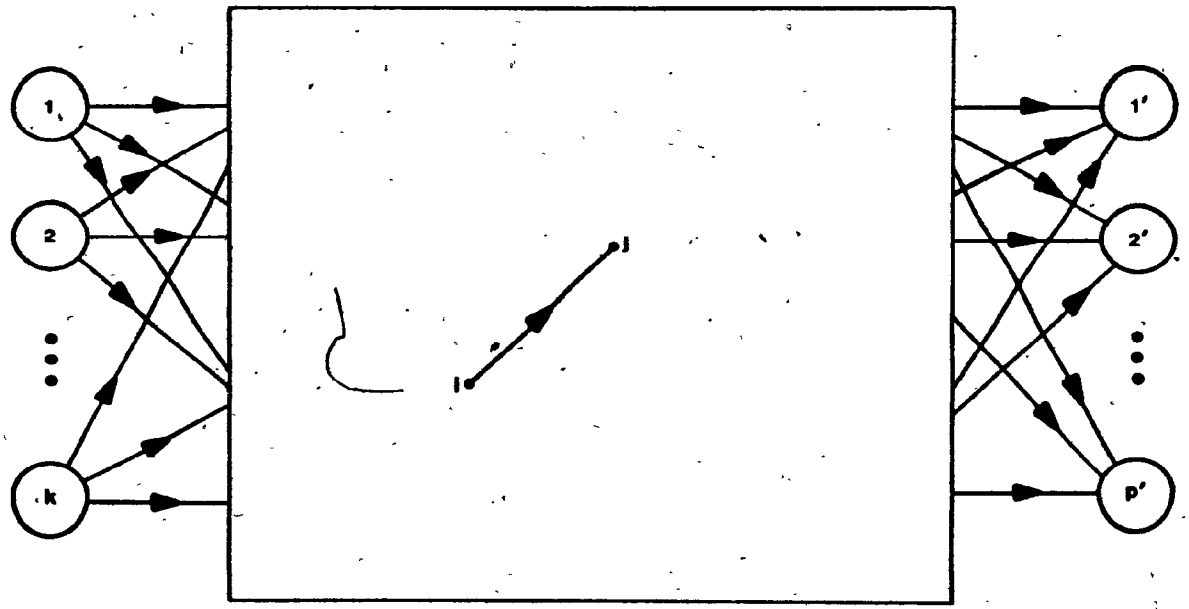


Figure 2.1

Modeling with a marked graph

certain edges may have a nonzero token count. This may represent the initial state of the environment. The final state corresponds to a predefined level of production, made available for consumption to the external world through the distribution centers. In other words, in the final state, the edges incident into the sinks will be required to have certain specified numbers of tokens.

At each vertex, a *machine* is available for carrying out the corresponding activity. Starting a machine from the *shutdown* state may involve considerable overhead cost. It is obviously cost-effective to minimize the idle running times of the machines, so, a machine at any vertex is shut down whenever that vertex cannot be activated. Thus, it is desirable to minimize the number of machine starts. It should be easy to see that the number of visits in a firing sequence leading from the initial marking to the final marking is a measure of the overhead cost incurred in reaching the desired production level. This motivates the introduction of the concept of scatter as well as the problem of determining minimum-scatter firing sequences. This model can be further generalized by modifying the firing rule or by incorporating a set of firing rules or semantics [68] as described for the design of a distributed operating system kernel.

To illustrate the concept of scatter in firing sequences, consider the markings M_0 and M_{10} of the directed

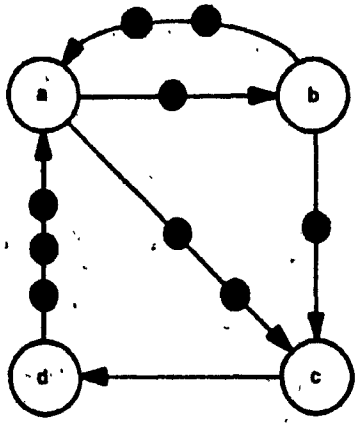


Figure 2.2(a)

Marking M_o

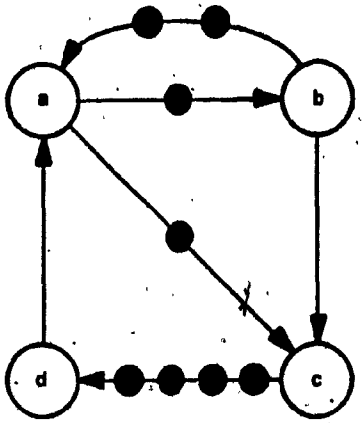


Figure 2.2(b)

Marking M_{1o}

graph shown in Figure 2.2. The firing-count vector $\Sigma_o \triangleq [\sigma_a^o, \sigma_b^o, \sigma_c^o, \sigma_d^o]^t = [3,3,4,0]^t$ is executable from M_o and leads to M_{10}. Some possible legal firing sequences are $a^2b^3ac^4$, $ba^3b^2c^4$, $ca^2b^3ac^3$ and $ab^2a^2bc^4$. The first two of these have scatter one and the last two have scatter two. The sequences $a^2b^3ac^4$ and $ba^3b^2c^4$ are minimum-scatter firing sequences executing Σ_o from M_o.

The *effective enabling number* of a vertex v is the minimum of the enabling and firing numbers of v. Thus, the effective enabling number of vertex v at any visit in a firing traversal is the maximum number of times v can be legally fired during that visit without exceeding the firing number at that visit. The *disabling number* of a vertex v, at any visit, is the difference between its firing and effective enabling numbers at that visit.

Before presenting the algorithms to determine minimum-scatter firing sequences, a few observations can be made about marked graphs.

1. Since a *datum* vertex always exists, at least one of the firing counts is zero. That is, at least one vertex of the graph need not be fired to bring any initial marking to a reachable final marking.

2. The markings on all input edges to a datum vertex must either remain the same or strictly increase from their initial values in any legal firing sequence.

3. The markings on all output edges of the datum vertex must either remain the same or strictly decrease from their initial values in any legal firing sequence.

4. If, at any visit during the generation of a firing sequence, some vertex is encountered that is enabled to its corresponding firing number, then it can be fired repeatedly until its firing number has been satisfied and then removed from the graph together with all edges incident on it. The problem then reduces to finding the remainder of the firing sequence on the resulting subgraph.

5. A *source* vertex (that is a vertex with no input edges) can be fired independently since it has no input edges. Thus, a source need be visited only once in a firing traversal.

2.3 The Greedy Firing Policy

In the example of the previous section, the effective enabling number λ_a of vertex a under the marking M_o is given by

$$\lambda_a \triangleq \min\{\sigma_a^o, M_o(e_1), M_o(e_2)\} = 2.$$

There are two possible ways of firing vertex a if it is visited under the marking M_o. The two firing sequences $ab^2a^2bc^4$ and $a^2b^3ac^4$ are examples of both. In general, if λ_i is the effective enabling number of vertex v_i under a marking M, then there are λ_i possible ways to fire vertex

v_i if it is visited under M (since it can be fired up to λ_i times and it must be fired at least once during a visit).

An algorithm that generates minimum-scatter firing sequences must employ some vertex-firing policy at each vertex visit in order to alleviate the ambiguity associated with multiply-enabled vertices. An obvious choice is to fire each vertex as much as possible at each visit. Before employing such a greedy firing policy, we must show that we do not overlook all optimal solutions by examining only those sequences which fire a vertex to its effective enabling number at each visit. In other words, we must prove that a minimum-scatter legal firing sequence exists between any reachable markings, on a marked graph, in which each visit fires some vertex to its effective enabling number at that visit of the vertex. Such a firing sequence will be referred to as a *greedy firing sequence*. Unless explicitly stated, all firing sequences referred to will be assumed to be legal.

Theorem 2.3: For any executable firing-count vector Σ_o, from a marking M_o on a marked graph G, there exists a greedy minimum-scatter firing sequence F_g, executing Σ_o from M_o, in which each visit of a vertex fires the vertex to its corresponding effective enabling number.

Proof: Let F be any minimum-scatter sequence executing Σ_o from M_o and let ℓ denote the total number of visits in F.

If F is not a greedy sequence then let v_i be the first vertex of G which F fails to fire to its effective enabling number at, say, the u^{th} visit. Construct another legal firing sequence F_g^u which is identical to F in the first u-1 visits and which fires vertex v_i to its effective enabling number λ_i at the u^{th} visit. The sequence F_g^u is u visits long and greedy at each visit. The effective enabling numbers of all vertices of G except vertex v_i after F_g^u executes from M_o, are greater than or equal to those after the first u visits of F from M_o. Thus, if F fires vertex v_j, $j \neq i$, at the $(u+1)^{th}$ visit, then we may legally construct the firing sequence F_g^{u+1} defined by the relation

$$F_g^{u+1} \triangleq F_g^u v_j^{\lambda_j^{u+1}}, \qquad (2.3)$$

where λ_j^{u+1} is the effective enabling number of vertex v_j at the $(u+1)^{th}$ visit. The firing sequence F_g^{u+1} visits vertices in the same order as the first u + 1 visits of F and it is greedy at each visit. Since the vertex v_i is not fired to its effective enabling number at the u^{th} visit in F, it follows that v_i will be visited at least once after the u^{th} visit in F, Clearly, $v_j \neq v_i$. So, F must be at least u + 2 visits long. In other words, $\ell \geq u + 2$.

Now, two cases may arise; F either returns to vertex v_i or visits some other vertex v_k, $k \neq i,j$ at the $(u+2)^{th}$ visit.

Case 1: If F returns to v_i at the $(u+2)^{th}$ visit, then firing vertex v_j at the $(u+1)^{th}$ visit must have increased the effective enabling number of vertex v_i since, by hypothesis, F has minimum-scatter and, therefore, does not visit vertices redundantly. So, the greedy firing sequence F_g^{u+1} which left vertex v_i disabled after the u^{th} visit must have reenabled it after firing vertex v_j at the $(u+1)^{th}$ visit and, therefore, it is legal to construct the sequence F_g^{u+2} defined by the relation

$$F_g^{u+2} \triangleq F_g^{u+1} v_i^{\lambda_i^{u+2}}, \tag{2.4}$$

where λ_i^{u+2} is the effective enabling number of vertex v_i at the $(u+2)^{th}$ visit.

Case 2: If F fires vertex v_k, $k \neq i, j$ at the $(u+2)^{th}$ visit, then by the previous argument, we may legally construct the sequence F_g^{u+2} defined by the relation

$$F_g^{u+2} \triangleq F_g^{u+1} v_k^{\lambda_k^{u+2}}, \tag{2.5}$$

where λ_k^{u+2} is the effective enabling number of vertex v_k at the $(u+2)^{th}$ visit.

In either case, F_g^{u+2} visits vertices in the same order as the first $u + 2$ visits of F and it is greedy at each visit. By repeated application of the above construction, we will arrive at a sequence $F_g^a = F_g$ executing Σ_o from M_o in which each visit of a vertex fires the vertex to its corresponding effective enabling number.■

Since the construction described in the proof of the above theorem applies to any minimum-scatter firing sequence executing Σ_o from M_o, we have the following corollary.

Corollary 2.3.1: For every minimum-scatter firing sequence F executing Σ_o from M_o on a marked graph G, there is a greedy minimum-scatter firing sequence F_g executing Σ_o from M_o which visits vertices in the same order as F does.■

This theorem implies that if we employ a greedy firing policy at each vertex visited when executing some firing-count vector, then it is possible to execute that vector with minimum scatter by visiting the vertices properly. In other words, we cannot obtain a firing sequence with less scatter by leaving a vertex effectively enabled after visiting and firing it, than we can by firing it to its effective enabling number at that visit. However, being less than greedy at each visit when firing vertices may lead to unnecessary scatter while searching for firing sequences, as illustrated by the firing sequences $ab^2a^2bc^4$ and $a^2b^3ac^4$ for the marked graph of Figure 2.2. Therefore, this is sufficient grounds for employing the greedy firing policy at each vertex while searching for a minimum-scatter firing sequence.

The firing sequence $ca^2b^3ac^3$ for the marked graph of Figure 2.2 shows that arbitrarily applying the greedy ver-

tex-firing policy over the vertices of a graph may not produce minimum-scatter firing sequences. Thus, along with the greedy vertex-firing policy, some vertex-visiting policy is needed to characterize an algorithm which generates a minimum-scatter legal firing sequence. We pursue this further for general graphs. First, we obtain results for marked graphs with different topological properties.

2.4 Minimum-Scatter Firing Sequences

This section examines the problem of determining a minimum-scatter firing sequence executing a given executable firing-count vector from a given initial marking on marked graphs with different topological complexities. It is shown that any executable firing-count vector, from a given initial marking on an acyclic marked graph, always possesses a zero-scatter firing sequence. This is shown to be true for the simple directed circuit but not for a general topology with directed circuits. Graphs in which all the directed circuits are vertex disjoint are then examined. An algorithm is given for each case.

2.4.1 Marked Acyclic Directed Graphs

By definition, an *acyclic* directed graph has no directed circuits. Thus, on an acyclic marked graph G, a marking M_r is reachable from an initial marking M_o if and only if KVL is satisfied by the differential markings. In an acyclic directed graph, there is at least one vertex

with zero in-degree called a *source* and at least one vertex with zero out-degree called a *sink*. A marked graph with the acyclic property must, therefore, possess at least one source and at least one sink. A source is independently firable since, by definition, it has no input edges. Thus, a firing traversal need only visit sources once.

The acyclic property of an acyclic marked graph can be used to generate a minimum-scatter firing sequence between two markings with an executable firing-count vector. If a source is located, it can be fired to its firing number and removed from the graph. This source is guaranteed in acyclic graphs. After removing this source from the graph, the resulting subgraph must also have the acyclic property and the procedure can then be recursively applied until all vertices have been removed. This is just a *topological sort* [69] applied to the vertices of the acyclic graph. Let $v_1, v_2, \cdots, v_n$ be the labels assigned to the vertices of an acyclic marked graph G according to a topological sort, then

$$v_1^{\sigma_1} v_2^{\sigma_2} \cdots v_n^{\sigma_n} \qquad (2.6)$$

is a zero-scatter firing sequence for a given executable firing-count vector $\Sigma \triangleq [\sigma_1, \sigma_2, \cdots, \sigma_n]^t$. Thus, we have the following theorem.

Theorem 2.4: A zero-scatter firing sequence exists for any given executable firing-count vector on any acyclic marked graph.■

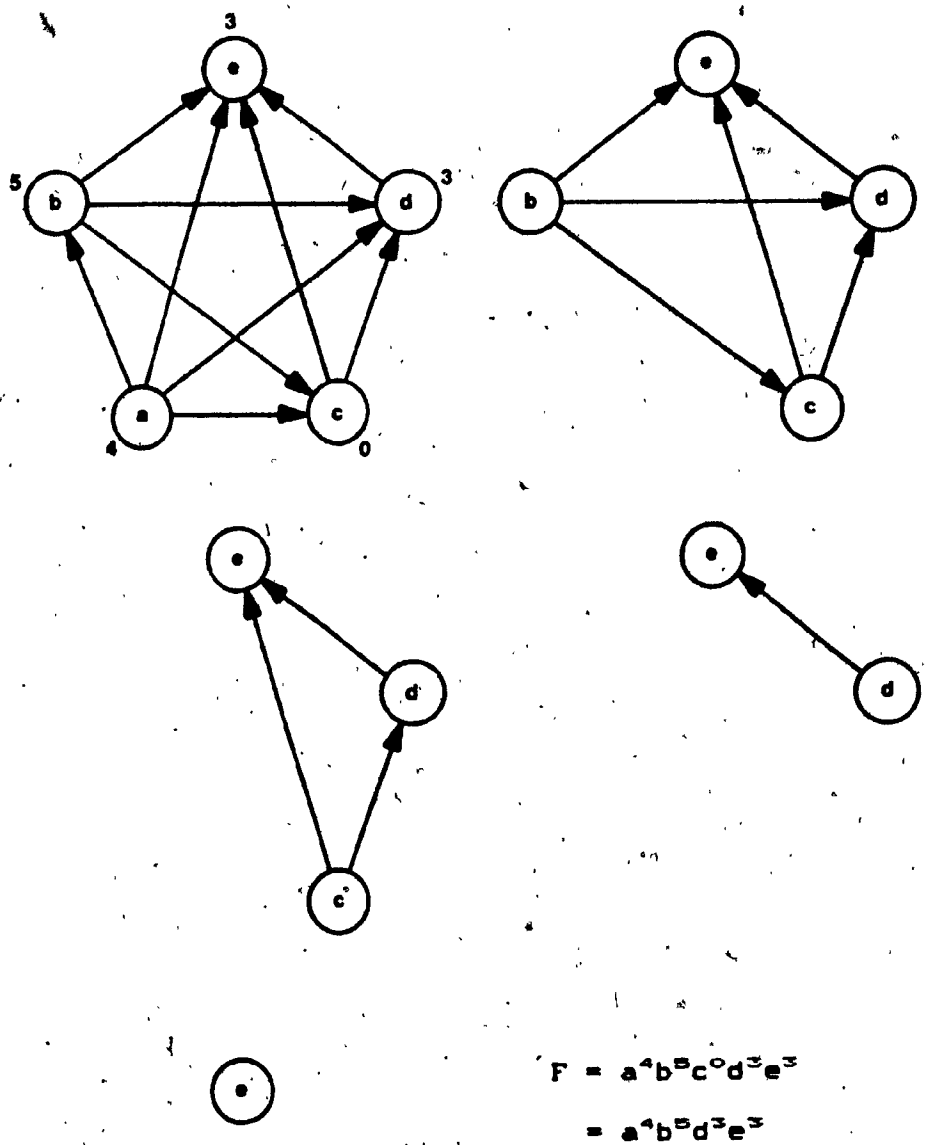
Figure 2.3 illustrates the procedure for an acyclic marked graph. The initial marking is irrelevant during topological sort. So, a minimum-scatter firing sequence can be obtained from the underlying graph and the firing-count vector.

2.4.2 The Marked Directed Circuit

We next consider the problem of generating a minimum-scatter firing sequence between two markings on a marked directed circuit with a legally executable, minimum firing-count vector. Let n be the number of vertices of a simple marked directed circuit C. Pick any vertex of C and label it v_0. Traverse the circuit from vertex v_0, in the circuit direction, labeling the vertices $v_1, v_2, \cdots, v_{n-1}$. Also, since the circuit has n edges, let e_j denote the edge incident into vertex v_j, for $j \in \{0,1,\cdots,n-1\}$. Let M_0 and M_f be the initial and final markings of C, respectively, and let Σ_0 be the corresponding minimum firing-count vector. Since Σ_0 is assumed to be executable, it follows from Equation 2.1 that

$$\sum_{j=0}^{n-1} M_0(e_j) = \sum_{j=0}^{n-1} M_f(e_j) = \tau_C, \qquad (2.7)$$

where τ_C is the *circuit token-count*. Hence, any two markings on C are mutually reachable if and only if they have the same circuit token count.



$$F = a^4b^5c^0d^3e^3$$

$$= a^4b^5d^3e^3$$

Figure 2.3

Obtaining a minimum-scatter sequence for an acyclic graph

Theorem 2.5: A zero-scatter firing sequence exists between any two reachable markings on a simple marked directed circuit C.

Proof: Observation 1 in Section 2.2 guarantees the existence of at least one datum vertex. Without loss of generality, let v_o be a datum, and let $e_o = (v_{n-1}, v_o)$ and $e_1 = (v_o, v_1)$ be the two edges incident on v_o. If we remove v_o from the circuit C, the resulting subgraph is acyclic and Theorem 2.4 guarantees the existence of a zero-scatter sequence for the firing-count vector $[\sigma_1, \sigma_2, \cdots, \sigma_{n-1}]^t$. Clearly, this is also a zero-scatter firing sequence for the vector Σ_o, since $\sigma_o = 0$.■

Specifically, the firing sequence defined by

$$F \triangleq \prod_{j=1}^{n-1} v_{(i+j) \bmod (n)}^{\sigma_{(i+j) \bmod (n)}}, \tag{2.8}$$

where i is the index of any datum vertex of C, is always executable and has zero scatter. This sequence is obtained by starting at a datum vertex and sorting the circuit vertices in the order dictated by the circuit. Any vertices with zero firing count can be neglected. Figure 2.4 illustrates the construction of such a sequence.

2.4.3 Vertex-Disjoint Marked Directed Circuits

To extend the result for a simple marked directed circuit to the case in which all the directed circuits are

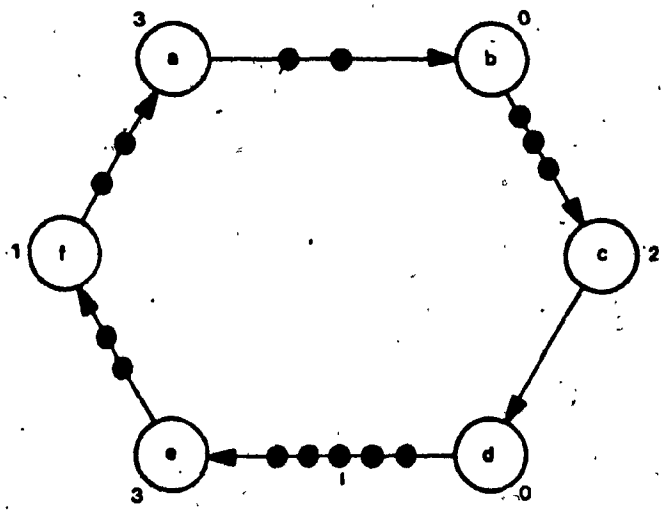


Figure 2.4(a)

The initial marking M_o

$F_1 = b^0c^2d^0e^3f^1a^3 = c^2e^3fa^3$

$F_2 = d^0e^3f^1a^3b^0c^2 = e^3fa^3c^2$

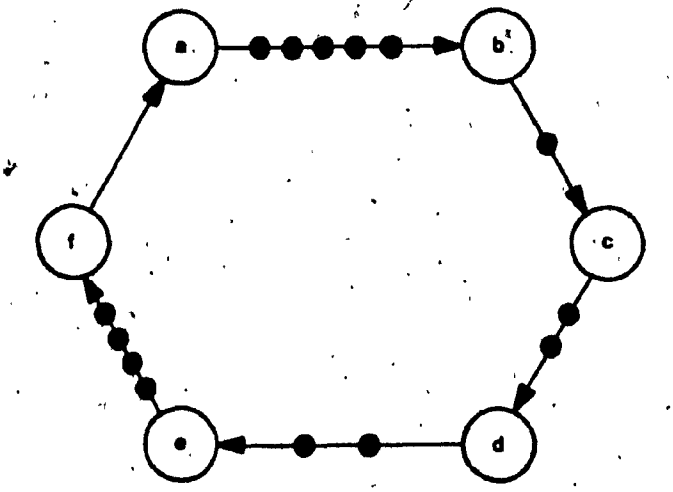


Figure 2.4(b)

The final marking M

vertex disjoint, we first note that the existence of a datum vertex in the circuit is not guaranteed if other vertices are present in the graph. Thus, it may be necessary to fire every vertex of a directed circuit in a marked graph. Since the vertices of a directed circuit C are restricted to at most τ_C firings per visit, scatter may occur. As an immediate consequence of Theorem 2.4, directed circuits are the only cause of scatter in a minimum-scatter firing sequence.

Assuming that the firings of the vertices of a marked directed circuit C, which is vertex disjoint with any other directed circuit in a marked graph G, are restricted only by the markings on the circuit edges, we can analyze C independently for a minimum-scatter subsequence. All vertices of G, except those of C, can be removed. The resulting subgraph is C with a corresponding set of firing numbers. One of two cases must occur. Either a vertex of C with a zero firing number exists or it does not. The former case reduces to the previous problem and zero scatter is possible in the subsequence for C by removing the vertex with a zero firing number and topologically sorting the resulting acyclic graph. In the latter case, a zero-scatter subsequence may still be possible but this is not generally true.

At this point,the question of scatter in a subsequence for the circuit can be answered by examining the disabling

numbers of the circuit vertices. If any disabling number is zero, then a zero-scatter subsequence is possible starting at that vertex. This follows from the fact that after this vertex is fired to its effective enabling number, its firing number becomes zero and it can be removed from C, resulting in the acyclic case discussed previously. Thus, we are left with the problem of finding a minimum-scatter firing sequence of C when the scatter is nonzero. We first solve this problem and then show how this solution can be used to construct minimum-scatter sequences in the case where all the directed circuits of G are vertex-disjoint.

In the following, whenever it is used as a vertex or edge index, the notation (i+j) will denote addition of the integers i and j, modulo n, where n is the number of vertices of C. Also, the notation (j) will represent j mod(n). With C labeled as in the previous subsection, we define a *cyclic* firing sequence of C, of length ℓ, as one with the form

$$\prod_{j=1}^{\ell} v_{(k+j-1)}^{\alpha_j}, \tag{2.9}$$

where $\alpha_j > 0$, $\forall\, j \in \{1,2,\cdots,\ell\}$. Expressing ℓ as $qn + r$, where the quotient $q \triangleq \lfloor \ell/n \rfloor$, and the remainder $r \triangleq (\ell)$, the cyclic sequence can be written as

$$v_k^{\alpha_1} v_{(k+1)}^{\alpha_2} \cdots v_{(k+n-1)}^{\alpha_n} v_k^{\alpha_{n+1}} \cdots v_{(k+n-1)}^{\alpha_{qn}} \cdots v_{(k+r-1)}^{\alpha_\ell}, \tag{2.10}$$

and has scatter $\ell - n$ if $\ell \geq n$.

Theorem 2.6: Every minimum nonzero-scatter firing sequence F_C of a disjoint directed circuit C in a marked graph G, is cyclic.

Proof: Consider an arbitrary minimum nonzero-scatter firing sequence F_C of the vertices $\{v_0, v_1, \cdots, v_{n-1}\}$ of a disjoint directed circuit C in a marked graph G. The vertices are assumed to be cyclically labeled. Let F_C be written as

$$F_C \triangleq \prod_{j=1}^{\ell} x_j^{\alpha_j}, \tag{2.11}$$

where x_j is the vertex visited at the j^{th} visit and α_j is the number of times x_j is fired at the j^{th} visit. Since F_C has nonzero scatter, let v_u be the first vertex of C which repeats in F_C. Let p and t denote the first and second visits of v_u in F_C, respectively. The following statements hold for F_C:

- $x_p, x_{p+1}, \cdots, x_{t-1}$ are distinct,
- $t - p \leq n$,
- $v_{(u-1)} \in \{x_j\}_{j=p+1}^{t-1}$.

The first statement follows from the hypothesis that v_u is the first vertex which repeats in F_C. The second statement follows from the first and the fact that there are only n vertices in C. The last statement must be true if F_C is a minimum-scatter firing sequence; otherwise, it is possible to construct another legal firing sequence F_C' by absorb-

ing the t^{th} visit into the p^{th} visit, resulting in a sequence which has less scatter than F_c.

Identify the maximal set of visits $\{t_u\}_{u=1}^{h}$ satisfying the following properties:

a: $p + 1 \leq t_u \leq t - 1$, $\forall\, u \in \{1,2,\cdots,h\}$,

b: $x_{t_u} = v_{(k-u)}$, $\forall\, u \in \{1,2,\cdots,h\}$,

c: $t_{u+1} < t_u$, $\forall\, u \in \{1,2,\cdots,h-1\}$.

Thus, F_c has the form

$$F_c = x_1^{\alpha_1}\cdots x_{p-1}^{\alpha_{p-1}} v_k^{\alpha_p}\cdots v_{(k-h)}^{\alpha_{t-h}}\cdots v_{(k-2)}^{\alpha_{t-2}}\cdots v_{(k-1)}^{\alpha_{t-1}}\cdots v_k^{\alpha_t} x_{t+1}^{\alpha_{t+1}}\cdots x_n^{\alpha_n}. \tag{2.12}$$

Two cases of interest are considered.

Case 1: $(k - h) \neq (k + 1)$.

In this case, α_t is independent of the firing of v_k at the p^{th} visit. Thus, we may construct another legal firing sequence F_c' by concatenating the visits $\{t_u\}_{u=1}^{h}$, in the order in which they appear in F_c, and placing them immediately before the p^{th} visit. After this rearrangement, the p^{th} and t^{th} visits may be grouped together as $v_k^{\alpha_p+\alpha_t}$ at the $(h+p)^{th}$ visit of F_c'. The constructed sequence F_c' will have the form

$$F_c' = x_1^{\alpha_1}\cdots x_{p-1}^{\alpha_{p-1}} v_{(k-h)}^{\alpha_{t-h}}\cdots v_{(k-2)}^{\alpha_{t-2}} v_{(k-1)}^{\alpha_{t-1}} v_k^{\alpha_p+\alpha_t}\cdots x_{t+1}^{\alpha_{t+1}}\cdots x_n^{\alpha_n}. \tag{2.13}$$

and is $\ell - 1$ visits long. This construction would reduce the scatter in F_c but since F_c has minimum scatter, this is not possible. In other words, Case 1 cannot occur.

Case 2: $(k - h) = (k + 1)$; i.e., $h = n - 1$.

In this case, the second statement holds with equality. That is, $t - p = n$ and so, $\{x_{t_u}\} = \{v_{(k+u)}\}_{u=1}^{n-1}$. Thus, the subsequence from the p^{th} to the t^{th} visit in F_c is a cyclic firing traversal of C with the form

$$v_k^{\alpha_p} v_{(k+1)}^{\alpha_{p+1}} v_{(k+2)}^{\alpha_{p+2}} \cdots v_{(k+n-1)}^{\alpha_{p+n-1}} v_k^{\alpha_{p+n}}. \qquad (2.14)$$

Now, we may assume that the firing of $v_{(k+i)}$ at the $(p+i)^{th}$ visit depends on the firing of $v_{(k+i-1)}$ at the $(p+i-1)^{th}$ visit and the subsequence from the p^{th} to the t^{th} visit in F_c cannot be reduced as in Case 1. Furthermore, since v_k is assumed to be the first vertex of C which repeats in F_c and all n vertices of C are present in the subsequence, it follows that $p = 1$. So, v_k must be the first vertex visited in the firing sequence F_c. Thus, $t = n + 1$ and F_c has the form

$$v_k^{\alpha_1} v_{(k+1)}^{\alpha_2} v_{(k+2)}^{\alpha_3} \cdots v_{(k+n-1)}^{\alpha_n} v_k^{\alpha_{n+1}} x_{n+2}^{\alpha_{n+2}} \cdots x_s^{\alpha_s}. \qquad (2.15)$$

Specifically, Case 2 must occur between nearest visits of any vertex which repeats in F_c. Otherwise, a reduction would be possible. Nearest visits of any vertex of C, in F_c, must necessarily be n visits apart. From this observation and the fact that the first n visits of F_c are cyclic, it follows that F_c must be cyclic.■

This result, along with the fact that we need only examine greedy firing sequences, allows us to immediately

conclude that we may obtain a minimum-scatter firing sequence F_C for a disjoint directed circuit C in a marked graph G by examining all the greedy cyclic firing sequences of C which execute a given firing-count vector Σ. There is exactly one such sequence starting at each legally firable vertex v_k of C at M_o. Since there can be at most n vertices of C which are legally firable under a marking of C, this problem is at most n times as complex as the problem of determining one cyclic firing sequence of C.

Specifically, the greedy cyclic firing sequence F_C, executing the firing-count vector $\Sigma = [\sigma_o, \sigma_1, \cdots, \sigma_{n-1}]^t$, from an initial marking M_o of C, is uniquely determined by its starting vertex v_k. Since we need only consider the case where σ_j is nonzero for each vertex $v_j \in C$, it follows that F_C is at least n visits long for each firable vertex $v_k \in C$ at M_o. The first n visits of F_C^k, the cyclic firing sequence starting at v_k, are

$$v_k^{\alpha_1} v_{(k+1)}^{\alpha_2} v_{(k+2)}^{\alpha_3} \cdots v_{(k+n-1)}^{\alpha_n}, \qquad (2.16)$$

where the α_k's are computed according to the recursive relation

$$\alpha_j \triangleq \min\{\sigma_{(k+j-1)}, M_o(e_{(k+j-1)}) + \alpha_{j-1}\}, \quad j = 2,3,\cdots,n,$$

where

$$\alpha_1 \triangleq M_o(e_k).$$

Here, we have assumed that the disabling number of vertex v_k is nonzero under the marking M_o. Otherwise, a zero-

scatter firing sequence exists starting at vertex v_k, given by the above definition, where α_j is replaced with $\sigma_{(k+j-1)}$ for all $j \in \{1,2,\cdots,n\}$. Let $\Sigma' = [\sigma'_0,\sigma'_1,\cdots,\sigma'_{n-1}]^t$ be the firing-count vector corresponding to the first n visits of F_C^k and let $\Sigma'' \triangleq [\sigma''_0,\sigma''_1,\cdots,\sigma''_{n-1}]^t = \Sigma - \Sigma'$ be the residual firing-count vector. Note that the firing-count vector Σ' is some cyclic permutation of $[\alpha_1,\alpha_2,\cdots,\alpha_n]^t$.

Theorem 2.7: If the firing numbers σ_i, $i = 0,1,\cdots,n-1$, of a marked directed circuit C are expressed modulo the circuit token count τ_C as $\sigma_i = q_i\tau_C + r_i$, where the quotient $q_i \triangleq \lfloor \sigma_i/\tau_C \rfloor$ and the remainder $r_i \triangleq \sigma_i \bmod(\tau_C)$, then

$$|q_i - q_j| \leq 1, \ \forall\ i, \ \forall\ j \in \{0,1,\cdots,n-1\}. \qquad (2.17)$$

Proof: Consider any two firing numbers σ_i,σ_j of a marked directed circuit C with a circuit token count τ_C. First, we make the following claim. Any two firing numbers σ_i,σ_j of a marked directed circuit C can differ by at most τ_C in every firing-count vector realizing every marking reachable from an initial marking of C. That is,

$$|\sigma_i - \sigma_j| \leq \tau_C, \ \forall\ i, \ \forall\ j \in \{0,1,\cdots,n-1\}. \qquad (2.18)$$

If v_i is adjacent to v_j, then the difference $\sigma_i - \sigma_j$ is plus or minus the change in the marking on the edge connecting them. This change can be at most τ_C in either direction. If v_i is not adjacent to v_j, then the difference $\sigma_i - \sigma_j$ can be expressed as the sum of the

differential markings along the directed path in the circuit from v_i to v_j. Again, this difference can be at most τ_c, thus establishing the claim.

Expressing the firing numbers σ_i, σ_j modulo τ_c,

$$\sigma_i = q_i\tau_c + r_i, \quad \sigma_j = q_j\tau_c + r_j. \tag{2.19}$$

If $q_i \neq q_j$, then, without loss of generality, let $q_i > q_j$. Then,

$$\sigma_i - \sigma_j = (q_i - q_j)\tau_c + r_i - r_j. \tag{2.20}$$

If, contrary to the statement of the theorem, $q_i > q_j + 1$, then $(q_i - q_j)\tau_c \geq 2\tau_c$. Clearly, we must have $|r_i - r_j| < \tau_c$ since $0 \leq r_i, r_j \leq \tau_c - 1$. So, it follows that

$$\sigma_i - \sigma_j = (q_i - q_j)\tau_c + r_i - r_j > \tau_c, \tag{2.21}$$

contradicting the claim established above.■

Theorem 2.8: The residual firing numbers have the property

$$\sigma''_k \geq \sigma''_{(k+1)} \geq \sigma''_{(k+2)} \geq \cdots \geq \sigma''_{(k+n-1)} \geq 0.$$

Proof: Assume, without loss of generality, that $k = 0$. Then, we must show that $\sigma''_0 \geq \sigma''_1 \geq \cdots \geq \sigma''_{n-1} \geq 0$. The firing numbers σ_j satisfy the system

$$\sigma_j - \sigma_{j+1} = M_-(e_{j+1}) - M_0(e_{j+1}), \quad j = 0,1,\cdots,n-1. \tag{2.22}$$

The first $n - 1$ of these equations form an independent set which can be written as

$$\sigma_{j+1} = \sigma_j + M_o(e_{j+1}) - M_-(e_{j+1}),\ j = 0,1,\cdots,n-2. \quad (2.23)$$

If this system of equations is solved in terms of σ_o, the firing numbers are

$$\sigma_k = \sigma_o + \sum_{j=1}^{k} M_o(e_j) - \sum_{j=1}^{k} M_-(e_j),\ k = 1,2,\cdots,n-1. \quad (2.24)$$

Recursively applying the relation defining the α_j's,

$$\alpha_k = \min\{\sigma_{k-1}, \sum_{j=0}^{k-1} M_o(e_j)\},\ k = 1,2,\cdots,n. \quad (2.25)$$

Since $\sigma'_k = \alpha_{k+1}$, for $k = 0,1,\cdots,n-1$, we have

$$\sigma''_k = \sigma_k - \min\{\sigma_k, \sum_{j=0}^{k} M_o(e_j)\},\ k = 0,1,\cdots,n-1 \quad (2.26)$$

or,

$$\sigma''_k = \max\{\sigma_k - \sum_{j=0}^{k} M_o(e_j), 0\},\ k = 0,1,\cdots,n-1. \quad (2.27)$$

Substituting the solution for σ_k in (2.24) into (2.27) gives

$$\sigma''_k = \max\{\sigma_o - M_o(e_o) - \sum_{j=1}^{k} M_-(e_j), 0\},\ k = 1,2,\cdots,n-1 \quad (2.28)$$

Since the markings are nonnegative vectors, it follows that the numbers defined by

$$\beta_o \triangleq \sigma_o - M_o(e_o),$$

$$\beta_k \triangleq \beta_o - \sum_{j=1}^{k} M_-(e_j),\ k = 1,2,\cdots,n-1, \quad (2.29)$$

have the property $\beta_k \geq \beta_{k+1}$ for $k = 0,1,\cdots,n-2$. Thus, the residual firing numbers $\sigma''_k = \max\{\beta_k,0\}$, for $k = 0,1,\cdots,n-1$, must have the property

$$\sigma''_0 \geq \sigma''_1 \geq \sigma''_2 \geq \cdots \geq \sigma''_{n-1} \geq 0. \qquad (2.30)$$

Since this result is independent of the vertex labeling, we have

$$\sigma''_k \geq \sigma''_{(k+1)} \geq \sigma''_{(k+2)} \geq \cdots \geq \sigma''_{(k+n-1)} \geq 0, \quad (2.31)$$

for the residual firing numbers σ''_i corresponding to the firing sequence F^k_C.■

To completely characterize the greedy cyclic firing sequence F^k_C, two cases need to be considered.

Case 1: $\sigma''_{(k+n-1)} = 0$.

Here, vertex $v_{(k+n-1)}$ may be removed from C and Theorem 2.4 guarantees the existence of a zero-scatter firing sequence executing Σ'' on the resulting acyclic subgraph. The number of visits needed to execute Σ'' is just the number of nonzero entries in Σ''.

Case 2: $\sigma''_{(k+n-1)} \neq 0$.

In this case, all residual firing numbers are nonzero and the greedy cyclic firings of the vertices of C have placed all τ_C tokens on edge e_k. Since any vertex of C can be fired at most τ_C times per visit, each vertex v_i must be visited at least q_i times in order to execute σ''_i, where q_i

$\triangleq \lfloor \sigma''_1/\tau_c \rfloor$. Since $q_{(k+n-1)}$ is a minimum in $\{q_i\}$, then the entire circuit must be traversed at least $q_{(k+n-1)}$ times to execute Σ''_k. Each traversal will have the form

$$v_k^{\tau_c} v_{(k+1)}^{\tau_c} v_{(k+2)}^{\tau_c} \cdots v_{(k+n-1)}^{\tau_c}, \tag{2.32}$$

and the firing count Σ''' remaining after $q_{(k+n-1)}$ such traversals, is obtained by subtracting $q_{(k+n-1)}\tau_c$ from each entry in Σ''. If $q_{(k+n-1)}$ is zero, then $\Sigma''' = \Sigma''$ and such a traversal is not possible. Clearly, the residual firing numbers Σ''' satisfy the property of Theorem 2.8. Therefore, $\sigma'''_k = (q_k - q_{(k+n-1)})\tau_c + r_k$ is a maximum σ'''_i. From Theorem 2.7 and Theorem 2.8, we have $q_k - q_{(k+n-1)} \leq 1$ and, therefore, vertex v_k need not be visited more than twice to execute σ'''_k. At this point, the number of visits needed to execute Σ''' may be counted. Specifically, two visits are required for each σ'''_i greater than τ_c and one visit is required for each nonzero σ'''_i less than or equal to τ_c. Thus, the number of visits needed to execute Σ''' is the number of nonzero entries in Σ''' plus the number of entries in Σ''' greater than τ_c. Let w be the number of entries in Σ''' greater than τ_c, where $0 \leq w \leq n-1$.

In either case, the firing sequence F_c^k may be formally written as

$$F_c^k \triangleq F_1 F_2 F_3 \tag{2.33}$$

where

$$F_1 \triangleq v_k^{\sigma'_k} v_{(k+1)}^{\sigma'_{(k+1)}} v_{(k+2)}^{\sigma'_{(k+2)}} \cdots v_{(k+n-1)}^{\sigma'_{(k+n-1)}},$$

$$F_2 \triangleq \left[v_k^{\tau_c} v_{(k+1)}^{\tau_c} v_{(k+2)}^{\tau_c} \cdots v_{(k+n-1)}^{\tau_c} \right]^{q_{(k+n-1)}},$$

$$F_3 \triangleq v_k^{\tau_C} \cdots v_{(k+w-1)}^{\tau_C} v_{(k+w)}^{r_{(k+w)}} v_{(k+w+1)}^{r_{(k+w+1)}} \cdots v_{(k+w+n)}^{r_{(k+w+n)}}, \quad (2.34)$$

and the visits of the form $v_j^{\tau_C}$ are not present in F_3, if $w = 0$. In equation (2.34), the symbol $[]^x$ refers to the operation of repeating the sequence [] x times.

The problem remains to determine k.

The above results are summarized in the following theorem, where

$len(F_C^k)$ = *length* of F_C^k in visits,

$s_{min} \triangleq \min_i \{\lfloor \sigma_i'' / \tau_C \rfloor\}$,

$n_1 \triangleq$ number of nonzero entries in Σ'',

$n_2 \triangleq$ number of nonzero entries in Σ''',

$n_3 \triangleq$ number of nonzero entries greater than τ_C in Σ'''.

<u>Theorem 2.9</u>:

i) If $\sigma''_{(k+n-1)} = 0$, then $len(F_C^k) = n + n_1$.

ii) If $\sigma''_{(k+n-1)} > 0$, then $len(F_C^k) = (s_{min}+1)n + n_2 + n_3$. ■

The following algorithm determines a vertex v_k which should be fired first to arrive at a legal minimum-scatter firing sequence F_C^k for a disjoint directed circuit C (one in which vertex firings are restricted only by the tokens on its edges) in a marked graph G, given an executable firing-count vector, from an initial marking M_o of G. Correctness of the algorithm follows from Theorem 2.9. We assume that the disabling numbers and, hence, the firing

numbers are all nonzero. Let Σ be the firing count vector for the circuit vertices. Let C be labeled as in the previous subsection.

Algorithm 2.1:

Step 1: For each legally firable vertex v_i of C at M_o, obtain $\Sigma^{i''}$, as follows:

$$\sigma'^{i}_{(i+j) \bmod (n)} \triangleq \min \left\{ \sum_{k=0}^{j} M_o(e_{(i+k) \bmod (n)}), \sigma_{(i+j) \bmod (n)} \right\},$$

$$\sigma''^{i}_{(i+j) \bmod (n)} \triangleq \sigma_{(i+j) \bmod (n)} - \sigma'^{i}_{(i+j) \bmod (n)}, \qquad (2.35)$$

for $j = 0,1,\cdots,n-1$. This step is of complexity $O(n^2)$.

Step 2: For each of the $\Sigma^{i''}$ vectors generated in Step 1, the residual firing number $\sigma''^{i}_{(i-1) \bmod (n)}$ is a minimum one and σ''^{i}_{i} is a maximum one. Now, let $q_{min} = \lfloor \sigma_{min}/\tau_c \rfloor$, where the residual count $\sigma_{min} = \min_i \{\sigma''^{i}_{(i-1) \bmod (n)}\}$. Obtain all $\Sigma^{i'''}$ vectors by reducing all firing numbers σ''^{i}_{j} in each $\Sigma^{i''}$ by $q_{min}\tau_c$. If any of the $\Sigma^{i'''}$ vectors contain zero entries, then proceed to Step 3. Otherwise, proceed to Step 4.

Step 3: From among all the $\Sigma^{i'''}$ vectors, pick any one of them having a maximum number of zero entries. Let this vector be $\Sigma^{k'''}$. Stop.

Step 4: From among all the $\Sigma^{i'''}$ vectors pick any one of them having a maximum number of entries less than or equal to τ_c. Let this vector be $\Sigma^{k'''}$. Stop.

At termination of this algorithm, the vertex v_u is the

first vertex in the minimum-scatter firing sequence of C as defined before. The minimum-scatter firing sequence F_c can be determined from the vectors $\Sigma^{k''}$, $\Sigma^{k'''}$ and the number q_{min}, or by starting at vertex v_u and traversing the circuit in its direction, firing each vertex to its effective enabling number, and updating the effective enabling numbers, until all effective enabling numbers for the circuit are zero. It is easy to see that each step in the above algorithm is of complexity $O(n^2)$, and the overall complexity of this algorithm is $O(n^2)$.

Figure 2.5 is used to illustrate the steps involved in the above algorithm. It is assumed that this directed circuit C is a subgraph of some marked graph G, which is vertex disjoint with all other directed circuits in G. The minimum firing-count vector $\Sigma = [\sigma_a, \sigma_b, \sigma_c, \sigma_d, \sigma_e, \sigma_f]^t = [23,24,24,21,22,22]^t$. There are four legally firable vertices of C at M_o. Step 1 produces the following:

$$\Sigma^{a'} = [2,3,3,6,7,7]^t, \quad \Sigma^{a''} = [21,21,21,15,15,15]^t,$$
$$\Sigma^{b'} = [7,1,1,4,5,5]^t, \quad \Sigma^{b''} = [16,23,23,17,17,17]^t,$$
$$\Sigma^{d'} = [6,7,7,3,4,4]^t, \quad \Sigma^{d''} = [17,17,17,18,18,18]^t,$$
$$\Sigma^{e'} = [3,4,4,7,1,1]^t, \quad \Sigma^{e''} = [20,20,20,14,21,21]^t.$$

The minimum residual firing number is 14. Since $\tau_c = 7$, then $q_{min} = 2$. Thus, step 2 yields the vectors:

$$\Sigma^{a'''} = [7,7,7,1,1,1]^t,$$
$$\Sigma^{b'''} = [2,9,9,3,3,3]^t,$$

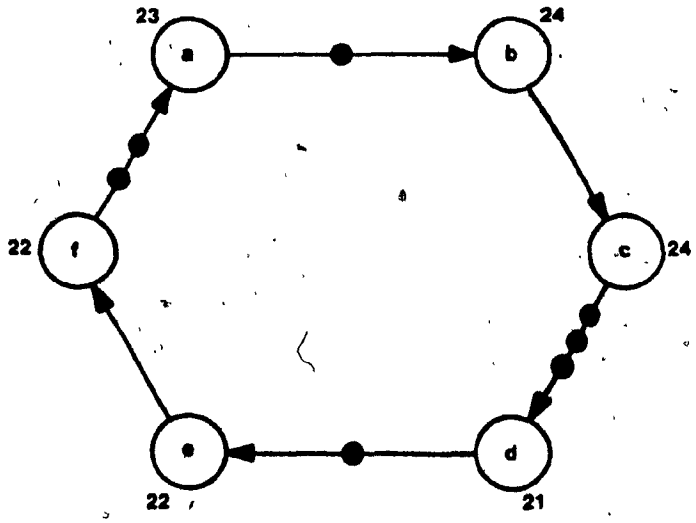


Figure 2.5(a)

The initial marking M_o

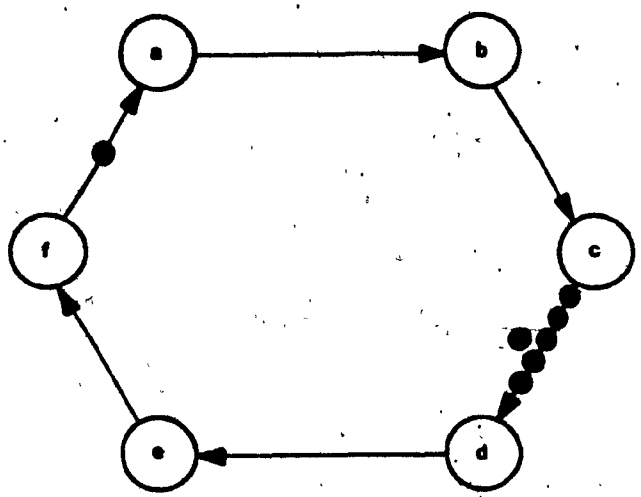


Figure 2.5(b)

The final marking M_{15a}

$$\Sigma^{d''} = [3,3,3,4,4,4]^t,$$
$$\Sigma^{e''} = [6,6,6,0,7,7]^t.$$

Since $\Sigma^{e''}$ is the only residual vector with a zero entry, the greedy cyclic firing sequence executing Σ, starting at vertex e, is a minimum-scatter firing sequence leading from M_0 to M_{134}. From Theorem 2.9, the length of this sequence is 23 and the scatter is 17. The minimum firing sequence is

$$F_c = efa^3b^4c^4d^7[e^7f^7a^7b^7c^7d^7]^2e^7f^7a^4b^4c^4. \qquad (2.36)$$

The above algorithm could be used to find a minimum-scatter legal firing sequence for a marked graph G, in which all directed circuits are vertex-disjoint, as follows. First, identify all the directed circuits and condense them [69]. The resulting subgraph G' is acyclic. Let $v_1, v_2, \cdots, v_n$ be the labels assigned to the vertices of G' according to a topological sort. Then, the sequence

$$F_a = F_1F_2F_3\cdots F_n, \qquad (2.37)$$

is a minimum-scatter firing sequence for G, where F_i is the minimum-scatter subsequence, determined as before, if the vertex v_i corresponds to a condensed directed circuit. Otherwise, $F_i \triangleq v_i^{\sigma_i}$. Implementing the above procedure involves the following steps:

- Performing a depth-first search to determine all the disjoint circuits. The complexity of this is O(m + n) [69], where m is the number of edges and n is the number of

vertices in the graph;

• Condensing the directed circuits and performing a topological sort on the resulting condensed graph. This can also be achieved by means of a depth-first search;

• Applying Algorithm 1 to each directed circuit. This is of complexity $O(n^2)$;

• The overall complexity of implementing the above procedure is $O(n^2)$.

2.5 On Minimum-Scatter Firing Sequences for a General Graph

We now outline an approach for determining minimum-scatter firing sequences in the case of any general graph and then show that the complexity of this problem reduces to that of the corresponding problem for the case of a strongly-connected graph. Consider a marked directed graph $G = (V,E)$, where V is the vertex set and E is the edge set. Let $G_1 = (V_1,E_1)$, $G_2 = (V_2,E_2)$, $\cdots$, $G_k = (V_k,E_k)$ be the strongly-connected components of G. Then, it is known that the sets $V_1V_2,\cdots,V_k$ form a partition of V [69]. Let G' be the graph obtained by contracting (short-circuiting) all the edges in each of the strongly-connected components and removing the resulting self loops. Then, it is known that G' has no directed circuits. Let the vertex set of G' be denoted by $\{x_1,x_2,\cdots,x_k\}$. Assume that x_i corresponds to G_i and without loss of generality, let a topological sort of G' assign number i to vertex x_i. Then, it is easy to see that the sequence

$$F = F_1F_2\cdots F_k \tag{2.38}$$

is a minimum-scatter firing sequence of G, where F_i is a minimum-scatter firing sequence of G_i. This follows from the fact that after executing $F_1F_2\cdots F_{i-1}$, the firings of the vertices of G_i are restricted only by the tokens of the edges E_i of G_i. Note that the problem considered in Section 2.4.3, concerning vertex-disjoint marked directed circuits, is, in fact, the special case where each G_i is a directed circuit. The problem of determining a minimum-scatter firing sequence for a general graph involves the following steps:

- determine all strongly-connected components of G [69];
- obtain the condensed graph G';
- perform a topological sort of the vertices of G';
- find a minimum-scatter firing sequence for each strongly-connected component of G, and then
- a minimum-scatter firing sequence F, of G is given by

$$F = F_1F_2\cdots F_k, \tag{2.39}$$

where F_i is the minimum-scatter subsequence of the strongly-connected component corresponding to the i^{th} vertex in the topologically-sorted vertex set of G'.

We now establish an upperbound on the enabling number μ_i of any vertex v_i under any marking reachable from a live initial marking M_o on a strongly-connected graph G. In a strongly-connected directed graph G, each edge e_i belongs

to at least one directed circuit. Since the circuit token count of all directed circuits of G must remain invariant under any legal sequence of vertex firings, the number of tokens on any edge e_i of a directed circuit can be no more than that circuit's token count. Extending this restriction to all directed circuits containing edge e_i implies that the token count on edge e_i cannot exceed the minimum circuit token count of all directed circuits containing edge e_i. Let v_i be the terminal vertex of edge e_i. Now, every circuit containing edge e_i also contains vertex v_i. Thus, the enabling number μ_i of vertex v_i cannot exceed the minimum circuit token count of all directed circuits containing edge e_i. Applying this argument to all edges incident into vertex v_i implies that the enabling number μ_i of any vertex v_i in a strongly-connected graph G cannot exceed the minimum circuit token count of all directed circuits containing vertex v_i. So,

$$\mu_i \leq \min_j\{\tau_{ij}\}, \quad \forall\, i \in \{1,2,\cdots,n\}, \tag{2.40}$$

where τ_{ij} is the token count of the j^{th} directed circuit passing through vertex i.

We may use this result to determine a lowerbound on the scatter of any firing sequence of a marked graph G. Since it is shown that only the strongly-connected components of G need be considered, we present the result for a strongly-connected graph. Let τ_i denote the bound on the

enabling number μ_i of vertex v_i, as defined above. Clearly, vertex v_i can be fired at most τ_i times per visit. Thus, vertex v_i must be visited at least $\lceil \sigma_i/\tau_i \rceil$ times to execute σ_i. If n is the number of vertices with a nonzero firing count σ_i in a strongly-connected graph G then for any legal firing sequence F executing Σ, we have

$$\text{scatter}(F) \geq \sum_{i=1}^{n} \lceil \sigma_i/\tau_i \rceil - n. \qquad (2.41)$$

2.6 Structure of the Reachability Problem for (0,1)-Capacitated Marked Graphs

Many classical results in graph theory are equivalent to the maximum-flow minimum-cut theorem of network flow theory [69]. These results include Tutte's characterization of maximum matchings in general graphs, Hall's theorem on bipartite matching and Menger's theorems on connectivity. Equivalence among these problems is established by constructing appropriate (0,1)-*communication networks* which permit flows of values of only 0 or 1 on each of its edges. This equivalence has made possible the design of efficient algorithms for matching and connectivity analysis because efficient algorithms are available for computing maximum flows in (0,1)-communication networks. These pioneering works have provided the motivation for the study presented in this section.

A marked graph is called a (0,1)-*capacitated marked graph* if under any marking on the graph, the number of

tokens allowed on each edge of the graph is equal to 0 or equal to 1. In this section, we study the structure of the reachability problem on (0,1)-capacitated marked graphs and derive a purely graph-theoretic characterization of this problem.

Consider a (0,1)-capacitated marked graph $G = (V,E)$ with a specified marking M_a. We define a *transformed graph* $G_a = (V_a,E_a)$ as follows:

i) G and G_a have the same underlying undirected graph; and

ii) If we denote by e_a the edge of G_a corresponding to the edge e in G, then the orientation of e_a is opposite to that of e if $M_a(e) = 1$; otherwise e_a and e have the same orientation.

As an example, a (0,1)-capacitated marked graph G with two markings M_a and M_b and the corresponding transformed graphs G_a and G_b are shown in Figures 2.6(a),(b),(c),(d) and (e), respectively.

Lemma 2.1: A (0,1)-capacitated marked graph G has no dead-locks under marking M_a if and only if the transformed graph is acyclic.

Proof: A dead-subgraph in G is a circuit C which is of one of the following forms:

i) C is a token-free directed circuit of G;

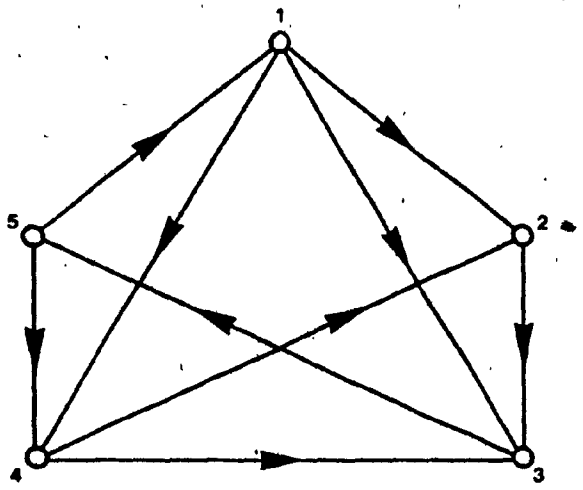


Figure 2.6(a)

Graph G

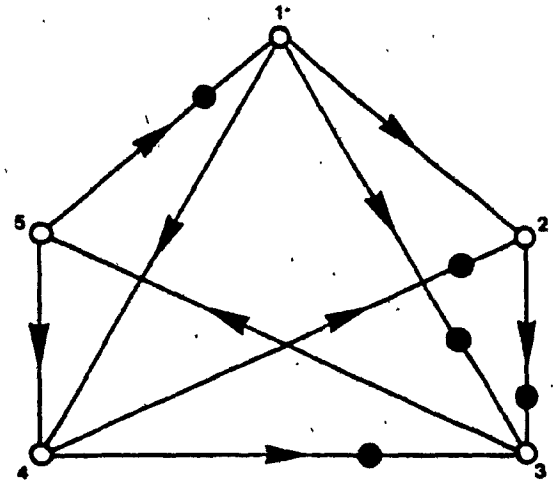


Figure 2.6(b)

Marking M_a

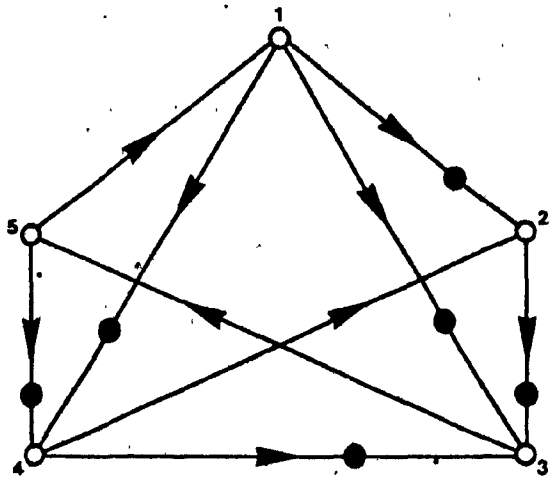


Figure 2.6(c)

Marking M_b

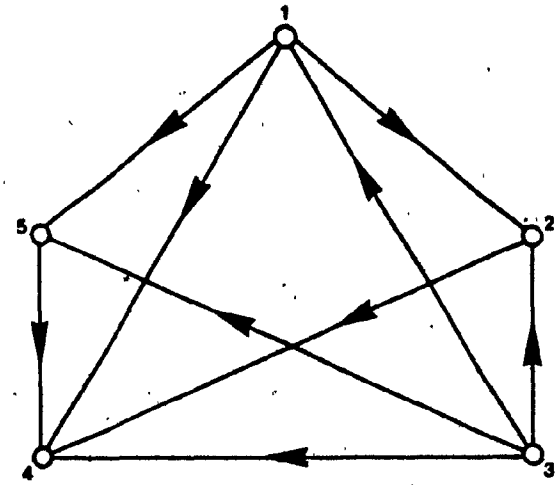


Figure 2.6(d)

Transformed graph G_a

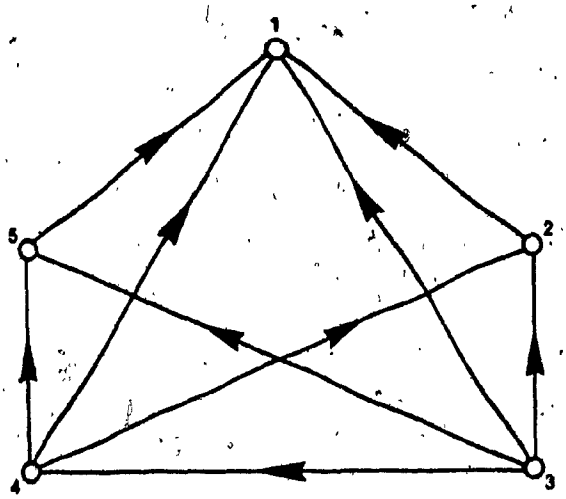


Figure 2.6(e)

Transformed graph G_b

ii) C is a directed circuit of G in which all of the edges are saturated (that is, the number of tokens on each edge of C is unity);

iii) C is a circuit of G in which all the edges in one direction are token-free and those in the opposite direction are saturated.

If we denote by C_a the circuit in G_a corresponding to a circuit C in G, then it is easy to see from the definition of G_a that C_a is a directed circuit if and only if C is of one of the three forms mentioned above. Thus, G_a is acyclic if and only if G has no dead-subgraphs.■

Theorem 2.10: A (0,1)-capacitated marked graph G is live if and only if the transformed graph G_a is acyclic.

Proof: Proof follows from Lemma 2.1 and the fact that G is live if and only if it has no dead-subgraphs(Theorem 1.1).■

Consider any two markings M_a and M_b on a (0,1)-capacitated marked graph G. Let G_a and G_b denote the corresponding transformed graphs, respectively. As before, the edges in G_a and G_b which correspond to the edge e in G will be denoted by e_a and e_b, respectively. Similar notation will be used to denote the corresponding circuits. Let

$$REV(M_a,M_b) = \{e_a | e_a \text{ and } e_b \text{ have opposing orientations}\}.$$

For example, with respect to the markings M_a and M_b of Figure 2.6(b) and 2.6(c)

$$REV(M_a,M_b) = \{(1,2),(1,4),(1,5),(2,4),(5,4)\}.$$

Consider a circuit C in G and define an arbitrary circuit orientation on C. The same orientation will be assigned to the corresponding circuits C_a and C_b. Let

$C_a^+ = \{e_a | e_a \in REV(M_a,M_b), e_a$ follows the orientation of $C_a\}$

$C_a^- = \{e_a | e_a \in REV(M_a,M_b), e_a$ opposes the orientation of $C_a\}$.

In the following, the phrase "contribution of $\Delta_M(e) = M_b(e) - M_a(e)$" will refer to the quantity Δ_M if e has the same orientation as that of C; otherwise, it will refer to the quantity $-\Delta_M(e)$.

Theorem 2.11:

i) For any edge $e \in C$, the contribution of $\Delta_M(e)$ to C is positive if and only if $e_a \in C_a^+$;

ii) For any edge $e \in C$, the contribution of $\Delta_M(e)$ to C is negative if and only if $e_a \in C_a^-$.

Proof:

Proof of i):

Necessity: Consider any edge $e \in C$ such that the contribution of $\Delta_M(e)$ to C is positive. Then, either one of the following two cases should occur:

i) $M_a(e) = 0$, $M_b(e) = 1$ and e has the same orientation as that of C;

ii) $M_a(e) = 1$, $M_b(e) = 0$ and e has an orientation opposite to that of C.

In either of these two cases, $e_a \in C^+$.

Sufficiency: Consider any edge e such that $e_a \in C^+$. Then, either of the following two cases should occur:

i) $M_a(e) = 0$, $M_b(e) = 1$ and e has the same orientation as that of the circuit C;

ii) $M_a(e) = 1$, $M_b(e) = 0$ and the orientation of e is opposite to that of C.

In either of these two cases, the contribution of $\Delta_M(e)$ to C is positive.

Proof of statement ii) follows along the same lines as above.■

Theorem 2.12: Consider a (0,1)-capacitated marked graph G. Let M_a and M_b be two live markings on G. M_b is reachable from M_a if and only if, for every circuit C in G

$$|C_a^+| = |C_a^-|.$$

Proof: By Theorem 1.5, M_b is reachable from M_a if and only if, the differential markings $\Delta_M(e)$'s satisfy KVL in G. By Theorem 2.11, the differential markings satisfy KVL in G if and only if, for each circuit C in G

$$|C_a^+| = |C_a^-|.$$■

Note that a vertex v is enabled in G under the marking M_a if and only if v is a source in G_a. Furthermore, firing v in G corresponds to reversing the orientations of all the edges incident on v in G_a. Thus, we may define *firing a source* in G_a as the operation of reversing the orientations of all the edges incident on that source. In view of these observations, we can conclude from Theorem 2.12 that the acyclic graph G_a can be transformed into the acyclic graph G_b through a sequence of source firings if and only if the condition of this theorem is satisfied.

An easy consequence of Theorem 2.12 is stated next.

Corollary 2.12.1: If M_b is reachable from M_a on G, then $REV(M_a,M_b)$ is a cut in G_a.

Proof: First note that $REV(M_a,M_b)$ has an even number of common edges with every circuit in G_a. It is well known [69] that such a subset of edges is a cut in G_a.■

At this point, we wish to draw attention to a related work on the concept of similarity introduced and studied in [66]. Consider a directed graph G. The operation of reversing the orientations of all the edges incident on a vertex v in G is called *switching* the vertex v in G [66]. Note that, whereas firing is done only at a source vertex, switching is permitted at any vertex. Two directed graphs G_a and G_b having the same underlying undirected graph are *similar* if and only if one can be transformed into the

other through a sequence of switchings. It has been proved in [66], that G_a and G_b are similar if and only if $REV(M_a,M_b)$ is a cut in G_a. Now, note from Corollary 2.12.1 that this necessary and sufficient condition for similarity is only a necessary condition for reachability of M_b from M_a. As in [66], we can also design, using depth-first-search, algorithms to test the reachability of M_b from M_a and construct a sequence of source firings leading G_a to G_b.

Finally, we wish to point out that Gafni and Bertsekas [67] have introduced destination oriented acyclic directed graphs in their study of a routing problem in computer communication networks. An acyclic directed graph (in short, ADG) with a special vertex (called the destination) is *destination oriented* if for every vertex there exists a directed path originating at this vertex and terminting at the destination. The problem considered in [17] is the following. Given a connected destination *disoriented* ADG, transform it to a destination oriented ADG by reversing the orientations of some of its edges. Gafni and Bertsekas have developed distributed algorithms for this problem. The fact that this problem is closely related to the problem considered in this section suggests the possibility of designing efficient distributed algorithms for the reachability problem on (0,1)-capacitated marked graphs.

2.7 Summary

In this chapter, we first gave an algorithmic proof to the reachability theorem on marked graphs. We have also extended this proof to cover the capacitated case.

We then introduced the concept of scatter in a firing sequence. Using the notion of a greedy firing policy, we have presented algorithms for generating minimum-scatter firing sequences for different classes of graphs. More specifically, we have considered three cases: acyclic directed graphs, directed circuits and graphs in which all directed circuits are vertex-disjoint. We have pointed out that in the general case, this problem reduces to that of determining a minimum-scatter firing sequence for a strongly-connected graph.

Finally, we have presented a purely graph-theoretic characterization of the reachability problem on (0,1)-capacitated marked graphs. The relationship between this work and the results presented in [66], [67] have been pointed out. This relationship suggests the possibility of designing efficient distributed algorithms for the reachability problem on (0,1)-capacitated marked graphs.

Chapter 3

MAXIMUM-WEIGHT MARKINGS IN MARKED GRAPHS: ALGORITHMS AND INTERPRETATIONS BASED ON THE SIMPLEX METHOD

In this chapter, we define and study the maximum-weight marking problem on a marked graph. Our study is based on a linear-programming formulation of the problem. We develop an algorithmic solution to the problem based on the simplex method of linear programming. As we develop our main algorithm, we offer interpretations, in terms of marked graph operations, to those which one encounters in the theory of linear programming. Our study covers live as well as nonlive capacitated marked graphs.

3.1 The Maximum-Weight Marking Problem

Given a marked graph $G = (V,E)$ with an initial marking M_o, we consider the problem of obtaining an $M \in R(M_o)$ which is maximum or minimum in some sense. An obvious objective is simply $\sum_e M(e)$, the token count of G under M. In fact, a solution to this problem for live, strongly-connected marked graphs can be found in [4], where the authors employ a circulation-flow approach for solving its dual.

In what follows, we generalize this problem by introducing a per token weight $W(e)$ associated with each edge $e \in E$ which represents the weight or cost of one token

residing there. Then, the product W(e)M(e) is the weight of M(e) tokens residing on edge e and the quantity $\sum_e W(e)M(e)$ is the weight of M on G. Thus, we focus on the maximum-weight marking problem

$$\underset{M \in R(M_o)}{\text{maxmize}} \; WM \qquad (3.1)$$

where W is the row vector of edge weights. The minimization problem is equivalent to (3.1) with -W replacing W. In the following section, we then go one step further and consider a capacitated extension of this problem in which an upperbound is specified on the token count on each edge. We would like to point out that the problem of determining a maximum-weight marking in a marked graph is equivalent to the problem of determining a maximum marking in a computation graph when the input quantum, the output quantum and the threshold of each edge of the computation graph are all equal to unity [3].

Our discussion begins with a linear-programming formulation of the maximum-weight marking problem based on the reachability theorem, namely Theorem 1.3. Let T be a spanning tree of G and let $\bar{T}$ be the corresponding cospanning tree of G. Let B_f be the fundamental-circuit matrix of G with respect to the tree T. Let $Z_{\bar{T}}$ be the vector $B_f M_o$. The reachability theorem provides a circuit-theoretic characterization of the reachability set of M_o on G. If we relax or neglect the dead-subgraph condition of the theorem

by considering live marked graphs only, then clearly, the maximum-weight marking problem is equivalent to the linear program

$$\begin{aligned} &\text{maximize } WM \\ &\text{subject to } B_f M = Z_\tau, \\ &\qquad M \geq 0. \end{aligned} \tag{3.2}$$

It is not obvious how to incorporate the dead-subgraph condition into this linear-program format since the dead-subgraph condition involves the firing counts of the vertices of G and Program 3.2 is stated in M only. For this reason, we will focus first on the live class of problems characterized by (3.2). We consider in Section 3.2, an alternative formulation of Problem 3.1 in terms of the vertex firing numbers which is equivalent to (3.2) for the live class of problems and which captures the nature of the nonlive case presented in Sections 3.3 and 3.4.

3.1.1 Basic Markings

Central to the methods of linear programming, namely the simplex method and its variants, is the concept of a basic solution to a consistent, underspecified system of independent, linear equations. For such systems, it is always possible to express a subset of the variables, called *basic variables*, explicitly in terms of the remaining *nonbasic variables*, implying the existence of many solutions to the system by virtue of the fact that we may freely specify values for the nonbasic variables and com-

pute the associated basic variables. Only such systems render the optimization problem nontrivial. If we specify zero values for the nonbasic variables then we obtain a ***basic solution*** - one in which only basic variables may have nonzero values. The fundamental theorem of linear programming [70] states that if a linear program has an optimal solution then it has a basic optimal solution. This can be proved rigourously using the properties of convex polyhedra, however, as Chvátal points out in [70], the fundamental theorem of linear programming follows easily after proper development of the simplex method. Since our intent is to apply the simplex method to Program 3.2, the fundamental theorem of linear programming becomes our springboard into the analysis.

Any assignment of the variables in a linear program which satisfies all of its constraints constitutes a ***feasible solution***. Besides the circuit equations, Program 3.2 has the nonnegativity constraint $M \geq 0$. Thus, any nonnegative solution to $B_f M = Z_T$ is a feasible solution of (3.2). The feasible solutions of (3.2) are in one to one correspondence with the elements of $R(M_o)$.

The simplex method examines only ***basic feasible solutions*** during its search of an optimal one. Hence, we must define a basic marking. In order to do so, we must know what constitutes a set of basic or nonbasic variables. This becomes obvious once we examine the fundamental cir-

cuit matrix B_f. The canonical or echelon form of a fundamental-circuit matrix implies that the cospanning tree variables can be expressed in terms of the spanning tree variables and hence, the cospanning tree variables become the basic variables and the tree variables become the nonbasic variables. Thus, we have the following definition.

A marking M of G is called a *basic marking* if there exists a token-free spanning tree of G under M.

In order to apply the simplex method, we must obtain a basic feasible solution of (3.2). From the marked graph point of view, it is not even clear that there exists a basic marking $M \in R(M_o)$. Indeed, as can be easily seen, there may be no basic marking reachable from M_o, if M_o is not live on G. However, as we will prove, it is always possible to obtain a basic $M \in R(M_o)$ for the live class of problems.

Generally, in a linear-programming problem, we are faced with the subproblem of obtaining a basic feasible solution in order to start the simplex method. It should be obvious that since $M_o \in R(M_o)$, then M_o is a feasible solution of (3.2). If M_o is also basic then we can start simplex with M_o. Otherwise, it seems reasonable that knowledge of a feasible solution of (3.2) should help us find a basic feasible solution of (3.2). Indeed, this is the case as we will demonstrate after introducing the notion of a

diakoptic transition. Let us assume for now that M_o is basic with token-free spanning tree T.

3.1.2 Tokens and Flows

In the following, we illustrate the structure of the maximum-marking problem and its relation to the transshipment problem of network flow theory.

The vector $Z_{\bar{T}}$ is an *invariant* of the equivalence class $R(M_o)$ with respect to $\bar{T}$, for obvious reasons. Every $M \in R(M_o)$ satisfies $B_f M = Z_{\bar{T}}$. The k^{th} component of $Z_{\bar{T}}$ is an invariant of the k^{th} fundamental circuit. It is simply the algebraic sum of the tokens in the k^{th} fundamental circuit when traversed in the direction of its defining chord. This number must be the same for every $M \in R(M_o)$.

Lemma 3.1: $Z_{\bar{T}} \geq 0$ when M_o is basic with tree T.

Proof: When M_o is basic with spanning tree T, all tokens reside on the chords of the cospanning tree $\bar{T}$. Since each chord defines exactly one fundamental circuit and each fundamental circuit contains only one chord, then the invariant of each fundamental circuit is just the token count on the defining chord which is nonnegative.■

Before illustrating the structure of this problem, we introduce one more definition. The *cost* or *weight* of a fundamental cutset is defined as the algebraic sum of its edge weights with respect to the forward orientation of the

cutset. In other words, a cutset's weight is the sum of its forward edge weights minus the sum of its backward edge weights. Note that this definition is dual to that of a circuit invariant. If $K = K_+ \cup K_-$ is a cutset with forward edge set K_+ and backward edge set K_-, then we shall denote the weight of K under the weighting W of E as

$$W(K) = \sum_{e \in K_+} W(e) - \sum_{e \in K_-} W(e). \qquad (3.3)$$

Before formulating this problem's dual, let us eliminate the chord (basic) variables from the objective to obtain an equivalent objective in terms of the branch (nonbasic) variables only. To do this, we need the *basic dictionary* [70], which is readily available from the canonical form of the constraints. The partition is as follows

$$\text{maximize } J = [W_{\bar{T}}, W_T]\begin{bmatrix} M_{\bar{T}} \\ M_T \end{bmatrix} = W_{\bar{T}}M_{\bar{T}} + W_T M_T$$

$$\text{subject to } [I_{\bar{T}} \; B_{\bar{T}T}]\begin{bmatrix} M_{\bar{T}} \\ M_T \end{bmatrix} = M_{\bar{T}} + B_{\bar{T}T}M_T = Z_{\bar{T}}, \; M \geq 0, \qquad (3.4)$$

where the subscript $\bar{T}$ denotes the cospanning tree, the subscript T denotes the spanning tree and $I_{\bar{T}}$ is a unit or identity matrix of dimension $|\bar{T}|$. The basic dictionary is simply

$$M_{\bar{T}} = Z_{\bar{T}} - B_{\bar{T}T}M_T. \qquad (3.5)$$

Replacing the cospanning tree marking $M_{\bar{T}}$ with its dictionary, in the objective J, gives

$$J = W_{\bar{T}}(Z_{\bar{T}} - B_{\bar{T},T}M_T) + W_T M_T,$$
$$= W_{\bar{T}}Z_{\bar{T}} + (W_T - W_{\bar{T}}B_{\bar{T},T})M_T,$$
$$= W_{\bar{T}}Z_{\bar{T}} + \tilde{W}_T M_T. \qquad (3.6)$$

The number $W_{\bar{T}}Z_{\bar{T}}$ is the value of the original objective J under the marking M_o since the tree variables are all zero. Thus, an equivalent objective for our optimization problem is

$$\tilde{J} \triangleq \tilde{W}M, \qquad (3.7)$$

where

$$\tilde{W} \triangleq [\tilde{W}_{\bar{T}}, \tilde{W}_T],$$
$$\tilde{W}_{\bar{T}} \triangleq 0, \text{ and}$$
$$\tilde{W}_T \triangleq W_T - W_{\bar{T}}B_{\bar{T},T}.$$

We see from the expressions for J and $\tilde{J}$ that we can increase their values by increasing, by an appropriate amount, the token count on any tree edge with a corresponding positive coefficient in $\tilde{W}_T$. The elements of $\tilde{W}$ are the familiar *relative-cost coefficients* from the simplex method. It is easily seen that the objective J or $\tilde{J}$ cannot be increased by increasing values of tree variables if $\tilde{W}_T \leq 0$ and hence, M is optimal if and only if $\tilde{W}_T \leq 0$.

Theorem 3.1: The relative branch costs are the associated fundamental-cutset weights.

Proof: The relative-cost vector of the tree is

$$\tilde{W}_T = W_T - W_{\bar{T}}B_{\bar{T},T},$$

Transposing,

$$\begin{aligned}\tilde{W}_T^t &= W_T^t - B_{f\,T}^t W_f^t, \\ &= I_T W_T^t - B_{f\,T}^t W_f^t, \\ &= [-B_{f\,T}^t \;\; I_T] \begin{bmatrix} W_f^t \\ W_T^t \end{bmatrix}, \\ &= Q_f W^t. \blacksquare\end{aligned}$$

Transposing again,

$$\tilde{W}_T = W Q_f^t,$$

where Q_f is the fundamentel-cutset matrix of G with respect to the tree T.■

This result enables us to translate the algebraic criterion for optimality, $\tilde{W}_T \leq 0$, into a structural criterion for optimality, namely, all fundamental cutsets of T have nonpositive weight, and exploit this structural property of an optimal solution in an efficient graph algorithm. First, let us obtain the dual of this problem.

We may state the primal as

$$\begin{aligned}&\text{maximize } \tilde{W}M, \\ &\quad \text{subject to } B_f M = Z_T, \; M \geq 0,\end{aligned} \tag{3.8}$$

and the dual [70] as

$$\begin{aligned}&\text{minimize } F_T Z_T, \\ &\quad \text{subject to } F_T B_f \geq \tilde{W}.\end{aligned} \tag{3.9}$$

That this is a minimum-cost flow problem is not obvious in this form. However, when we partition the problem

canonically, we obtain

$$\begin{aligned} &\text{minimize } F_{\bar{T}}Z_{\bar{T}}, \\ &\quad \text{subject to } F_{\bar{T}}[I_{\bar{T}} \; B_{\bar{T}T}] \geq [\tilde{W}_{\bar{T}}, \tilde{W}_T] = [0, \tilde{W}_T]. \end{aligned} \tag{3.10}$$

This decomposes to

$$\begin{aligned} &\text{minimize } F_{\bar{T}}Z_{\bar{T}}, \\ &\quad \text{subject to } F_{\bar{T}}B_{\bar{T}T} \geq \tilde{W}_T, \; F_{\bar{T}} \geq 0. \end{aligned} \tag{3.11}$$

Adding *slack* variables (one for each branch of T), we obtain

$$\begin{aligned} &\text{minimize } F_{\bar{T}}Z_{\bar{T}}, \\ &\quad \text{subject to } F_{\bar{T}}B_{\bar{T}T} - F_T = \tilde{W}_T, \; F_{\bar{T}}, F_T \geq 0. \end{aligned} \tag{3.12}$$

Let $F = [F_{\bar{T}}, F_T]$ and $Z = [Z_{\bar{T}}^t, Z_T^t]^t = [Z_{\bar{T}}^t, 0^t]^t$, then the dual program is

$$\begin{aligned} &\text{minimize } FZ, \\ &\quad \text{subject to } FQ_f^t = -\tilde{W}_T, \; F \geq 0. \end{aligned} \tag{3.13}$$

This is a minimum-cost flow problem since the constraints are seen to be flow-conservation equations (current equations) of G. The vector $-\tilde{W}_T$ is a supply vector or, equivalently, $\tilde{W}_T$ is a demand vector whose components correspond to commodity demand for a node or group of nodes defined by the corresponding branch in T. This problem is well known in network optimization theory and it can be solved efficiently using the network simplex method [70] which exploits the structure inherent in the flow problem. We could, therefore, specify a dual procedure for solving the maximum-weight marking problem by a direct analogy with

the network simplex algorithm [70]. However, attacking the maximum-marking problem from scratch leads us to interesting interpretations, concepts and results.

3.1.3 Diakoptic Transitions

For expositional convenience, we shall extend the concept of enablement to a subgraph of G and thus introduce the notion of a *diakoptic* transition in a marked graph.

Let S and $\bar{S} = V - S$ be a partition of V and let $\langle S,\bar{S}\rangle$ denote the *cut* $\langle S,\bar{S}\rangle_+ \cup \langle S,\bar{S}\rangle_-$ consisting of the forward cut edges $\langle S,\bar{S}\rangle_+$ directed from S to $\bar{S}$ and the backward cut edges $\langle S,\bar{S}\rangle_-$ directed from $\bar{S}$ to S. Let $G(S)$ be the subgraph induced on S by removing $\langle S,\bar{S}\rangle$ from G. Similarly, let $G(\bar{S})$ be the subgraph induced on $\bar{S}$ by removing $\langle S,\bar{S}\rangle$ from G. If $G(S)$ and $G(\bar{S})$ are both connected, then $\langle S,\bar{S}\rangle$ is called a *cutset* of G. Let us assume $\langle S,\bar{S}\rangle$ is an arbitrary cut of G.

We define the *enabling numbers* of $G(S)$ and $G(\bar{S})$ as

$$\mu(G(S)) \triangleq \min_{e\in\langle S,\bar{S}\rangle_-} \{M(e)\} \tag{3.14}$$

and

$$\mu(G(\bar{S})) \triangleq \min_{e\in\langle S,\bar{S}\rangle_+} \{M(e)\}. \tag{3.15}$$

An *elementary diakoptic firing* of a vertex-induced subgraph $G(\bullet)$ of a marked graph G is any legal firing sequence confined to vertices in $G(\bullet)$ which fires each

vertex in G(•) exactly once. Note that this definition includes subgraphs G(•) consisting of disjoint components.

Theorem 3.2 (Diakoptic-Transition Theorem): An elementary diakoptic firing of a vertex-induced subgraph G(S) of a marked graph G is legal under a live marking M if and only if $\mu(G(S)) > 0$.

Proof: First, we note that the markings of G(S) and $G(\bar{S})$ are unaffected by the diakoptic firing of G(S) since vertices in $\bar{S}$ are not fired and each vertex in S is fired exactly once. The only edges of G whose markings change in an elementary diakoptic firing of G(S) are edges of $\langle S,\bar{S}\rangle$. Each edge of $\langle S,\bar{S}\rangle_-$ loses one token and each edge of $\langle S,\bar{S}\rangle_+$ gains one token. By hypothesis, $\mu(G(S)) > 0$ and so, each edge $e \in \langle S,\bar{S}\rangle_-$ has at least one token. Further, the edges of $\langle S,\bar{S}\rangle_+$ play no role in determining the legality of a firing sequence confined to vertices in S. Hence, all edges in the cut $\langle S,\bar{S}\rangle$ may be removed from G, isolating G(S) from $G(\bar{S})$ and so, we need only show that there exists a legal firing sequence of any marked graph G from a live marking M which fires each vertex of G exactly once, returning to M. This question has already been resolved in [4]. However, we present an alternate proof which is easily extendable for the capacitated case.

Let $G(E_\tau)$ be the token-free subgraph of G under M induced by the token-free edge set $E_\tau = \{e|M(e) = 0\}$.

Property 1: $G(E_f)$ is acyclic.

Property 2: A source in $G(E_f)$ is an enabled vertex in G under M.

Property 1 follows from the liveness of M. That is, G has no token-free directed circuits under M. Property 2 follows from the observation that a source of $G(E_f)$ is either a source of G or a vertex of G with at least one token on each edge incident into it under M. In either case, a source of $G(E_f)$ is an enabled vertex of G under M. Since $G(E_f)$ is acyclic, it contains a source and hence, G contains an enabled vertex v under M. Firing this vertex v results in a new marking of G obtained by subtracting one token from each edge incident into v and adding one token to each edge incident out of v. Since v has been fired and each edge incident out of v has at least one token then v may be removed from G. The above argument recursively applies to the resulting subgraph of G since it is another live marked graph for which an elementary diakoptic-firing sequence is sought.■

As noted in the proof of Theorem 3.2, an elementary diakoptic firing of a subgraph G(S) affects the marking of G only on edges of the cut $\langle S,\bar{S}\rangle$. Specifically, one token is subtracted from all edges of $\langle S,\bar{S}\rangle$ incident into G(S) and one token is added to all edges of $\langle S,\bar{S}\rangle$ incident out of G(S). Thus, we may view the state-transition process diakoptically. That is, we may consider G(S) as a *super-*

node of G. This is the essence of the above theorem. It tells us that we can move from marking to marking by firing supernodes or clusters at a time, ignoring the actual firing sequence involved within the cluster since the theorem guarantees its existence.

3.1.4 A Diakoptic-Reachability Theorem

The diakoptic-transition theorem for live marked graphs provides us with a diakoptic-reachability theorem for such cases. The interesting aspect of this is that every reachable marking from an initial marking of a live marked graph G defines its own unique diakoptic firing sequence leading from the initial marking to that marking. We now proceed to develop this result.

Let $G = (V,E)$ be a marked graph on vertex set V, edge set E and live initial marking M_o. Let $M_r \in R(M_o)$ and, as before [5], let $\Sigma_o = [\sigma_1^o, \sigma_2^o, \cdots, \sigma_n^o]^t$ be the minimum nonnegative solution to $A^t\Sigma = M_r - M_o$, where A is the incidence matrix of G. Identify the entities defined by the following algorithm.

```
k ← 0
While Σ_k ≜ [σ_1^k, σ_2^k, ..., σ_n^k]^t ≠ 0 do
     Begin
          S_k ← {v | σ_v^k > 0}
          G_k ← G(S_k)
          x_k ← min {σ_v^k}
                v ∈ S_k
        σ_v^(k+1) ← { σ_v^k - x_k, ∀ v ∈ S_k
                    { 0,           ∀ v ∈ S̄_k
          k ← k + 1
     End
r ← k
```

Theorem 3.3 (Diakoptic-Reachability Theorem): The diakoptic-firing sequence defined by the expression

$$\prod_{k=0}^{r-1} G_k^{x_k}$$

of length $r \leq n-1$ legally transforms M_o into M_f on G, where $G_k^{x_k}$ denotes x_k successive elementary diakoptic firings of G_k.

Proof: First, let us recall that a diakoptic firing of a vertex induced subgraph $G(S)$ of a marked graph G affects the marking of G on edges of the cut $\langle S,\bar{S}\rangle$ only. Also, for a live G, a diakoptic firing of $G(S)$ is legal if and only if $\mu(G(S)) > 0$. Thus, we need only show that $x_k \leq \mu(G_k)$ for $0 \leq k \leq r-1$. We demonstrate that $x_o \leq \mu(G_o)$ and then deduce that $x_k \leq \mu(G_k)$ for $1 \leq k \leq r-1$.

Let $\bar{S}_k = V - S_k$, for $0 \leq k \leq r-1$. Further, as before, let $\langle S_k,\bar{S}_k\rangle_+$ and $\langle S_k,\bar{S}_k\rangle_-$ denote the forward and backward edge sets of the cut $\langle S_k,\bar{S}_k\rangle$, respectively, for $0 \leq k \leq r-1$. From the state equation, we have $M_f(e) = M_o(e) + \sigma_i^o - \sigma_j^o$, for every edge $e = (i,j) \in E$. Now, by definition, $\sigma_i^o = 0$, $\forall\, i \in \bar{S}_o$. So, $M_f(e) = M_o(e) - \sigma_j^o$, $\forall\, e = (i,j) \in \langle S_o,\bar{S}_o\rangle_-$. Imposing nonnegativity on M_f, we obtain $\sigma_j^o \leq M_o(e)$, $\forall\, e = (i,j) \in \langle S_o,\bar{S}_o\rangle$. By definition, $x_o \leq \sigma_j^o$, $\forall\, j \in S_o$, and, therefore, it follows that $x_o \leq M_o(e)$, $\forall\, e \in \langle S_o,\bar{S}_o\rangle_-$. The enabling number of G_o is simply

$$\mu(G_o) = \min_{e \in \langle S_o, \bar{S}_o \rangle_-} \{M_o(e)\}$$

and it follows that $x_o \leq \mu(G_o)$. To deduce that the $k+1^{th}$ diakoptic firing is legal given that the first k diakoptic firings are legal, we need merely note that Σ_k is the minimum nonnegative solution to $A^t\Sigma = M_- - M_k$, where M_k is the marking of G after the first k diakoptic firings. Thus, the above argument is true when S_o, x_o, M_o and Σ_o are replaced with S_k, x_k, M_k and Σ_k, respectively.■

3.1.5 Obtaining a Basic Marking in $R(M_o)$

With the notion of a diakoptic transition established, we describe an algorithm for obtaining a basic $M \in R(M_o)$ when M_o is live on G. We assume that G is connected.

Suppose that, by some means, we have obtained an $M \in R(M_o)$ such that the subgraph G(S), induced over some subset of vertices S, has a token-free spanning tree T under M. Since G is connected, $\langle S,\bar{S} \rangle \neq \emptyset$. Since G is live under M, G(S) is also live under M. If $\langle S,\bar{S} \rangle_- \neq \emptyset$ then fire G(S) diakoptically $\mu = \mu(G(S))$ times. This transition is legal and results in a marking M′ obtained from M by subtracting μ tokens from each edge in $\langle S,\bar{S} \rangle_-$ and adding μ tokens to each edge in $\langle S,\bar{S} \rangle_+$. If μ is not equal to zero, then at least one edge $e = (i,j) \in \langle S,\bar{S} \rangle_-$ is token-free under M′ and since the marking of G(S) is unaffected by the diakoptic firing of G(S), then T is also

token-free under M'. Clearly, this is the case when $\mu = 0$. Thus, the tree $T \cup \{e\}$ is a token-free spanning tree of $G(S \cup \{i\})$ under $M' \in R(M_o)$. If $\langle S,\bar{S}\rangle_- = \emptyset$ then $\langle S,\bar{S}\rangle_+ \neq \emptyset$ and so, by similar reasoning, we may fire $G(\bar{S})$, diakoptically, $\mu(G(\bar{S}))$ times. This transition is also legal and results in the marking M' obtained from M by subtracting $\mu(G(\bar{S}))$ tokens from each edge in $\langle S,\bar{S}\rangle_+$. At least one edge $e = (i,j) \in \langle S,\bar{S}\rangle_+$ is token-free under M' and so, $T \cup \{e\}$ is a token-free spanning-tree of $G(S \cup \{j\})$ under $M' \in R(M_o)$. Hence, we have an algorithm. We simply start with $S = \{v\}$ for any vertex $v \in G$ and with $T = \emptyset$.

The following algorithm will generate a basic marking M, reachable from a live marking M_o, for a connected marked graph G.

1. Set $T = \emptyset$, $M = M_o$ and $S = \{v\}$, for any $v \in V$.
2. While $|S| < |V|$ do
   If $\langle S,\bar{S}\rangle_- \neq \emptyset$
   Then
   Begin
   Compute $\mu \triangleq \mu(G(S))$ under M.
   Fire $G(S)$ μ times, updating M.
   $T \leftarrow T \cup \{e\}$ and $S \leftarrow S \cup \{i\}$,
   where $e = (i,j)$ is a token-free edge of $\langle S,\bar{S}\rangle_-$ under M.
   End
   Else
   Begin
   Compute $\mu \triangleq \mu(G(\bar{S}))$ under M.
   Fire $G(\bar{S})$ μ times, updating M.
   $T \leftarrow T \cup \{e\}$ and $S \leftarrow S \cup \{j\}$,
   where $e = (i,j)$ is a token-free edge of $\langle S,\bar{S}\rangle_+$ under M.
   End
3. Stop. M is a basic marking in $R(M_o)$ with the token-free spanning tree T.

We may modify this algorithm by including as many token-free edges as possible at each iteration, so long as a circuit does not form.

3.1.6 Pivoting and Diakoptic Firing

Next, let us interpret the fundamental operation of simplex, pivoting, in terms of vertex firing. This must be possible since simplex moves from marking to marking by pivoting and each such movement must be characterized by some legal vertex-firing sequence. Pivoting is quite easily interpreted in terms of a firing sequence, as the following definitions and discussion illustrate.

Each pivot of simplex selects one nonbasic variable (tree branch) with a corresponding positive relative cost (fundamental-cutset weight), and exchanges it with one basic variable (cospanning tree chord). If no such exchange is possible (all relative branch costs are nonpositive) then we have an optimal marking. Let us examine the details of this branch-chord exchange.

Let $b = (u,v)$ be the branch of T that has been selected to enter the basis (cospanning tree), where b is incident out of vertex u and into vertex v. Now, breaking the branch b splits T into exactly two fragments, T_u and T_v, where $u \in T_u$ and $v \in T_v$. Let S be the set of vertices that T_u spans and then T_v spans $\bar{S} = V - S$. The fundamental cutset of G that b defines is simply $(S,\bar{S})$. Thus,

since b has been selected to leave the tree and enter the cotree, then $W(\langle S,\bar{S}\rangle) > 0$ because the relative cost of branch b is the weight of $\langle S,\bar{S}\rangle$.

Let this first pivot move the state from M_0 to, say M_1. Simplex moves from one basic feasible solution to another basic feasible solution with each pivot. Thus, for M_1 to be basic, it must also possess a token-free spanning tree T_1. To ensure M_1 is a basic marking, we must select a chord from $\bar{T}$ so as to reconnect the fragments T_u and T_v. The only edges of G which connect vertices in T_u to vertices in T_v are edges of $\langle S,\bar{S}\rangle$. Hence, we must exchange b with one of the chords in its fundamental cutset. When the exchange is made, the cutset associated with the new tree branch is the same cutset associated with b in the original tree, but its orientation is now defined by the new branch. Now, we must determine a selection rule and the update procedure.

This is where the diakoptic property of the state-transition process is useful in explaining the pivot operation. Let $\mu = \mu(G(S))$. We know that G(S) can be legally, diakoptically fired μ times and this subtracts μ tokens from each edge of $\langle S,\bar{S}\rangle_-$ and adds μ tokens to each edge of $\langle S,\bar{S}\rangle_+$. This operation increases the objective function WM by an amount $\mu W(\langle S,\bar{S}\rangle)$. This is exactly the pivot operation of simplex. If $\mu = 0$ then we are experiencing degeneracy.

Now, at least one edge $d \in (S,\bar{S})$ is token-free under M_1. If there are more, then each token-free edge in $(S,\bar{S})$ is a candidate for leaving the basis. Let us disregard the possibility of more than one leaving candidate, for now, by assuming that d is unique in each pivot. However, we must consider degeneracy as this is an integer programming problem. Then, the basis exchange is denoted by the expressions $\bar{T}_1 = \bar{T} + \{b\} - \{d\}$ and $T_1 = T - \{b\} + \{d\}$.

Instead of computing the relative branch costs associated with the branches of T_1 from scratch, we would rather have a method that transforms the relative cost vector of T into the relative branch costs of T_1. This would be computationally efficient if the relative costs of T_1 can be obtained from those of T through a reasonably simple transformation. This looks like a fruitful approach since T and T_1 differ in only two edges. Indeed, this is the case, as the following theorem illustrates.

Since the relative branch costs are the corresponding fundamental-cutset weights, we must transform the fundamental-cutset weights of T into those of T_1. Now, $T + \{d\}$ contains exactly one circuit C, whose orientation is defined by the orientation of chord d. The branch b is a member of C and follows the circuit direction. A moment's reflection will ascertain that the only fundamental cutsets affected by the transformation

$$T_1 = T + \{d\} - \{b\}$$

are those associated with edges of C (think of the column associated with d in Q_t). Therefore, we need only update the relative costs associated with C_t in moving from T to T_1.

Theorem 3.4: The relative-cost vector $\tilde{W}_1$ of $T_1 = T - \{b\} + \{d\}$ is equal to the relative-cost vector $\tilde{W}$ of T minus $W(\langle S,\bar{S}\rangle)$ on all forward edges of C plus $W(\langle S,\bar{S}\rangle)$ on all backward edges of C.

Proof: The result is true for edges b and d since they are in the same direction with respect to C and the relative costs of both b and d decrease by $W(\langle S,\bar{S}\rangle)$ when they exchange roles. Let (x,y) be an edge of C other than b and d, let $\langle S_u,\bar{S}_u\rangle$ denote its fundamental cutset in T, and let $\langle S_u^1,\bar{S}_u^1\rangle$ denote its fundamental cutset in T_1. There are four cases to consider.

Case 1: $(x,y) \in T_v$ and follows the orientation of C.

In this case, $S \subset S_u$ and $S_u^1 = S_u - S$. Therefore, we may write the weight of the new cutset as

$$\begin{aligned} W(\langle S_u^1,\bar{S}_u^1\rangle) &= W(\langle S_u-S,(V-S_u)\cup S\rangle), \\ &= W(\langle S_u-S,V-S_u\rangle) + W(\langle S_u-S,S\rangle), \\ &= W(\langle S_u,V-S_u\rangle) - W(\langle S,V-S_u\rangle) - W(\langle S,S_u-S\rangle), \\ &= W(\langle S_u,\bar{S}_u\rangle) - W(\langle S,\bar{S}\rangle). \end{aligned}$$

Case 2: $(x,y) \in T_v$ and opposes the orientation of C.

In this case, $S_u \subset \bar{S}$ and $S_u^1 = S_u \cup S$. Clearly, this

will lead to

$$W((S_\alpha^1,\bar{S}_\alpha^1)) = W((S_\alpha,\bar{S}_\alpha)) + W((S,\bar{S})).$$

Case 3: $(x,y) \in T_\alpha$ and follows the orientation of C. Here, $S_\alpha \subset S$ and $S_\alpha^1 = S_\alpha \cup \bar{S}$, which will yield

$$\begin{aligned} W((S_\alpha^1,\bar{S}_\alpha^1)) &= W((S_\alpha,\bar{S}_\alpha)) + W((\bar{S},S)), \\ &= W((S_\alpha,\bar{S}_\alpha)) - W((S,\bar{S})). \end{aligned}$$

Case 4: $(x,y) \in T_\alpha$ and opposes the orientation of C. Now, we have $\bar{S} \subset S_\alpha$ and $S_\alpha^1 = S_\alpha - \bar{S}$, which leads to the expression

$$W((S_\alpha^1,\bar{S}_\alpha^1)) = W((S_\alpha,\bar{S}_\alpha)) + W((S,\bar{S})).\blacksquare$$

This theorem implies that we may treat the relative costs as currents or flows, because, as we move from tree to tree or more appropriately from one cospanning tree to another, the relative costs always satisfy the same nodal equations. Thus, we may now state the basic step of our algorithm. First, compute the n-1 fundamental cutset weights by inspection, and establish a current $I_o(b)$ on each branch b of the initial spanning tree $T_o = T$, equal to its corresponding fundamental-cutset weight $W((S,\bar{S}))$. Set $I_o(d) = 0$, for all chords of the initial cospanning tree $\bar{T}_o = \bar{T}$. We have avoided the problem of degeneracy so far, but we must specify an anticycling rule for completeness. Assuming degeneracy is not present, then the k^{th} basic step of the algorithm is

A: Locate a branch $b \in T_\mu$ with a positive current $I_\mu(b)$. If no such branch exists, then M_μ is optimal;

B: Compute $\mu = \mu(G(S))$ under M_μ;

C: Fire $G(S)$, μ times. That is, subtract μ tokens from all backward edges of $(S,\bar{S})$ and add μ tokens to all forward edges of $(S,\bar{S})$, to yield $M_{\mu+1}$;

D: Augment the current flowing in C. That is, subtract $I_\mu(b)$ flow units from all forward edges of C and add $I_\mu(b)$ flow units to all backward edges of C to yield $I_{\mu+1}$;

E: Let $\bar{T}_{\mu+1} = \bar{T}_\mu + \{b\} - \{d\}$, $T_{\mu+1} = T_\mu - \{b\} + \{d\}$.

As an example, to illustrate our maximum-weight marking algorithm, consider the marked graph with marking M_0 shown in Figure 3.1(a). In Figure 3.1(b) is shown a basic marking M_1 reachable from M_0. M_1 is obtained from M_0 through the sequence of diakoptic firings $G_1^0G_2^1G_3^0G_4^0G_5^1G_6^1G_7^0G_8^0$, where the subgraphs are induced on the vertex sets

$$S_1 = \{1\},$$

$$S_2 = \{1, 9\},$$

$$S_3 = \{1, 2, 9\},$$

$$S_4 = \{1, 2, 4, 9\},$$

$$S_5 = \{1, 2, 4, 5, 9\},$$

$$S_6 = \{1, 2, 4, 5, 8, 9\},$$

$$S_7 = \{1, 2, 4, 5, 7, 8, 9\},$$

$$S_8 = \{1, 2, 3, 4, 5, 7, 8, 9\},$$

respectively.

The relative costs associated with the branches defining M_1 are as shown in Figure 3.1(b). Note the relative

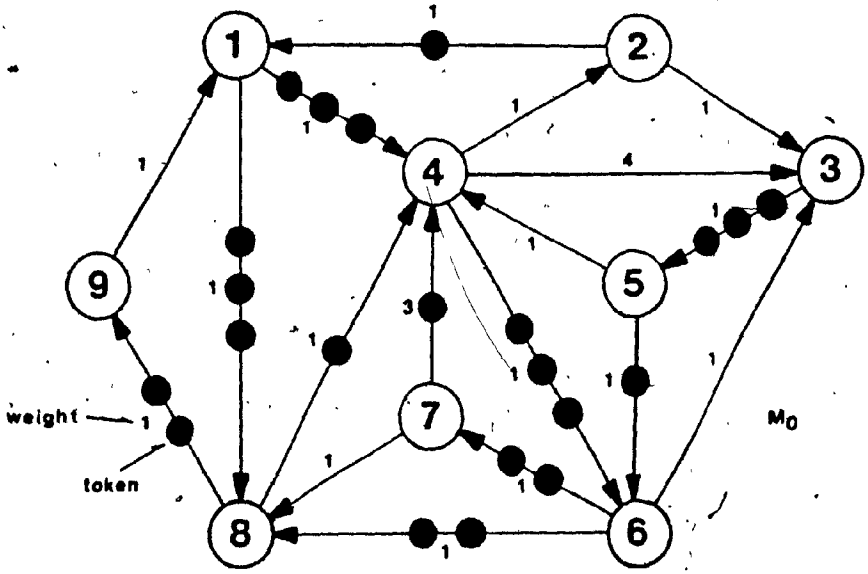


Figure 3.1(a)

Marking M_0

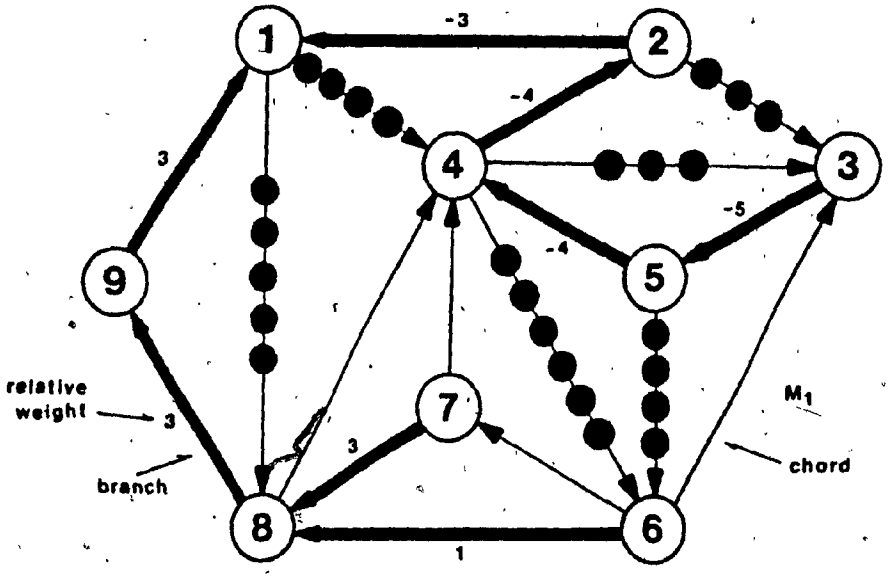


Figure 3.1(b)

A basic marking $M_1 \in R(M_0)$

costs for the chords are all zero.

It can be seen that the relative cost for the branch (8,9) is greater than zero. One chord (5,6) in the fundamental cutset of this branch has a minimum marking. Thus, we have to exchange (8,9) with (5,6). This requires firing the subgraph on the vertex set {6,7,8} 4 times, and we get the new basic marking M_2 shown in Figure 3.1(c). Updating the relative costs as described in Section 3.1.6, we get the relative costs of the branches defined by M_2 as shown in Figure 3.1(c).

Continuing the algorithm, we obtain the maximum-weight marking M_3 shown in Figure 3.1(d). This is achieved through the sequence of diakoptic firings $G^0(\{7\})G^4(\{1,2,3,4,5,6,7,9\})$. It may be noted from Figure 3.1(d) that all the relative costs are nonpositive indicating that M_3 is an optimum marking. The weight of this optimum marking is 50.

3.1.7 The Greedy Firing Sequence

Let S_k be the subset of V examined in the k^{th} step of our algorithm in Section 3.1.6. If the algorithm terminates in a basic maximum marking in r steps, for some instance of a maximum marking problem, then its execution specifies the greedy legal firing sequence F_r, defined by

$$F_r = \prod_{k=1}^{r} (G(S_k))^{\mu_k}, \qquad (3.16)$$

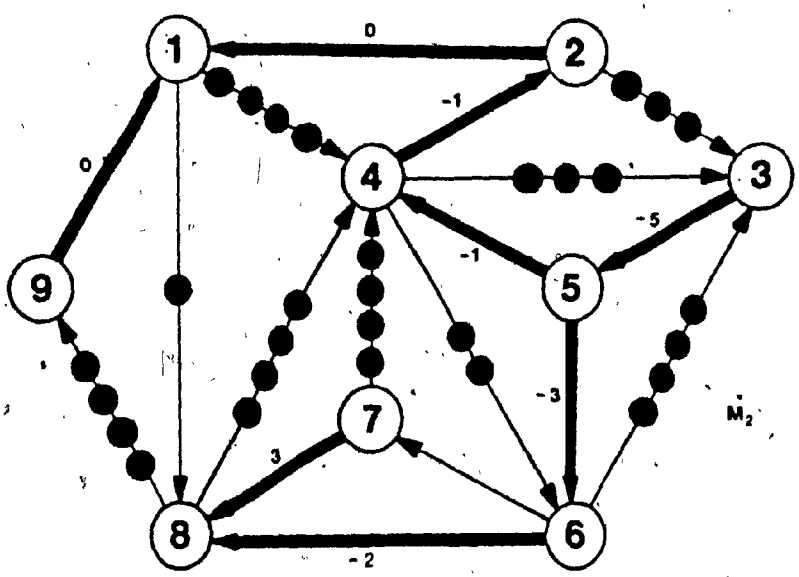


Figure 3.1(c)

A new basic marking $M_2 \in R(M_0)$

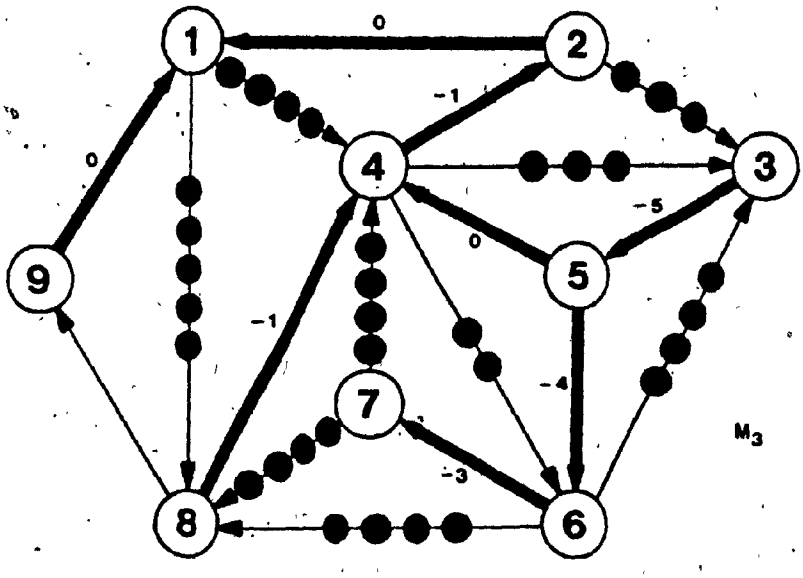


Figure 3.1(d)

A maximum-weight marking $M_3 \in R(M_0)$

where μ_k is the enabling number of $G(S_k)$ at the k^{th} step. The sequence F_r is considered greedy because simplex fires a subgraph as much as possible in each pivot. Simple examples illustrate that simplex does not necessarily execute a firing sequence with minimum firing count, in moving from M_o to a maximum marking $M \in R(M_o)$. Nor does it help in constructing a minimum-scatter firing sequence.

Each null firing of a subgraph $G(S_k)$ in the sequence F_r corresponds to a degenerate pivot. We define the progress sequence associated with F_r as

$$p(F_q) = \sum_{k=1}^{q} \mu_k W((S_k, \bar{S}_k)), \quad q = 1,2, \cdots, r. \qquad (3.17)$$

The number $p(F_q)$ represents the progress made in the subsequence F_q of F_r consisting of the first q firings in F_r. In the absence of degeneracy, the progress sequence increases monotonically with q and, in such a case, the algorithm must terminate in a maximum marking. However, in the presence of degeneracy, a bound on r may not exist; i.e., F may contain an infinite number of null firings representing the cycling phenomenon that is associated with this algorithm. We can prevent cycling by employing an anticycling rule such as Bland's least subscript rule [70]. However, there exists an even simpler anticycling rule due to Cunningham [70] for the dual algorithm. This leads us to conjecture the existence of an equally-elegant dual count-

erpart of Cunningham's anticycling rule for the algorithm we have described.

3.1.8 Boundedness

The conditions governing problem boundedness follow as an easy afterthought.

Suppose that simplex encounters a directed cutset $\langle S,\bar{S}\rangle$, defined by branch b of spanning tree T, at some pivot, for some instance of a maximum-weight marking problem. Then the enabling number of G(S) is undefined. G(S) is a diakoptic source. That is, G(S) can be diakoptically fired an indefinite number of times. Every diakoptic firing of G(S) increases the marking on each edge of $\langle S,\bar{S}\rangle$. Since b has been selected to enter the basis $W(\langle S,\bar{S}\rangle) > 0$. Thus, the problem is unbounded. Boundedness of the maximum-weight marking problem is characterized in the following theorem.

Theorem 3.5: The weight of a marking WM is bounded on $R(M_o)$ if and only if G does not contain a positive-weight directed cutset.

Proof: Necessity follows from the above argument. Sufficiency follows from the simplex method. If G does not contain a positive-weight directed cutset, then simplex cannot encounter one at any pivot. With a correct anticycling strategy, simplex must then locate a finite maximum

after a finite number of pivots since it examines a finite set.■

Note that the boundedness condition for the marking M of a marked graph, follows as a special case of Theorem 3.5 when W = [1,1,•••,1]. Thus, we have the following corollary.

Corollary 3.5.1: The marking of a marked graph G is bounded over $R(M_o)$ if and only if G is strongly connected.■

3.2 Maximum-Weight Marking for a Capacitated Marked Graph

The results of the previous sections are extended to cover capacitated marked graphs. The motivation for this stems from the queueing-network model. In any real system, the queues associated with the edges of the marked graph must have finite length and this can be taken into account using a capacitated marked graph.

Recall that a *capacitated marked graph* is a marked graph G = (V,E) in which for each edge $e \in E$, a lowerbound L(e) and an upperbound U(e) are specified on the token count M(e).

The introduction of lowerbounds puts the problem in its most general form and does not complicate matters significantly. We make the usual consistency assumption $L(e) \leq U(e), \forall e \in E$ which we denote in vector format as $L \leq U$. We make the further assumption $L < U$ since if

$L(e) = U(e)$ for some edge $e = (i,j) \in E$ then vertices i and j are dead in every marking of G. In other words, the vertices i and j are not enabled under any marking reachable from M_o.

As before, let us relax the dead-subgraph condition in the reachability theorem by considering live problems only. Then, for capacitated marked graphs, Program 3.2 is equivalent to the linear program

$$\begin{aligned} &\text{maximize } WM \\ &\quad \text{subject to } B_f M = Z_{\mp}, \\ &\qquad\qquad L \leq M \leq U. \end{aligned} \tag{3.18}$$

We need only outline the extensions from this point.

3.2.1 Basic Markings

A cospanning tree still constitutes a basis since the equation constraints are circuit equations of G. However, we must modify our definition of a basic marking. To this end, let $M \in R(M_o)$. Recall that an edge $e \in E$ is called *depleted* under the marking M if $M(e) = L(e)$. Similarly, an edge $e \in E$ is *saturated* under M if $M(e) = U(e)$. So, we define a basic marking in these terms:

A marking M of G is called a ***basic marking*** if there exists a spanning tree of G composed solely of branches that are either depleted or saturated under M.

3.2.2 Diakoptic Transitions

We must establish a diakoptic-transition theorem for the class of live problems on capacitated marked graphs. We recall that the enabling number of a vertex induced subgraph G(S) of G under M is defined as

$$\mu(G(S)) \triangleq \min\{ \min_{e\in(S,\bar{S})} \{M(e) - L(e)\}, \min_{e\in(S,\bar{S})} \{U(e) - M(e)\}\}, \quad (3.19)$$

Theorem 3.6: An elementary diakoptic firing of G(S) is legal if and only if $\mu(G(S)) > 0$.

Proof: Again, the proof follows from the reachability theorem for capacitated marked graphs (Theorem 1.5). However, we can provide a constructive procedure similar to that in the proof of Theorem 3.2. As in the proof of this theorem, we need only prove that there exists a legal firing sequence of any capacitated marked graph G from a live marking M which fires each vertex of G exactly once, returning to M.

Construct a graph G' from G by open-circuiting all edges of G which are neither depleted nor saturated under M and reversing the direction of all saturated edges under M.

Property 1: G' is acyclic.

Property 2: A source in G' is an enabled vertex in G under M.

As in the proof of Theorem 3.2, Property 1 follows

from the liveness assumption since, by definition, a dead-subgraph in G under M is represented by a directed circuit in G'. Property 2 follows from the observation that any source of G' is a vertex of G with no depleted input edges and no saturated output edges under M. Properties 1 and 2 imply that G has an enabled vertex v under M. After vertex v fires, it has no depleted output edges and no saturated input edges and hence, it may be removed from G since the edges incident on it do not restrict the remaining sequence. Again, as in the proof of Theorem 3.2, the above argument applies to the remaining subgraph and proof is established by recursion.■

3.2.3 Obtaining a Basic Marking in $R(M_o)$

We show, by construction, that it is possible to obtain a basic marking $M \in R(M_o)$, as defined for a capacitated marked graph, using the notion of a diakoptic transition.

If G is connected and U is finite then it is clear from the definition that the enabling number of any subgraph G(S) of G is always defined. However, to present the result in it most general form, we assume that U may have infinite entries.

To simplify the algorithm description, we define the input and output enabling numbers of a subgraph G(S) under M as

$$\mu_1(G(S)) \triangleq \begin{cases} \infty & \text{if } \langle S,\bar{S}\rangle_- = \emptyset \\ \min_{e\in\langle S,\bar{S}\rangle_-} \{M(e) - L(e)\} & \text{otherwise} \end{cases}$$

and

$$\mu_\infty(G(S)) \triangleq \begin{cases} \infty & \text{if } \langle S,\bar{S}\rangle_+ = \emptyset \\ \min_{e\in\langle S,\bar{S}\rangle_+} \{U(e) - M(e)\} & \text{otherwise,} \end{cases}$$

respectively. Then, $\mu(G(S)) \triangleq \min\{\mu_1(G(S)), \mu_\infty(G(S))\}$. Using these definitions, the following algorithm constructs a basic marking $M \in R(M_o)$, for a capacitated marked graph $G = (V,E)$.

1. Set $M = M_o$, $T = \emptyset$ and $S = \{v\}$ for any $v \in V$.
2. While $|S| < |V|$ do
   Begin
   $\mu_1 \leftarrow \mu_1(G(S))$, $\mu_\infty \leftarrow \mu_\infty(G(S))$ and $\mu \leftarrow \min\{\mu_1,\mu_\infty\}$ under M.
   If $\mu < \infty$
   Then
   Begin
   Fire $G(S)$ μ times, updating M.
   If $\mu_1 < \mu_\infty$
   Then $T \leftarrow T \cup \{e\}$ and $S \leftarrow S \cup \{i\}$, where $e = (i,j) \in \langle S,\bar{S}\rangle_-$ with $M(e) = L(e)$.
   Else $T \leftarrow T \cup \{e\}$ and $S \leftarrow S \cup \{j\}$, where $e = (i,j) \in \langle S,\bar{S}\rangle_+$ with $M(e) = U(e)$.
   End
   Else
   Begin
   $\mu \leftarrow \mu(G(\bar{S}))$ under M.
   Fire $G(\bar{S})$ μ times, updating M.
   $T \leftarrow T \cup \{e\}$ and $S \leftarrow S \cup \{j\}$, where $e = (i,j) \in \langle S,\bar{S}\rangle_+$ with $M(e) = L(e)$.
   End
   End
3. Stop. M is a basic marking in $R(M_o)$ with spanning tree T.

3.2.4 Conditions for Optimality

As usual, $\langle S,\bar{S}\rangle$ denotes the fundamental cutset de-

fined by branch b ∈ T. Optimality is characterized in the following definition.

A basic marking $M \in R(M_o)$ with basis $\bar{T}$ is maximum over $R(M_o)$ if and only if $W((S,\bar{S})) \geq 0$ for every saturated branch $b = (i,j) \in T$ and $W((S,\bar{S})) \leq 0$ for every depleted branch $b \in T$, where $i \in S$ and $j \in \bar{S}$.

To establish that this is indeed a sufficient condition for optimality of a basic marking $M \in R(M_o)$, we need only substitute the corresponding basic dictionary into the objective from which we conclude that the objective cannot be increased by effecting a change in the marking M(b) of any branch $b \in T$ and that the current value of the objective is an upperbound on WM over $R(M_o)$. Hence, such a basic marking is optimal.

3.2.5 Pivoting and Diakoptic Firing

We may restate the optimality condition as follows.

A basic marking $M \in R(M_o)$ is maximum over $R(M_o)$ if and only if T contains no depleted branch $b = (i,j)$ with $W((S,\bar{S})) > 0$ and no saturated branch $b = (i,j)$ with $W((S,\bar{S})) < 0$, where $i \in S$ and $j \in \bar{S}$.

Hence, we may use the method described in Section 3.1.6 to achieve an optimal basic marking with some slight modifications. From the above condition, a branch $b \in T$ is a candidate for entering the basis if it is depleted and

$W(\langle S,\bar{S}\rangle) > 0$ or if it is saturated and $W(\langle S,\bar{S}\rangle) < 0$.

Now, if the entering branch b is depleted, then $W(\langle S,\bar{S}\rangle) > 0$ indicates that firing $G(S)$ will increase the objective. Similarly, for a saturated branch b, $W(\langle S,\bar{S}\rangle) < 0$ indicates that firing $G(\bar{S})$ will increase the objective. Hence, the pivot operation follows. If the entering branch b is depleted then we fire $G(S)$ $\mu(G(S))$ times. Otherwise, we fire $G(\bar{S})$ $\mu(G(\bar{S}))$ times. The progress achieved in the pivot is $\mu(G(S))W(\langle S,\bar{S}\rangle) \geq 0$ if b is depleted or $\mu(G(\bar{S}))W(\langle \bar{S},S\rangle) \geq 0$ if b is saturated. The basis exchange is easily seen. The greedy diakoptic firing of $G(S)$ or $G(\bar{S})$ either depletes or saturates at least one chord $d \in \langle S,\bar{S}\rangle$. Each depleted or saturated chord $d \in \langle S,\bar{S}\rangle$, after a pivot, is a candidate for leaving the basis. In general, there may be multiple depleted chords and/or multiple saturated chords competing for the leaving variable. This represents a degenerate basis and we cannot simply ignore the multiplicity by selecting any candidate at random, as this can lead to cycling in the algorithm. However, since the details of degeneracy are subtle,we shall assume that simplex will not encounter a degenerate basis and note that there exists a number of anticycling techniques which are applicable to general LP problems. Hence, the assumption uniquely specifies a chord $d \in \langle S,\bar{S}\rangle$ which is either depleted or saturated after the pivot. Again, the basis exchange is denoted as $\bar{T} \leftarrow \bar{T} + \{b\} - \{d\}$, or equivalently, $T \leftarrow T - \{b\} + \{d\}$.

The relative-cost coefficients are updated in exactly the same manner as in the uncapacitated case. This is intuitively obvious since the capacities restrict the markings of G and are in no relation with the weights.

3.2.6 Boundedness

The boundedness condition follows from arguments similar to those used in the uncapacitated case and is summarized in the following theorem.

Theorem 3.7: The objective WM is bounded over $R(M_o)$ if and only if G contains no positive-weight directed cutset $(S,\bar{S})$ with $U(e) = \infty$ for all $e \in (S,\bar{S})$.■

3.2.7 Alternative Formulation of the Problem

We now present an alternative formulation of the maximum-weight marking problem, which is suitable for studying the nonlive case.

Let $\omega_i \triangleq W(\langle\{i\},V-\{i\}\rangle)$ denote the *weight* of vertex $i \in V$. The row vector of *vertex weights* $\Omega = [\omega_i]$ is then

$$\Omega \triangleq WA^t, \tag{3.20}$$

where A is the incidence matrix of G. The number ω_i represents the gain achieved in the objective WM each time vertex i is fired. Thus, if the initial marking M_o has an objective value $J_o = WM_o$ and vertex i is fired σ_i times, resulting in a marking M with an objective value $J = WM$,

then the number $\omega_i \sigma_i$ is the increase, $J - J_o$, achieved in the objective, in moving from M_o to M. Since J_o is fixed, then maximizing $J - J_o$ is equivalent to maximizing J. In fact, multiplying the state equation by W gives

$$WM = WM_o + \Omega\Sigma, \tag{3.21}$$

where Σ is the firing-count vector realizing M from M_o. Hence, we may replace the objective WM with the equivalent *relative* objective $\Omega\Sigma$.

Using the state equation, $M = M_o + A^t\Sigma$, we may pose the feasibility condition for a marking M of G in terms of the firing numbers. That is, $L \leq M \leq U$ means $L \leq M_o + A^t\Sigma \leq U$. Thus, the maximum-weight marking problem may be stated with the alternative linear program

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } L - M_o \leq A^t\Sigma \leq U - M_o. \end{aligned} \tag{3.22}$$

The special case of Program 3.22 when $L(e) = 0$ and $U(e) = \infty$, for all $e \in E$, namely

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } A^t\Sigma \geq -M_o, \end{aligned}$$

represents the uncapacitated problem.

Note that Σ is not explicitly restricted in either of these programs and that for any solution Σ, we can obtain the minimum nonnegative solution Σ_o.

3.3 Maximum-Weight Markings for Nonlive Marked Graphs

It should be easy to see that the dead-subgraph condition in the reachability theorem can be incorporated into the alternative formulation of the maximum-weight marking problem as follows. Let D be the set of all vertices in G belonging to a dead-subgraph at M_o. By definition, no vertex in D can fire at any $M \in R(M_o)$. Thus, we need merely fix the firing count of each vertex $i \in D$ at zero. Then, for any marked graph G, the maximum-weight marking problem may be stated as

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } L - M_o \leq A^t\Sigma \leq U - M_o, \\ &\qquad\qquad \sigma_i = 0, \ \forall\, i \in D. \end{aligned} \tag{3.23}$$

The above program can be simplified as follows. Consider the subgraph G(D) of G. Since each vertex $i \in D$ is dead then the marking of G(D) cannot change in any legal firing sequence of G from M_o. That is, $M(e) = M_o(e)$, $\forall\, e \in G(D)$, $\forall\, M \in R(M_o)$. Thus, all vertices in D may be short-circuited together into a single dead vertex d and then the self-loops induced on vertex d may be removed. To incorporate the dead-subgraph condition, we simply fix $\sigma_d = 0$ and solve the problem for the reduced graph. The set D is easy to identify.

3.4 Structure of the Alternate Formulation

Program 3.23 is posed in terms of the firing counts

and not in terms of markings as in the previous sections. So it might appear that the methods of Sections 3.1 and 3.2 cannot be used to study (3.23). This is not so. We now demonstrate that the maximum-weight marking problem for a nonlive marked graph can be formulated as an *auxiliary* capacitated maximum-weight marking problem which implicitly incorporates explicit restrictions on the firing numbers as explicit restrictions on the markings, and hence, that the methods discussed thus far are applicable to nonlive problems as well.

To simplify our presentation, we consider only the maximum-weight marking problem for nonlive uncapacitated marked graphs. The linear program form of this problem is the special case of Program 3.23, namely,

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } A^t\Sigma \geq -M_o, \\ &\qquad\qquad \sigma_i \geq 0, \ \forall\, i \in D. \end{aligned} \tag{3.24}$$

The first step in solving this problem with simplex is to convert the inequality constraints to equality constraints by introducing *slack* variables. It is easy to see that the column vector of slack variables for Program 3.24 is the final marking M that Σ realizes from M_o. Thus, incorporating slack variables in Program 3.24 yields the equivalent program

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } A^t\Sigma - M = -M_o, \\ &\qquad\qquad \sigma_i \geq 0, \ \forall\, i \in D, \end{aligned}$$

or

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } M - A^t\Sigma = M_o, \\ &\qquad\qquad \sigma_i = 0, \ \forall\, i \in D. \end{aligned} \tag{3.25}$$

Using the unit or identity matrix I_m of dimension m, we may then organize Program 3.25 as

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } [I_m \ -A^t]\begin{bmatrix} M \\ \Sigma \end{bmatrix} = M_o \\ &\qquad\qquad \sigma_i = 0, \ \forall\, i \in D. \end{aligned} \tag{3.26}$$

Now, construct an *auxiliary graph* $\hat{G}$ from the original graph G by introducing an *artificial reference* vertex external to G and then connecting each vertex in G to this reference vertex with an *artificial edge*. Formally, define the auxiliary graph $\hat{G} = (\hat{V},\hat{E})$ for the original graph $G = (V,E)$ according to

$$\hat{V} \triangleq V \cup \{r\}, \tag{3.27}$$

$$\hat{E} \triangleq E \cup T^*, \tag{3.28}$$

where r is an artificial reference vertex external to G and

$$T^* \triangleq \{e \mid e = (i,r),\ i \in V\} \tag{3.29}$$

is a *star tree* consisting of artificial edges directed into the artificial reference. In fact, T^* is a spanning tree of $\hat{G}$, and consequently, E is a cospanning tree of $\hat{G}$. Hence, each edge of G defines a fundamental circuit of $\hat{G}$. Let $\hat{B}_f$ be the fundamental-circuit matrix of $\hat{G}$ defined by its cospanning tree E. Then, the canonical form of $\hat{B}_f$ is

$[I_m \; \hat{B}_{f,n}]$. It is easy to see that

$$\hat{B}_{f,n} = -A^t \tag{3.30}$$

and hence, $[I_m \; -A^t]$ is the fundamental-circuit matrix $\hat{B}_f$ of $\hat{G}$ with respect to the tree/cotree partition (T^*,E) of $\hat{G}$. The solution to Program 3.26 follows easily from this point onwards. The vector Σ is a vertex associated variable. However, we may unify the notions of vertex variables and edge variables to some degree, through the use of the unique auxiliary graph $\hat{G}$ associated with every graph G. From the above discussion, it is apparent that we may construct an *auxiliary problem* for any instance of Program 3.24 as follows.

Define an initial marking $\hat{M}_o$ of $\hat{G}$ according to

$$\hat{M}_o(e) \triangleq \begin{cases} M_o(e) \text{ if } e \in E, \\ 0 \text{ if } e \in T^*, \end{cases}$$

or simply,

$$\begin{aligned} \hat{M}_o(E) &\triangleq M_o, \\ \hat{M}_o(T^*) &\triangleq 0, \end{aligned} \tag{3.31}$$

where $\hat{M}_o(E)$ is the restriction of $\hat{M}_o$ to E. Note that $\hat{M}_o$ is a basic marking of $\hat{G}$ with basis E and cobasis T^*.

Now clearly, the dead-subgraph constraint is a special case of the more general explicit upper and lowerbounding of Σ. Specifically, the constraint $\sigma_i = 0$, $\forall$ i $\in$ D is a

special case of the constraint $\ell_i \leq \sigma_i \leq u_i$, $\forall\, i \in V$, where ℓ_i and u_i are the lower bounds and upperbounds, respectively, on the firing count σ_i of vertex $i \in V$.

Define an upperbound $\hat{U}$ on the marking $\hat{M}$ of $\hat{G}$ according to

$$\hat{U}(e) \triangleq \begin{cases} \infty \text{ if } e \in E \\ u_i \text{ if } e \in T^* \text{ and } e = (i,r), \end{cases} \tag{3.32}$$

where

$$u_i \triangleq \begin{cases} 0 \text{ if } i \in D \\ \infty \text{ if } i \in \bar{D} \triangleq V - D. \end{cases} \tag{3.33}$$

Define a weighting $\hat{W}$ of $\hat{G}$ as

$$\hat{W}(e) \triangleq \begin{cases} 0 \text{ if } e \in E, \\ \omega_i \text{ if } e \in T^* \text{ and } e = (i,r), \end{cases}$$

or simply,

$$\begin{aligned} \hat{W}(E) &\triangleq 0, \\ \hat{W}(T^*) &\triangleq \Omega. \end{aligned} \tag{3.34}$$

We define the *auxiliary program* associated with Program 3.26 as

$$\begin{aligned} &\text{maximize } \hat{W}\hat{M} \\ &\quad \text{subject to } \hat{B}\hat{M} = M_o \\ &\qquad\qquad 0 \leq \hat{M} \leq \hat{U}, \end{aligned} \tag{3.35}$$

where $\hat{M}$ is a marking of $\hat{G}$.

Our intent, now, is to demonstrate that a solution to the auxiliary program yields a solution to the original

problem and vice-versa. In other words, we proceed to demonstrate that Programs 3.26 and 3.35 are equivalent.

Let $\hat{R}(\hat{M}_o)$ denote the solution space of the constraints in Program 3.35 and, as usual, let $R(M_o)$ denote the reachability set of M_o on G.

Consider any $\hat{M} \in \hat{R}(M_o)$. Clearly, $\hat{M}$ is a feasible solution to Program 3.35. Let $\hat{\Sigma} \triangleq [\hat{\sigma}_1, \hat{\sigma}_2, \cdots, \hat{\sigma}_n, \hat{\sigma}_r]^t$ be any solution to $\hat{A}^t\hat{\Sigma} = \hat{M} - \hat{M}_o$, where $\hat{A}$ is the incidence matrix of $\hat{G}$. We have $\hat{M}(e) = \hat{M}_o(e) + \hat{\sigma}_i - \hat{\sigma}_r$, $\forall\, e \in T^*$. Since $\hat{M}_o(e) = 0$, $\forall\, e \in T^*$, then $\hat{M}(e) = \hat{\sigma}_i - \hat{\sigma}_r$, $\forall\, e \in T^*$. Imposing nonnegativity on $\hat{M}$ yields $\hat{\sigma}_i - \hat{\sigma}_r \geq 0$, $\forall\, e \in T^*$ or $\hat{\sigma}_r \leq \hat{\sigma}_i$, $\forall\, i \in V$. Hence, $\hat{\sigma}_r^o = 0$ in the minimum nonnegative solution $\hat{\Sigma}_o = [\hat{\sigma}_1 - \hat{\sigma}_r, \hat{\sigma}_2 - \hat{\sigma}_r, \cdots, \hat{\sigma}_n - \hat{\sigma}_r, 0]^t$ of $\hat{A}^t\hat{\Sigma} = \hat{M} - \hat{M}_o$ and $\hat{\sigma}_i^o = \hat{M}(e)$, $\forall\, e \in T^*$. Since $\hat{M} \leq \hat{U}$, then $\hat{\sigma}_i^o \leq u_i$, $\forall\, i \in V$ or $\hat{\sigma}_i^o = 0$, $\forall\, i \in D$.

If we let

$$\Sigma_o \triangleq [\hat{\sigma}_1^o, \hat{\sigma}_2^o, \cdots, \hat{\sigma}_n^o]^t, \tag{3.36}$$

then we can see that

$$A^t\Sigma_o = \hat{M}(E) - \hat{M}_o(E) = \hat{M}(E) - M_o.$$

This means that $M = \hat{M}(E)$ and Σ_o as defined in (3.36) constitutes a feasible solution of Program 3.26. Furthermore, this solution of (3.26) corresponding to $\hat{M}$ is unique since $\hat{\sigma}_i^o = 0$ for at least one value of $i \in V$. Thus, it follows that each feasible solution $\hat{M}$ of (3.35)

defines a unique feasible solution of (3.26). Further, the objective values corresponding to these solutions are both equal to $\Omega\Sigma_o$. But, $\Omega\Sigma_o = WM - WM_o$.

Starting from any feasible solution of (3.26) and retracing the above arguments, we can show that each such feasible solution defines a unique feasible solution of (3.35), and the corresponding objectives are both equal. These discussions prove the following.

Theorem 3.8: Programs 3.26 and 3.35 are equivalent.■

The equivalence proved in the above theorem demonstrates that the maximum-weight marking problem for nonlive marked graphs possesses the same structure as that for the live marked graphs.

3.5. Summary

In this chapter, we first presented a linear programming formulation of the maximum-weight marking problem on live, marked graphs. We have described the details of an algorithm (based on the simplex method) to obtain a maximum-weight marking. The concepts of basic markings and diakoptic firings have been defined. It is shown that each pivot in the simplex method corresponds to a diakoptic firing. An algorithm requiring only vertex firings is given to construct a basic feasible marking from a given initial marking. In addition to constructing a maximum-weight mark-

ing, our algorithm constructs a firing sequence leading from the initial marking to a maximum-weight marking. However, this firing sequence may not have minimum scatter as defined in Chapter 2. We have also established the diakoptic-reachability theorem.

We have given details of an algorithm to construct a maximum-weight marking in the case of capacitated marked graphs. Finally, a formulation of the problem is given in terms of firing counts only. Using this formulation, we have studied the maximum-weight marking problem for the nonlive class of problems. We have shown that the maximum-weight marking problem has the same structure in the cases of both live and nonlive graphs.

One important advantage of the linear-programming formulation of a problem is that it makes sensitivity analysis of the problem easy. Thus, our formulations would facilitate the study of the effects of small changes in the initial markings on the optimal solution.

We conclude by again pointing out that the problem of determining the maximum resource requirements in the computation graph model of Karp and Miller [3] reduces to the maximum-weight marking problem in the case where the input and the output quanta as well as the threshold of each edge of the computation graph are equal.

Chapter 4

STRUCTURE OF THE SUBMARKING-REACHABILITY PROBLEM

The *reachability* problem studied in Chapter 2 concerns determining whether or not a given marking M is a member of the reachability set $R(M_o)$ of a given initial marking M_o of a marked graph G. The reachability theorem due to Murata [5] provides a circuit-theoretic characterization of $R(M_o)$ for a marked graph G. This theorem has allowed us to solve different problems related to reachability for marked graphs. In this chapter, we explore a generalization of the reachability problem called the *submarking-reachability* problem introduced and studied by Kumagai, Kodama and Kitagawa [6]. Whereas, in the reachability problem, a final marking is completely specified or specified on each edge of G, in the submarking-reachability problem, a final marking is specified only on a subset of the edges of G and no marking is specified for the remaining edges. The submarking-reachability problem concerns determining whether or not there exists a marking with the specified final token distribution in the reachable space of a given initial marking of a marked graph G. This generalization introduces degrees of freedom into the reachability problem. The reachability problem is the special case of the submarking-reachability problem when the final marking is specified on all edges of G, leaving no degrees of freedom.

In their pioneering work, Kumagai, Kodama and Kitagawa [6] have provided an approach for the study of the submarking-reachability problem and have described an algorithm for constructing a marking with the specified final submarking, that is reachable from an initial marking, whenever it exists. However, their study does not fully expose the structure of the problem. As a result, extension of their approach to the study of the submarking-reachability problem for the capacitated case has not been easy [7].

In this chapter, we formulate the submarking-reachability problem as a linear program and demonstrate how to solve this problem after reducing it to an equivalent smaller problem by relaxing the feasibility constraints and by introducing an intermediate state. We then define the submarking-reachability problem for capacitated marked graphs and show that a similar reduction and solution technique applies to the capacitated case. As we shall see in Chapter 5, this approach enables us to establish the link between the submarking-reachability problem and a feasibility testing problem in operations research.

4.1. Formulation of the Problem

We are given a marked graph $G = (V,E)$ with an initial marking M_o. The edge set is partitioned into the *controlled set* E_c for which a final marking $M_a(E_c)$ is specified and the remaining edges constitute the *free set* E_f for

which a final marking $M_f(E_F)$ is not specified. The partitioning of E into E_C and E_F partitions V into V_C and V_F, where V_C is the set of all vertices of G incident to an edge of E_C and $V_F = V - V_C$. The vertices in V_C are called the *controlled vertices* and those in V_F are called the *free vertices*. The incidence matrix A of G partitions as follows.

$$A = \begin{array}{c} \begin{matrix} E_C & E_F \end{matrix} \\ \begin{bmatrix} A_{CC} & A_{CF} \\ 0 & A_{FF} \end{bmatrix} \begin{matrix} V_C \\ V_F \end{matrix} \end{array}$$

As an example, Figure 4.1(a) illustrates a marked graph G with an initial marking M_o. Here, edges are numbered 1,2,3,4,•••,44. The final submarking is shown in Figure 4.1(b). The controlled edges are drawn as heavy lines to distinguish them from the remaining free edges. The controlled vertex set $V_C = \{a,b,\bullet\bullet\bullet,s\}$ and the free vertex set $V_F = \{t\}$.

Let $\Sigma = [\Sigma_C^t, \Sigma_F^t]^t$ be a firing-count vector associated with V, partitioned according to V_C and V_F, respectively. Every executable Σ on G from the initial marking M_o results in a marking M given by the state equation $M = M_o + A^t\Sigma$. Partitioning the state equation accordingly gives

$$M(E_C) = M_o(E_C) + A_{CC}^t\Sigma_C, \tag{4.1}$$

$$M(E_F) = M_o(E_F) + A_{CF}^t\Sigma_C + A_{FF}^t\Sigma_F. \tag{4.2}$$

The submarking-reachability problem is to decide

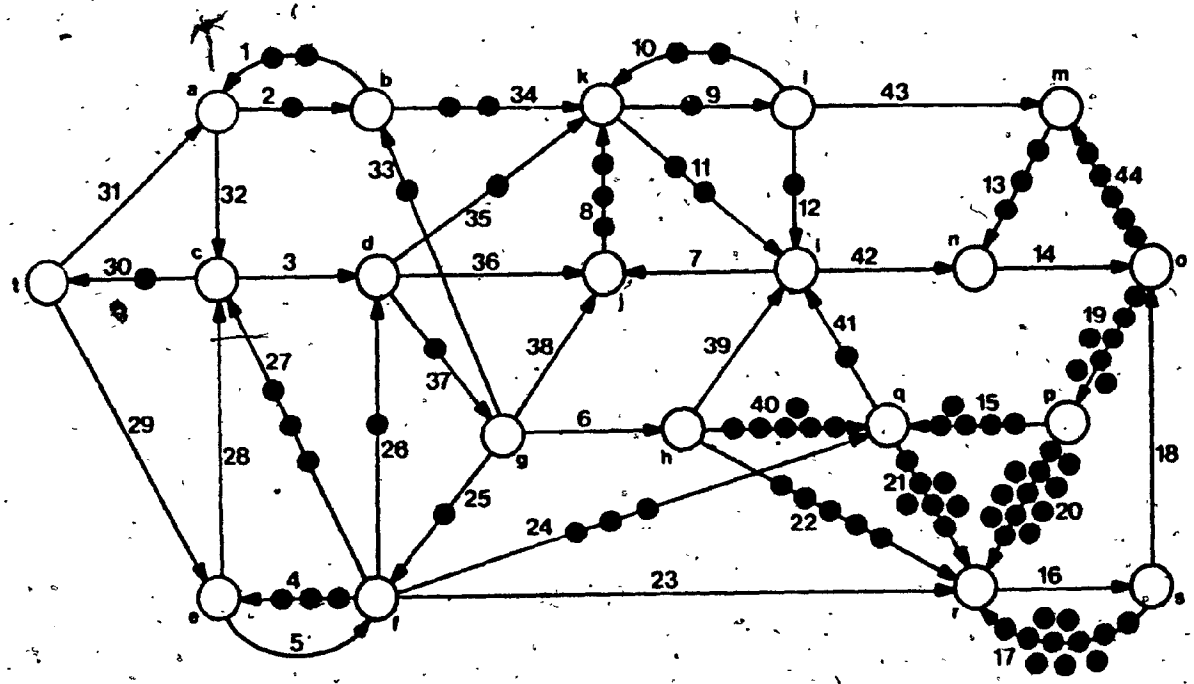


Figure 4.1(a)

A marked graph with initial marking M_o

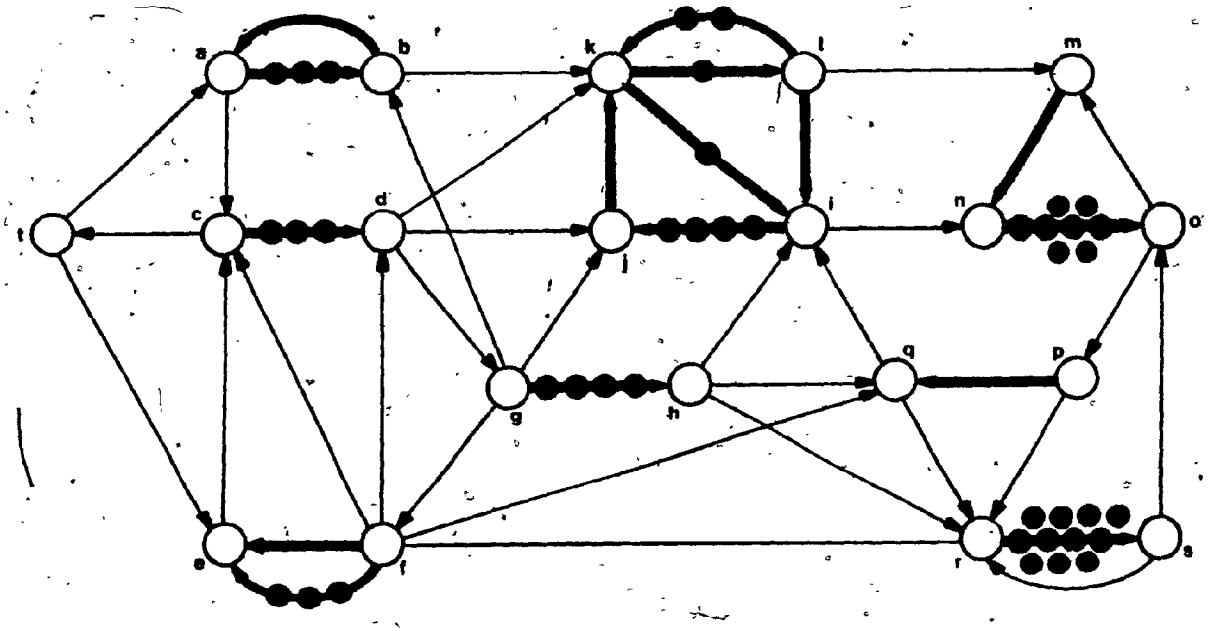


Figure 4.1(b)

The final submarking

whether or not $M_s(E_C)$ is reachable from M_o and if so, determine the minimum firing-count vector Σ leading from M_o to some marking M for which $M(E_C) = M_s(E_C)$ and $M(E_r)$ is feasible; i.e., $M(E_r) \geq 0$. As usual, the minimum firing-count vector realizing such an M from M_o must satisfy the dead-subgraph condition of the reachability theorem of Chapter 2. Since the dead-subgraph condition must be satisfied and can be checked easily after M is known, then we may simply neglect the dead-subgraph restriction for now by considering live problems only. At first, this seems as if it could be computationally difficult if there exist multiple markings M satisfying Equations 4.1 and 4.2. However, by further specifying that M should be the *nearest* such marking to M_o, in the sense that the minimum firing-count vector realizing M from M_o is minimum among all minimum firing-count vectors realizing such a marking from M_o, we render the solution to the submarking-reachability problem unique. If the minimum firing-count vector realizing the nearest M, whenever it exists, does not satisfy the dead-subgraph condition of the reachability theorem, then, nor will any other. Thus, we may state the submarking-reachability problem for live marked graphs as the linear program

$$\begin{aligned} &\text{minimize } \Sigma \\ &\quad \text{subject to} \\ &\qquad A_{CC}^{t}\Sigma_C = M_s(E_C) - M_o(E_C), \\ &\qquad A_{Cr}^{t}\Sigma_C + A_{rr}^{t}\Sigma_r \geq -M_o(E_r), \\ &\qquad \Sigma \geq 0, \end{aligned} \tag{4.3}$$

in which the notation "minimize Σ" means minimize $\sum_{i=1}^{n} \sigma_i$.

4.2. Structure of the Problem

Let us investigate the structure of Program 4.3. The edge induced subgraph $G_c = G(E_c) = (V_c, E_c)$ may consist of a number of maximally-connected subgraphs or *components* and consequently, the equality constraints in (4.3) decompose into disjoint subsystems - one for each component of G_c. Let G_c consist of r components $G_c^1 = (V_c^1, E_c^1)$, $G_c^2 = (V_c^2, E_c^2)$, $\cdots$, $G_c^r = (V_c^r, E_c^r)$. Thus, the incidence matrix of G has the following structure

$$A = \begin{array}{c} \begin{array}{cccccc} E_c^1 & E_c^2 & & E_c^k & E_c^r & E_r \end{array} \\ \left[\begin{array}{cccc|c} A_{cc}^1 & & & & & V_c^1 \\ & A_{cc}^2 & & 0 & & V_c^2 \\ & & & & A_{cr} & \\ & & A_{cc}^k & & & V_c^k \\ & 0 & & & & \\ & & & A_{cc}^r & & V_c^r \\ \hline & & 0 & & A_{rr} & V_r \end{array}\right] \end{array} \tag{4.4}$$

and hence, the equality constraints in (4.3) decompose to

$$(A_{cc}^k)^t \Sigma_c^k = M_z(E_c^k) - M_o(E_c^k), \quad k = 1,2,\cdots,r, \tag{4.5}$$

where A_{cc}^k is the incidence matrix of $G_c^k = (V_c^k, E_c^k)$ and Σ_c^k is the firing-count vector associated with V_c^k.

To illustrate this structure, Figure 4.2 shows the controlled subgraph $G_c = (V_c, E_c)$ of the graph $G = (V, E)$ shown in Figure 4.1, obtained by simply removing the free edges and free vertices from G. Each of the isolated portions of G_c is a maximally-connected subgraph or component of G_c.

The solution to each subsystem in (4.5) is unique to within an additive constant, if it exists. Each one of these subsystems corresponds to an independent reachability problem on the corresponding controlled component. Clearly, if $M_s(E_c^k)$ is not independently reachable from $M_o(E_c^k)$ on any component $G_c^k = (V_c^k, E_c^k)$ of $G_c = (V_c, E_c)$, then $M_s(E_c)$ is not reachable from M_o on G. Thus, in order to solve the submarking reachability problem, we must first solve a number of reachability subproblems.

Assuming that (4.5) has a solution, let $\hat{\Sigma}_c^k$ denote the minimum nonnegative solution for the k^{th} subsystem. Clearly, $\hat{\Sigma}_c^k$ leads from $M_o(E_c^k)$ to $M_s(E_c^k)$. Each $\hat{\Sigma}_c^k$ contains at least one zero entry and every solution to the k^{th} subsystem in (4.5) can be expressed as

$$\Sigma_c^k = \hat{\Sigma}_c^k + [\gamma_k, \gamma_k, \cdots, \gamma_k]^t_{(1x|V_c^k|)} \qquad (4.6)$$

for some scalar constant γ_k. As an example, we have given in Figure 4.2 (the controlled subgraph) the components of $\hat{\Sigma}$ within the corresponding vertices. In order to obtain a uniform notation, let $w = |V_r|$ and consider each free vertex $i \in V_r$ as a trivial controlled component of G with no controlled edges. Thus, its minimum firing-count vector $\hat{\Sigma}_r^i = [\sigma_i]^t = 0$ and, obviously, any firing count of vertex i can be expressed as $\Sigma_r^i = \hat{\Sigma}_r^i + [\gamma_i]^t$, with $\gamma_i = \sigma_i$. We now have a γ_i for every controlled component of G, including the trivial components. Let $\hat{\Sigma}_c = [(\hat{\Sigma}_c^1)^t, (\hat{\Sigma}_c^2)^t, \cdots, (\hat{\Sigma}_c^r)^t]^t$

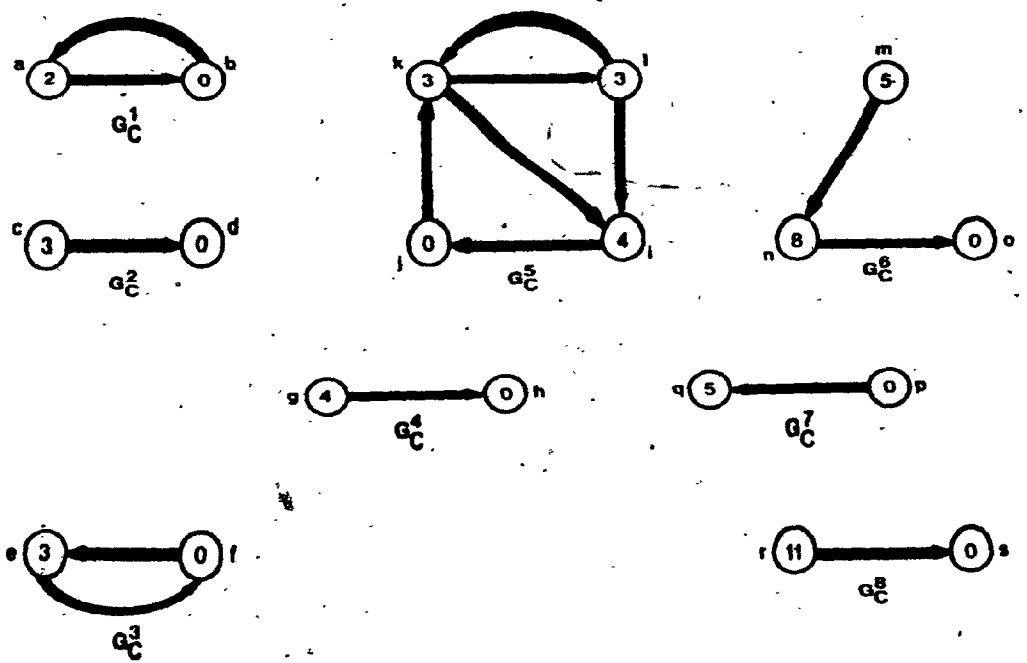


Figure 4.2

The controlled subgraph

denote the vector of $\hat{\Sigma}_c^w$ vectors, let $\hat{\Sigma}_r = [(\hat{\Sigma}_r^1)^t, (\hat{\Sigma}_r^2)^t, \cdots, (\hat{\Sigma}_r^w)^t]^t = 0$ and let $\hat{\Sigma} = [\hat{\Sigma}_c^t, \hat{\Sigma}_r^t]^t$. Then, any solution to (4.3) can be expressed as

$$\Sigma = \hat{\Sigma} + K\Gamma, \tag{4.7}$$

where K is a binary matrix of size $(n \times (r + w))$, whose $(i,j)^{th}$ entry is 1 if vertex i of G is in component j of G and 0 otherwise. The column vector

$$\Gamma = [\gamma_1, \gamma_2, \cdots, \gamma_{r+w}]^t \tag{4.8}$$

has an entry γ_j for each component j of G. We now proceed to pose (4.3) in terms of Γ, with $\hat{\Sigma}$ fixed.

4.3 Reduction of the Problem

Having obtained $\hat{\Sigma}$, let us take G to the marking $\tilde{M}$ reachable from M_o, through a possibly illegal firing sequence executing $\hat{\Sigma}$. Thus,

$$\tilde{M} = M_o + A^t\hat{\Sigma}. \tag{4.9}$$

Clearly, $\tilde{M}(E_c) = M_s(E_c)$. If $\tilde{M}(E_r) \geq 0$, then $\hat{\Sigma}$ is the unique solution to the Program 4.3; otherwise, $\tilde{M}$ is an infeasible state. Suppose that $\tilde{M}$ is infeasible. Then, our approach is to pull G out of $\tilde{M}$ and move it to a feasible state M of G through additional firings or determine that this is not possible. Assuming such a feasible state M is reachable, then, we can write M as

$$M = M_o + A^t(\hat{\Sigma} + K\Gamma),$$

$$= \tilde{M}_o + A^t K \Gamma, \qquad (4.10)$$

Since we require that $M(E_c) = M_a(E_c)$ it follows that $M(E_c) = \tilde{M}(E_c)$. So, we get from (4.10),

$$(A^k_{cc})^t \Gamma_k = 0, \quad k = 1,2,\cdots,r, \qquad (4.11)$$

where

$$\Gamma_k = [\gamma_k, \gamma_k, \cdots, \gamma_k]^t. \qquad (4.12)$$

But, (4.11) is satisfied for any arbitrary value of γ_k. Thus, in any firing sequence leading from $\tilde{M}$ to the feasible state M, all the vertices in each component G^k_c will be fired an equal number of times, namely γ_k times. So, we may consider such a firing sequence as a sequence of **diakoptic firings**. During this firing process, the markings on the edges in G^k_c are not altered and only those on the remaining edges may change. So, we need to focus our attention on the edges connecting different components including the trivial components. This suggests that we direct our attention only to the contracted graph $\tilde{G}$ obtained by short-circuiting all the vertices within each component and removing all the edges of E_c.

The graph $\tilde{G}$ may have free-edge self-loops. If any of these self-loops has a negative marking under $\tilde{M}$ then Program 4.3 is infeasible since the state of a self-loop can never change under a subgraph firing. This simple observation implies that we may remove all self-loops from $\tilde{G}$ and proceed from there, provided all self-loops are feasibly

marked under $\tilde{M}$. Thus, assume $\tilde{G}$ is free of self-loops. Also, $\tilde{G}$ may contain parallel edges. Let $E_{i,j}$ denote the set of all edges directed from vertex $i \in \tilde{G}$ to vertex $j \in \tilde{G}$. For each edge $e \in E_{i,j}$, we have $M(e) = \tilde{M}(e) + \gamma_i - \gamma_j$. Thus, the feasibility condition is

$$\gamma_i - \gamma_j \geq -\tilde{M}(e), \quad \forall\, e \in E_{i,j}. \tag{4.13}$$

Clearly, feasibility is satisfied for all $e \in E_{i,j}$ if and only if

$$\gamma_i - \gamma_j \geq \max_{e \in E_{i,j}} \{-\tilde{M}(e)\}. \tag{4.14}$$

Therefore, for any i and j, $i \neq j$, we may remove all the parallel edges, except the maximally marked one, that is, the one for which the right-hand side of (4.14) is obtained.

At this point $\tilde{G}$ has no self-loops nor parallel edges. If $\tilde{G}$ is disconnected then the problem breaks into smaller subproblems. Hence, we assume $\tilde{G}$ is connected. With the contracted graph established, let us relax the notation to help simplify our presentation. That is,from this point on, let $\tilde{A}$ denote the incidence matrix of $\tilde{G}$ and let M and $\tilde{M}$ mean the same as before, but for edges of $\tilde{G}$ only. The Program 4.3 reduces to

$$\begin{aligned} &\text{minimize } \Gamma \\ &\quad \text{subject to } \tilde{A}^t\Gamma \geq -\tilde{M}, \\ &\qquad\qquad \Gamma \geq 0. \end{aligned} \tag{4.15}$$

We wish to emphasize the fact that in the above formulation of the submarking-reachability problem, we do not distinguish between the firings of the controlled components, namely, the G_c^k's, and the firings of the free vertices. Also, in any firing sequence leading from $\tilde{M}$ to the feasible state M, firing a vertex of $\tilde{G}$ corresponds to a diakoptic firing of the corresponding component in G.

Returning to our example, Figure 4.3 illustrates the contraction of the marked graph. Figure 4.3(a) shows the marked graph of Figure 4.1 in state $\tilde{M}$. The number within each vertex v of G is the firing count σ_v of that vertex in the minimum solution of Equation 4.5. Figure 4.3(b) shows the graph which results after short-circuiting the vertices within each component. All self-loops are feasibly marked and Figure 4.3(c) shows the contracted graph $\tilde{G}$ with the reduced intermediate state $\tilde{M}$, obtained by removing all self-loops and all but minimally-marked parallel edges.

4.4. A Solution to the Submarking-Reachability Problem

As stated previously, if $\tilde{M} \geq 0$, then the solution to (4.15) is $\Gamma = 0$. Thus, we are interested in the case where $\tilde{M} \not\geq 0$. With this in mind, we proceed as follows. Let $\tilde{G} = (\tilde{V},\tilde{E})$. Let $p_{i,j}$ denote a directed path leading from vertex $i \in \tilde{V}$ to vertex $j \in \tilde{V}$. Also, let $p_{i,i}$ denote a directed circuit through vertex $i \in \tilde{V}$. Clearly, if we sum the inequality constraints in (4.15) along any directed

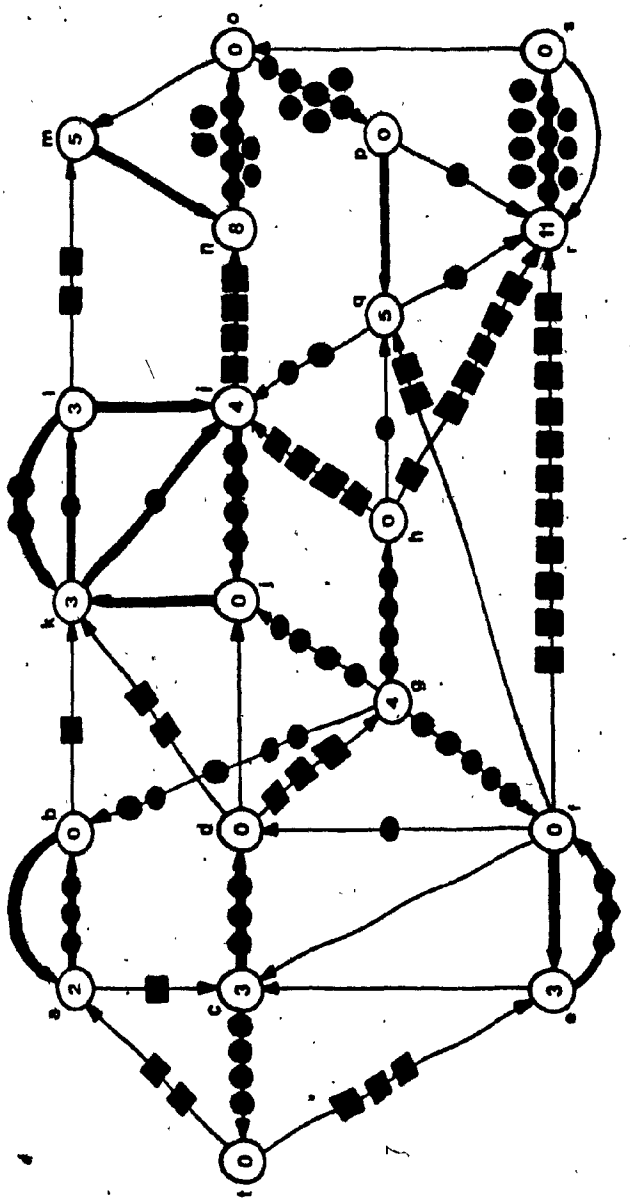


Figure 4.3(a)

The graph in state $\tilde{M}$

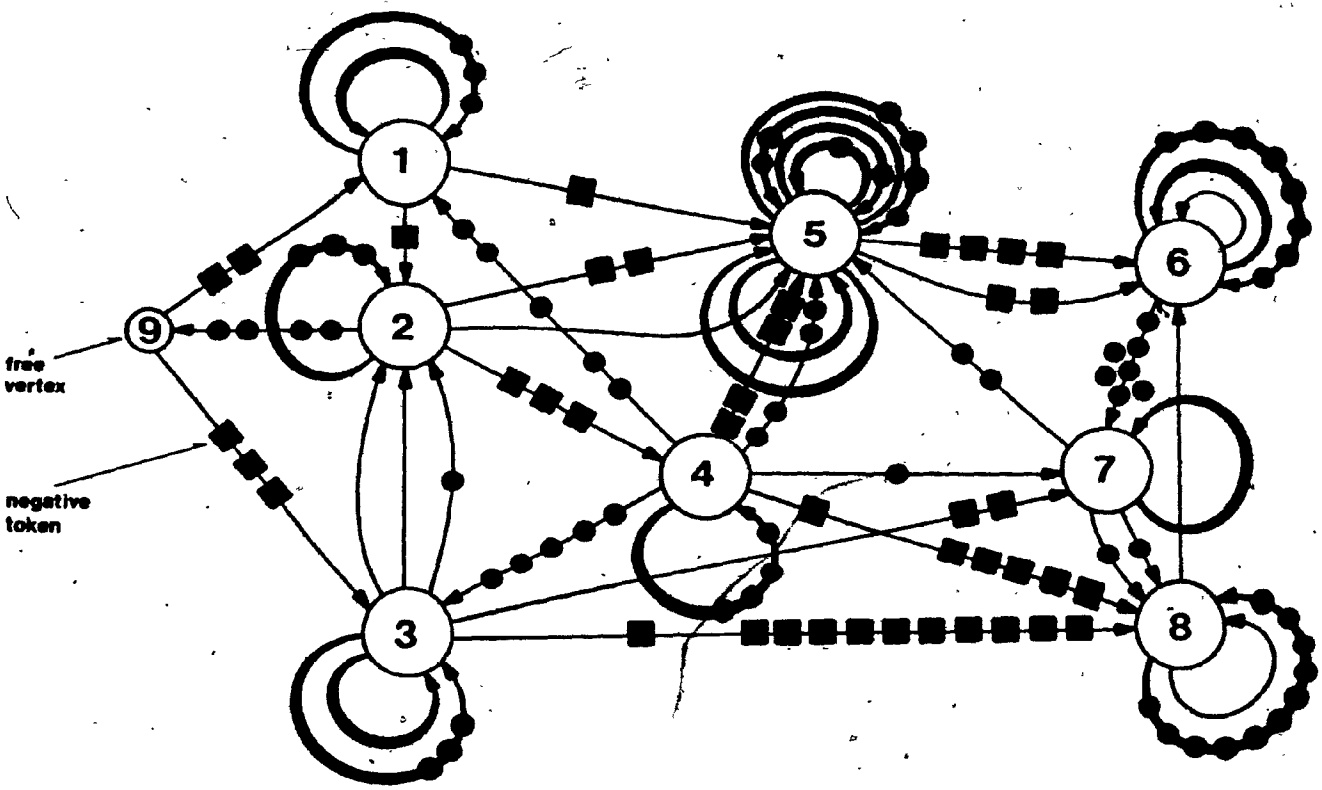


Figure 4.3(b)

Graph after contraction

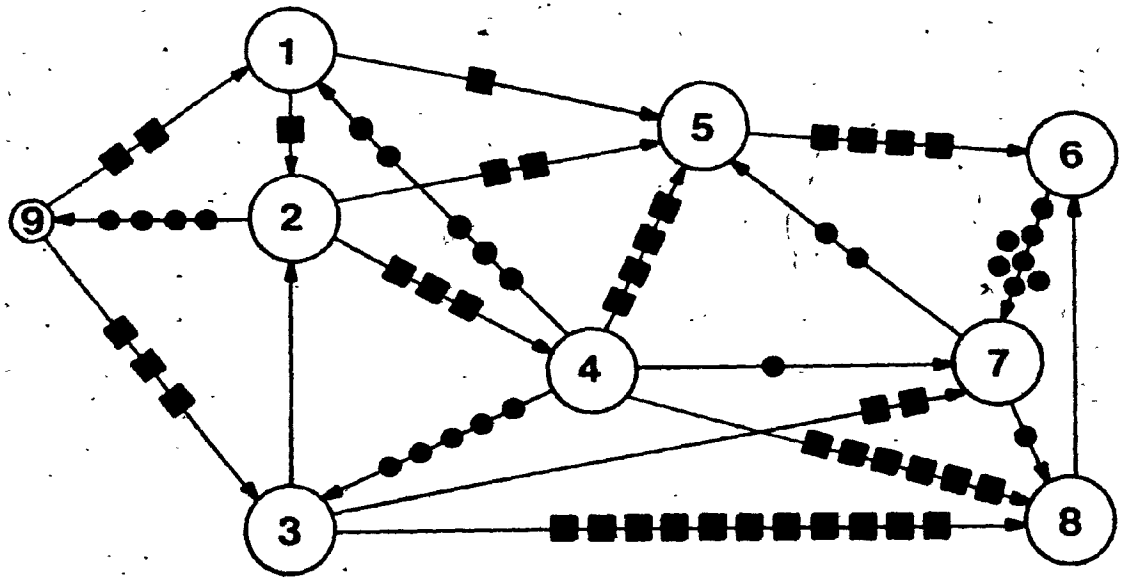


Figure 4.3(c)

The contracted graph $\tilde{G}$

circuit $p_{ii} = \{(i,u_1),(u_1,u_2),\cdots,(u_n,i)\} \in \tilde{G}$, we obtain

$$r_i - r_{u_1} + r_{u_1} - r_{u_2} + \cdots + r_{u_{n-1}} - r_{u_n} + r_{u_n} - r_i \geq -\sum_{e \in p_{ii}} \tilde{M}(e)$$

Thus, a necessary condition for the existence of a solution to (4.15) is simply

$$\sum_{e \in C} \tilde{M}(e) \geq 0, \quad \forall \text{ directed circuits } C \text{ in } \tilde{G}. \tag{4.16}$$

In other words, for the existence of a solution to (4.15), it is necessary that every directed circuit in $\tilde{G}$ has a nonnegative token count.

We now establish a simple lowerbound on the residual firing count r_i of the contracted component G_C^1. If we sum the constraints in Program 4.15 along a directed path $p_{ij} \triangleq \{(i,u_1), (u_1,u_2), \cdots, (u_{n-1},u_n), (u_n,j)\} \in \tilde{G}$ leading from vertex $i \in \tilde{V}$ to vertex $j \in \tilde{V}$ through the intermediate vertices $\{u_1,u_2, \cdots, u_n\}$, then we obtain

$$r_i - r_{u_1} + r_{u_1} - r_{u_2} + \cdots + r_{u_{n-1}} - r_{u_n} + r_{u_n} - r_j \geq -\sum_{e \in p_{ij}} \tilde{M}(e)$$

or simply,

$$r_i - r_j \geq -\sum_{e \in p_{ij}} \tilde{M}(e) \tag{4.17}$$

and hence, $r_i \geq r_j - \sum_{e \in p_{ij}} \tilde{M}(e)$. Imposing the nonnegativity requirement on Γ, namely $r_j \geq 0$, we obtain the following.

<u>Lemma 4.1</u>: $r_i \geq \max\{0, -\sum_{e \in p_{ij}} \tilde{M}(e)\}$, $\forall\, p_{ij} \in \tilde{G}$.■

Using this lowerbound on the residual firing numbers, we establish the solution to (4.15), whenever it exists, in the following theorem.

Theorem 4.1 (Submarking-Reachability Theorem): Let the token count of every directed circuit in $\tilde{G}$ be nonnegative under $\tilde{M}$. If $d_{i,j}$ denotes the shortest distance from vertex $i \in \tilde{G}$ to vertex $j \in \tilde{G}$, under $\tilde{M}(\tilde{E})$, then the unique solution to (4.15) is given by

$$r_i \triangleq \max\{0, -\min_j\{d_{i,j}\}\}, \quad \forall\, i \in \tilde{V}. \tag{4.18}$$

Proof: Let $p_{i,j}$ denote a shortest-path from vertex i to vertex j in $\tilde{G}$, under $\tilde{M}(\tilde{E})$, with distance $d_{i,j} = \sum_{e \in p_{i,j}} M(e)$. We must prove feasibility and optimality of the solution (4.18). Note that the assumption that $\tilde{G}$ under $\tilde{M}$ has no directed circuit of negative token count guarantees that all the shortest distances exist and they satisfy the triangle inequality. Thus, $d_{i,j} + d_{j,k} \geq d_{i,k}$ for distinct i,j, and k.

Feasibility: We prove the feasibility of (4.18) by showing that such an assignment, whenever it exists, results in a feasible final marking of $\tilde{G}$. We prove this by a series of contradictions. Assume that the assignments indicated in (4.18) do not correspond to a feasible final marking of $\tilde{G}$. Then, there exists at least one edge $e = (i,j) \in \tilde{E}$ such that

$$r_i - r_j < -\tilde{M}(e). \tag{4.19}$$

Now, consider the assignments of τ_i and τ_j as in (4.18). There are four cases to consider.

Case 1: $\min_k\{d_{ik}\} \geq 0$, $\min_k\{d_{jk}\} \geq 0$.

In this case, $\tau_i = \tau_k = 0$. Since $d_{ik} \geq 0$, $\forall\, k \in \tilde{V}$, it follows that $d_{ij} \geq 0$. However, with $\tau_i = \tau_j = 0$, assumption (4.19) implies $\tilde{M}(e) < 0$, contradicting that $d_{ij} \geq 0$.

Case 2: $\min_k\{d_{ik}\} \geq 0$, $\min_k\{d_{jk}\} < 0$.

In this case, $\tau_i = 0$. Let the minimization in (4.18) occur on vertex $v \in \tilde{V}$ for the assignment of τ_j. That is, vertex v is a closest one to the vertex j in $\tilde{G}$ under the intermediate marking $\tilde{M}$. By hypothesis, $d_{ik} \geq 0$, $\forall\, k \in \tilde{V}$ and specifically, $d_{iv} \geq 0$. The triangle inequality property of the shortest distances requires that $d_{iv} \leq \tilde{M}(e) + d_{jv}$. Combining this with the requirement $d_{iv} \geq 0$ implies that $\tilde{M}(e) + d_{jv} \geq 0$ or $d_{jv} \geq -\tilde{M}(e)$. However, the assumption (4.19) establishes the contradiction by requiring that $d_{jv} < -\tilde{M}(e)$.

Case 3: $\min_k\{d_{ik}\} < 0$, $\min_k\{d_{jk}\} \geq 0$.

Proof in this case follows as in Case 2.

Case 4: $\min_k\{d_{ik}\} < 0$, $\min_k\{d_{jk}\} < 0$.

Let the minimization in (4.18) occur on vertex $u \in \tilde{V}$ and vertex $v \in \tilde{V}$ for the assignments of τ_i and τ_j, respec-

tively, where possibly u = v. Then vertex u is a closest one to vertex i and vertex v is a closest one to vertex j. Thus, $d_{iu} \leq d_{ik}$, $\forall\, k \in \tilde{V}$ and, specifically, $d_{iu} \leq d_{iv}$. The triangle inequality property requires that $d_{iv} \leq \tilde{M}(e) + d_{jv}$. Hence, $d_{iu} \leq \tilde{M}(e) + d_{jv}$. Finally, assumption (4.19) implies $d_{iu} > \tilde{M}(e) + d_{jv}$, establishing the contradiction.

Thus, we have established the feasibility of the solution (4.18) to Program 4.15.

Optimality: With the feasibility of (4.18) established, optimality of (4.19) follows from Lemma 4.1.■

The existence conditions follow easily as a corollary of Theorem 4.1.

Corollary 4.1.1: The solution of program (4.15) defined in Theorem 4.1 exists if and only if $\sum_{e \in C} \tilde{M}(e) \geq 0$ for all directed circuits of $\tilde{G}$.

Proof: The existence of solution (4.18) is predicated on the existence of the shortest distances which, in turn, exist if and only if no negative-length directed circuit is present in $\tilde{G}$ under $\tilde{M}$.■

A vertex of $\tilde{G}$ is called a *datum* vertex of $\tilde{G}$ if it need not be fired in reaching the nearest feasible marking of $\tilde{G}$, whenever it exists. A datum vertex is characterized in the following Corollary of Theorem 4.1.

Corollary 4.1.2: A vertex $i \in \tilde{V}$ is a datum vertex of $\tilde{G}$

if and only if $\sum_{e \in p_{i,j}} \tilde{M}(e) \geq 0$ for all directed paths $p_{i,j}$ from vertex i in $\tilde{G}$.

Proof: From Theorem 4.1, $\gamma_i = 0$ if and only if $d_{i,j} \geq 0$, $\forall\, j \in \tilde{V}$ which is clearly equivalent to the condition

$$\sum_{e \in p_{i,j}} \tilde{M}(e) \geq 0, \ \forall\, p_{i,j} \in \tilde{G}, \ \forall\, j \in \tilde{V}. \blacksquare$$

Finally, we have the following.

Corollary 4.1.3: If the solution to Program 4.15 exists, then $\tilde{G}$ has at least one datum under $\tilde{M}$.

Proof: Consider any vertex i. Let u be a vertex closest to i so that $d_{i,u} \leq d_{i,v}$, for $v \in \tilde{V}$. We claim vertex u is a datum. If not, then $d_{u,j} < 0$ for some $j \in \tilde{V}$. Since $d_{i,u} \leq d_{i,j}$, we get, using the triangle inequality, $d_{i,u} \leq d_{i,j} \leq d_{i,u} + d_{u,j}$. So, $d_{u,j} \geq 0$, contradicting the assumption $d_{u,j} < 0$. $\blacksquare$

An interesting consequence of the above Corollary is that at least one $\gamma_i = 0$ in the solution (4.18) to Program 4.15.

We can now construct the solution to the submarking-reachability problem by combining the component solutions with the solution to the reduced problem. The minimum firing-count vector realizing the overall solution is defined as

$$\Sigma = \tilde{\Sigma} + K\Gamma \qquad (4.20)$$

where $\tilde{\Sigma}$ is the minimum nonnegative solution of (4.5) and Γ is the solution to (4.15) as defined by Theorem 4.1. The corresponding final marking of G is simply

$$M = M_o + A^t\Sigma. \tag{4.21}$$

Returning to our example, we have shown, again, the contracted graph in Figure 4.4. Here the weights on the edges refer to the token counts. To calculate τ_i's, we first obtain the shortest distances between all pairs of vertices using Floyd's Algorithm [69]. These distances are given below in matrix form.

$$[d_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \left[\begin{array}{ccccccccc} 0 & -1 & 0 & -4 & -8 & -12 & -5 & -11 & 3 \\ 2 & 0 & 1 & -3 & -7 & -11 & -4 & -10 & 4 \\ 2 & 0 & 0 & -3 & -4 & -11 & -4 & -11 & 4 \\ 5 & 4 & 5 & 0 & -4 & -8 & -1 & -6 & 8 \\ \infty & \infty & \infty & \infty & 0 & -4 & 3 & 4 & \infty \\ \infty & \infty & \infty & \infty & 9 & 0 & 7 & 8 & \infty \\ \infty & \infty & \infty & \infty & 2 & -2 & 0 & 1 & \infty \\ \infty & \infty & \infty & \infty & 9 & 0 & 7 & 0 & \infty \\ -2 & -3 & -3 & -6 & -10 & -14 & -7 & -14 & 0 \end{array}\right] \end{array}$$

Using this matrix we get τ_i's as follows:

$$\tau_1 = 12, \quad \tau_2 = 11, \quad \tau_3 = 11,$$
$$\tau_4 = 8, \quad \tau_5 = 4, \quad \tau_6 = 0,$$
$$\tau_7 = 2, \quad \tau_8 = 0, \quad \tau_9 = 14.$$

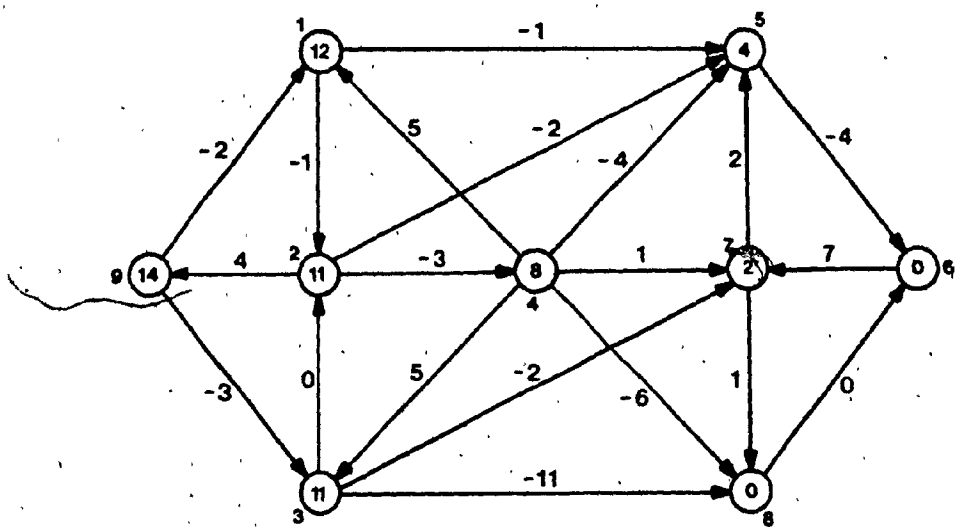


Figure 4.4

The contracted graph $\tilde{G}$ redrawn

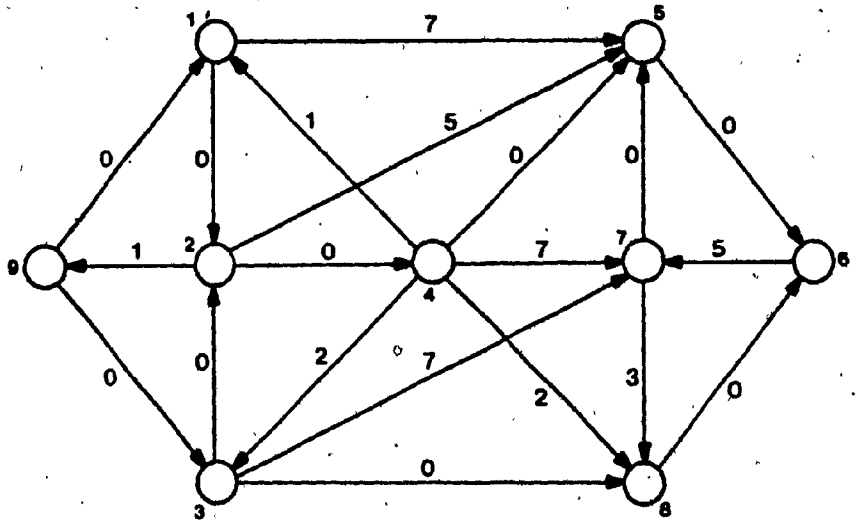


Figure 4.5

A feasible marking of the contracted graph

In Figure 4.4, we have indicated γ_i's within the circles representing the vertices. The marking shown in Figure 4.5 is a feasible marking of the contracted graph reachable from the marking shown in Figure 4.4, and it is obtained using the γ_i's shown in Figure 4.4. Using the marking in Figure 4.5, the specified final marking on the controlled edges, the intermediate marking of the free-edge self-loops and computing the final marking on the parallel edges that were removed, we obtain the feasible marking in Figure 4.6. The firing numbers realizing this marking are within indicated circles in this figure.

4.5. The Capacitated Submarking-Reachability Problem

In Chapter 2 the reachability theorem for uncapacitated graphs was easily extended for capacitated marked graphs. Similarly, the results in Chapter 3 for the maximum-weight marking problem on uncapacitated graphs were easily extended for capacitated graphs. In this section, we show how the submarking-reachability theorem (Theorem 4.1) of Section 4.4 also extends easily for capacitated marked graphs.

Again, recall that a capacitated marked graph is a marked graph $G = (V,E)$ with an integer lowerbound $L(e)$ and an integer upperbound $U(e)$ specified on the token count $M(e)$ of each edge $e \in E$. A feasible marking M of G is one satisfying $L(e) \leq M(e) \leq U(e)$, $\forall\, e \in E$ or simply, $L \leq M \leq U$

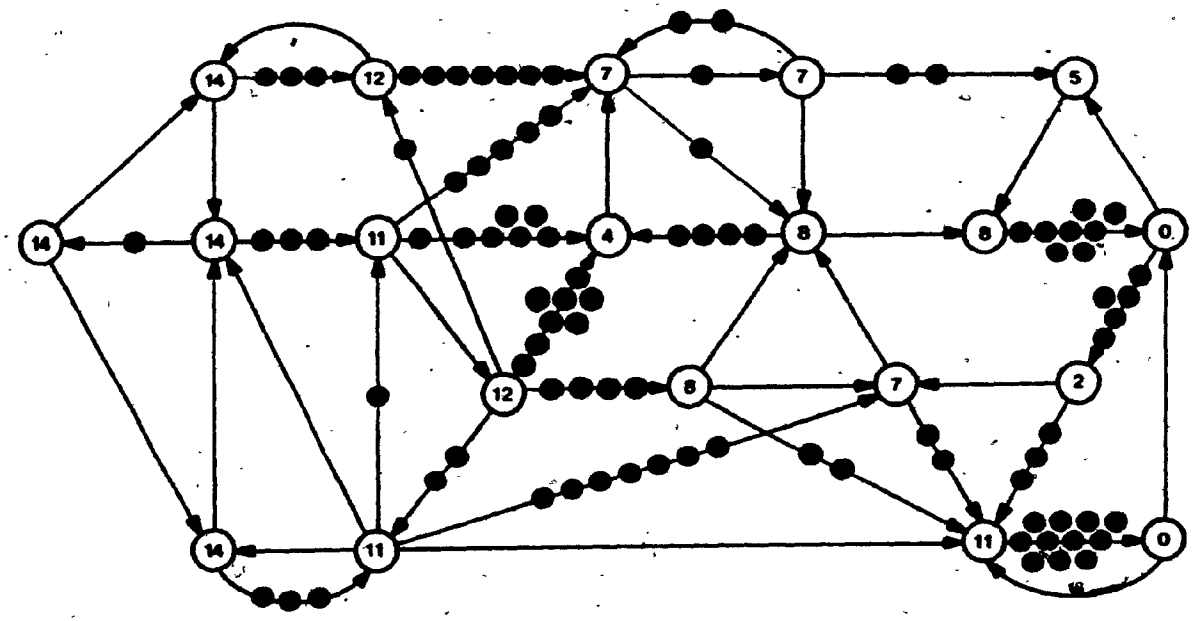


Figure 4.6

A feasible marking of the original graph

in column-vector format. We assume that L (U. The capacitated submarking-reachability problem is defined in a fashion similar to the uncapacitated version. The graph G is marked with a feasible initial marking M_o and the edge set E is partitioned into controlled edges E_c and free edges E_f induced by specifying a final feasible submarking $M_s(E_c)$ for G. Thus, the problem partitions as described in Section 4.2.

As usual, we will consider live marked graphs first and note that the arguments expressed in Section 4.1 are also valid in the context of the capacitated submarking-reachability problem. Using Equations 4.1 and 4.2 and the feasibility requirement, the capacitated submarking-reachability problem is equivalent to the linear program

$$\begin{aligned} &\text{minimize } \Sigma \\ &\quad \text{subject to} \\ &\qquad A_{cc}^{t}\Sigma_c = M_s(E_c) - M_o(E_c), \\ &\qquad L(E_f)-M_o(E_f) \leq A_{cf}^{t}\Sigma_c + A_{ff}^{t}\Sigma_f \leq U(E_f)-M_o(E_f), \\ &\qquad \Sigma \geq 0, \end{aligned} \tag{4.22}$$

where $L(E_f)$ and $U(E_f)$ denote the lowerbound and upperbound vectors associated with $M(E_f)$.

4.5.1 Structure, Decomposition and Reduction of the Problem

The topological partitioning of the marked graph is exactly as described in Section 4.2. The incidence matrix A of G has the form shown in Equation 4.4. The decomposition into subproblems is exactly as described in Section

4.2. This section may be referred to for details. We simply mention here that to solve the capacitated submarking-reachability problem, we must first solve a number of capacitated reachability subproblems.

Once the solution to the capacitated reachability subproblems has been obtained, reduction of the capacitated submarking-reachability problem follows exactly as described in Section 4.3, where we need only clarify how to handle parallel edges which may form when the controlled components of G are contracted.

In the notation of Section 4.2, let $\hat{\Sigma}$ denote the minimum solution, when it exists, to the capacitated reachability subproblems and let $\tilde{M} \triangleq M_o + A^t\hat{\Sigma}$ be the corresponding intermediate state. If $L \leq \tilde{M} \leq U$ then $\hat{\Sigma}$ is the unique solution to (4.22). Otherwise, we simply have a state $\tilde{M}$ reachable from M_o which violates the feasibility condition. Let $\tilde{G} = (\tilde{V},\tilde{E})$ denote the contracted graph which results after contracting the controlled components of G and removing the self-loops which form in this contraction. Note that if any free-edge self-loop is not feasibly marked, then (4.22) is infeasible. Let $E_{i,j}$ denote the set of all edges directed from vertex $i \in \tilde{V}$ to vertex $j \in \tilde{V}$. The feasibility condition is

$$L(e) - \tilde{M}(e) \leq \tau_i - \tau_j \leq U(e) - \tilde{M}(e), \ \forall\, e \in E_{i,j},$$

which is clearly covered by the single constraint

$$\max_{e \in E_{ij}} \{L(e)-\tilde{M}(e)\} \leq \tau_i - \tau_j \leq \min_{e \in E_{ij}} \{U(e)-\tilde{M}(e)\}. \quad (4.23)$$

At this point, a significant difference arises between the capacitated and uncapacitated problems. Although the constraint (4.23) reduces to the constraint (4.14) when $L(e) = 0$ and $U(e) = \infty$, $\forall e \in E$ and it is always consistent as defined, the covering constraint (4.23) may not be consistent in general. Specifically, when some edges are finitely upperbounded, there is no guarantee that

$$\max_{e \in E_{ij}} \{L(e)-\tilde{M}(e)\} \leq \min_{e \in E_{ij}} \{U(e)-\tilde{M}(e)\} \quad (4.24)$$

since the maximization and the minimization in (4.24) will, in general, occur on distinct edges. Clearly, if condition (4.24) is not satisfied for any set of parallel edges $E_{ij} \subseteq \tilde{E}$, then $M_R(E_C)$ is not reachable from M_o on G and hence, when reducing the capacitated submarking-reachability problem, we must further test the consistency condition (4.24) for all parallel edge sets E_{ij} which form when the controlled components of G are contracted.

Assuming (4.24) is satisfied for all parallel edge sets $E_{ij} \subseteq \tilde{E}$, we must resolve an ambiguity which may arise in removing parallel edges. Clearly, if both the minimization and the maximization in (4.23) occur on the same edge $e \in E_{ij}$ then edge e represents a most constraining edge of E_{ij} since its associated inequality is equivalent to condition (4.23) and thus, all edges of E_{ij} but edge e may be

removed from $\tilde{G}$. The ambiguity arises when the maximization occurs on, say, edge $e_L \in E_{i,j}$ and the minimization occurs on some other edge, say, $e_U \in E_{i,j}$, $e_U \neq e_L$. Clearly, all edges of $E_{i,j}$ but e_L and e_U may be removed from G. The problem is that neither edge e_L nor edge e_U alone suffices to represent inequality (4.23). The obvious next step is to replace the pair of edges $\{e_L, e_U\}$ with a single ***artificial edge***, say, e_A which does represent (4.23). If $\tilde{M}(e_L) = \tilde{M}(e_U)$, then we may simply define $L(e_A) \triangleq L(e_L)$, $U(e_A) \triangleq U(e_U)$ and $\tilde{M}(e_A) = \tilde{M}(e_L) = \tilde{M}(e_U)$ and replace $E_{i,j}$ with the artificial edge e_A. The apparent difficulty with this approach arises when $\tilde{M}(e_L) \neq \tilde{M}(e_U)$, which is the general case. We simply need an artificial edge e_A which will represent inequality (4.23) and this can be achieved by defining the parameters and intermediate state of edge e_A as

$$\begin{aligned} L(e_A) &\triangleq \max_{e \in E_{i,j}} \{L(e)-\tilde{M}(e)\}, \\ U(e_A) &\triangleq \min_{e \in E_{i,j}} \{U(e)-\tilde{M}(e)\}, \\ \tilde{M}(e_A) &\triangleq 0. \end{aligned} \tag{4.25}$$

Clearly, if all edges in $E_{i,j}$ are replaced with an artificial edge e_A whose parameters and state under $\tilde{M}$ are as defined in (4.25), then the inequality associated with edge e_A is exactly the constraint (4.23). Hence, parallel edges present no real difficulty in extending our solution to the capacitated problem. The important point is that the

existence of a solution to the capacitated submarking-reachability problem is also contingent upon the satisfaction of condition (4.24), whereas, the corresponding condition is always satisfied for the uncapacitated submarking-reachability problem.

Following the notation of Section 4.3, the reduced problem is then expressed as the linear program

$$\begin{aligned} &\text{minimize } \Gamma \\ &\text{subject to } L(\tilde{E})-\tilde{M}(\tilde{E}) \leq \tilde{A}^{t}\Gamma \leq U(\tilde{E})-\tilde{M}(\tilde{E}), \\ &\qquad\qquad \Gamma \geq 0, \end{aligned} \tag{4.26}$$

where $\tilde{A}$ is the incidence matrix of $\tilde{G} = (\tilde{V},\tilde{E})$ which may contain artificial edges as described above.

4.5.2 A Solution to the Capacitated Submarking-Reachability Problem

We now proceed to solve Program 4.26 using our solution to Program 4.15. If $L(\tilde{E}) \leq \tilde{M}(\tilde{E}) \leq U(\tilde{E})$, then it is easy to see that the solution to (4.26) is simply $\Gamma = 0$. Therefore, we consider the case when $\tilde{M}(\tilde{E})$ is not a feasible marking of $\tilde{G}$. Again, let $p_{i,j}$ denote a directed path from vertex $i \in \tilde{G}$ to vertex $j \in \tilde{G}$ and $p_{i,i}$ denote a directed circuit through vertex $i \in \tilde{G}$. Summing the inequalities in (4.25) along a directed circuit $p_{i,i} = \{(i,u_1),(u_1,u_2),\cdots,(u_{n-1},u_n),(u_n,i)\}$ in $\tilde{G}$ through the vertices $\{i,u_1,u_2,\cdots,u_n\}$ gives

$$\sum_{e \in p_{i,i}} (L(e) - \tilde{M}(e)) \leq$$

$$\pi_s - \pi_{u_1} + \pi_{u_1} - \pi_{u_2} + \cdots + \pi_{u_{h-1}} - \pi_{u_h} + \pi_{u_h} - \pi_s \leq \sum_{e \in p_{s,s}} (U(e) - \tilde{M}(e))$$

or simply

$$\sum_{e \in p_{s,s}} L(e) \leq \sum_{e \in p_{s,s}} \tilde{M}(e) \leq \sum_{e \in p_{s,s}} U(e), \tag{4.27}$$

as a necessary condition for the existence of a solution to (4.26) and, as expected, a sufficient condition is

$$\sum_{e \in C} L(e) \leq \sum_{e \in C} \tilde{M}(e) \leq \sum_{e \in C} U(e),$$

for every directed circuit C in $\tilde{G}$.

We establish the solution to (4.26) by first transforming this into an equivalent uncapacitated *auxiliary problem* and invoking Theorem 4.1. The solution follows easily once (4.26) is written in the canonical form as

$$\begin{aligned} &\text{minimize } \Gamma \\ &\quad \text{subject to } \tilde{A}^t\Gamma \geq L(\tilde{E}) - \tilde{M}(\tilde{E}), \\ &\qquad\qquad -\tilde{A}^t\Gamma \geq \tilde{M}(\tilde{E}) - U(\tilde{E}), \\ &\qquad\qquad \Gamma \geq 0. \end{aligned} \tag{4.28}$$

Now, if A is the incidence matrix of a directed graph G, then -A is the incidence matrix of the graph G′ obtained by reversing the directions of all edges of G. Hence, $-\tilde{A}$ is the incidence matrix of the graph $\tilde{G}'$ obtained by reversing the directions of all edges of $\tilde{G}$. Let $\tilde{E}' \triangleq \{(i,j) \mid (j,i) \in \tilde{E}\}$ denote the edge set of the reversed graph $\tilde{G}' \triangleq (\tilde{V},\tilde{E}')$,

Construct the matrix $\hat{A} \triangleq [\tilde{A}|-\tilde{A}]$ by horizontally concatenating $\tilde{A}$ and $-\tilde{A}$. Define a column vector N of dimension $2|\tilde{E}|$ as

$$N(\tilde{E}) \triangleq \tilde{M}(\tilde{E}) - L(\tilde{E}),$$

$$N(\tilde{E}') \triangleq U(\tilde{E}) - \tilde{M}(\tilde{E}). \qquad (4.29)$$

Then, program (4.28) is equivalent to the auxiliary program

$$\begin{aligned} &\text{minimize } \Gamma \\ &\text{subject to } \hat{A}^t\Gamma \geq -N, \\ &\qquad\qquad \Gamma \geq 0. \end{aligned} \qquad (4.30)$$

The matrix $\hat{A}$ is the incidence matrix of the *auxiliary graph* $\hat{G} \triangleq \tilde{G} \cup \tilde{G}' = (\tilde{V},\tilde{E} \cup \tilde{E}')$ obtained by superimposing $\tilde{G}'$ on $\tilde{G}$. For each edge $e \in \hat{G}$, the marking N(e) is defined as in (4.29). Clearly, in structure, Program 4.30 is equivalent to Program 4.15. Thus, at least from this point on, the capacitated submarking-reachability problem reduces to an equivalent uncapacitated problem on a larger graph. So, we state the solution to (4.26), assuming it exists, in the following theorem. Proof of correctness follows from that of Theorem 4.1.

Theorem 4.2: Let $\hat{G}$ have no directed circuit of negative token count under the marking N. If $\hat{d}_{i,j}$ denotes the shortest distance from vertex i to vertex j in the auxiliary graph $\hat{G} = (\hat{V},\hat{E}) \triangleq (\tilde{V},\tilde{E} \cup \tilde{E}')$ under N, then the unique solution to (4.30) and, hence, (4.26) is given by

$$r_i \triangleq \max\{0, -\min\{\hat{d}_{i,j}\}\}, \quad \forall\, i \in \hat{V}. \blacksquare \qquad (4.31)$$

Corollary 4.2.1: The solution of Program 4.26, as given in (4.31) exists if and only if $\sum_{e \in C} N(e) \geq 0$ for every directed circuit C in $\hat{G}$.■

As in the uncapacitated case, we can combine the solutions for the controlled components with the solutions for (4.30) and obtain the firing numbers realizing the specified final marking from the given initial marking.

4.6 Summary

In this chapter we have studied the structure of the submarking-reachability problem on marked graphs. We first formulated the problem as a linear program and demonstrated how to solve this after reducing it to an equivalent smaller problem by relaxing the feasibility constraints and by introducing an intermediate state. We then showed that a similar reduction and solution technique applies to the capacitated case also. Our approach in this chapter has, thus, resulted in a unified treatment of the submarking-reachability problem for both capacitated and uncapacitated cases. As we will see in the next chapter, this approach enables us to see the link between the submarking-reachability problem and a feasibility testing problem in operations research and motivates the need for an algorithmic solution.

Chapter 5

SUBMARKING REACHABILITY AND NETWORK PROGRAMMING

The transshipment problem [70] is one of the extensively studied problems in operations research. The network simplex method for solving this problem is a very efficient one and is well discussed in the literature. As a result, if a problem is formulated in a form dual to that of the transshipment problem, then, usually, it is converted to the equivalent transshipment problem and the solution is obtained using the network simplex method. The network simplex method involves two phases. In the first phase, the feasibility of the transshipment problem is tested and if it is feasible, a basic feasible solution is obtained. The second phase starts with this basic feasible solution and proceeds with optimization. It should be noted that the feasibility phase also requires the solution of a transshipment problem. This is so because we need a basic feasible solution to start the second phase. If feasibility alone is to be tested, then we can achieve that by solving a maximum-flow problem on the given network. However, any feasible solution obtained by this approach may not be basic which is very essential to start the second phase of the network simplex method.

In this chapter, we first point out the equivalence between the submarking-reachability problem considered in

the previous chapter and the problem of testing feasibility of the dual transshipment problem. We present an algorithm for this problem, prove its correctness and termination whenever a solution exists and finally, establish its complexity.

5.1 Submarking Reachability and Network Programming

Given a graph $G = (V,E)$ on n vertices and m edges. Let A denote the incidence matrix of G. Then, the dual transshipment problem is as follows

$$\begin{aligned} &\text{maximize } \Omega\Sigma \\ &\quad \text{subject to } A^t\Sigma \geq -M_0, \\ &\qquad\qquad \Sigma \geq 0, \end{aligned} \tag{5.1}$$

where Σ is a column vector of dimension n and Ω, called the *weight vector*, is a row vector, also of dimension n. Note that Ω and M_0 are specified.

The above problem can also be formulated as

$$\begin{aligned} &\text{maximize } WM \\ &\quad \text{subject to } B_f M = Z_T, \\ &\qquad\qquad M \geq 0, \end{aligned} \tag{5.2}$$

where B_f is a fundamental-circuit matrix of G and M is a column vector of dimension m and W is a row vector of dimension m. Here, Z_T and W are specified.

First, we note that the constraint part of (5.1) is

exactly the same as that of the Program 4.15. So, if we look upon M_o as an initial markng of G and Σ as a firing-count vector, then the problem of testing feasibility of (5.1) is exactly the same as the problem of determining a set of firing numbers for the vertices of G which transform M_o, which may be an infeasible marking, to a feasible marking M. Here, we consider all the edges of G as free edges.

On the other hand, suppose the dual transshipment problem appears as in (5.2). Then, we can identify each entry of Z_T as a marking on the corresponding chord of the cospanning tree which defines the fundamental-circuit matrix B_f. Again, testing the feasibility of (5.2) reduces to the problem of determining firing numbers which transform Z_T to a feasible marking M.

The above discussion shows that testing the feasibility of the dual-transshipment problem is the same as the submarking-reachability problem, where we are required to take G from an infeasible marking to a feasible one.

The feasible solution obtained as above may not be basic. But, we can convert it to a basic one using the algorithm presented in Section 3.1.5.

5.2 Algorithm REACH

The equivalence shown in the previous section underlines the importance of designing an efficient algorithm to solve Program 4.15. The solution to this problem is given

in Theorem 4.1. All that we need is to design an algorithm to determine τ_i's as given in this theorem. We propose the following algorithm for this purpose.

Algorithm REACH:

Let M be the given initial marking of the graph G.

While there exists an edge $e = (i,j) \in E$ with $M(e) < 0$ do
Fire vertex i, $-M(e)$ times, updating M.

The rest of this section is concerned with the proof of correctness and termination of the algorithm and its complexity analysis.

5.2.1 Proof of Correctness and Termination

In the following, the length $\ell(P_{i,j})$ of a directed path $P_{i,j}$ in G will refer to the sum of the markings under M of all the edges in $P_{i,j}$. τ_i's are as defined in Theorem 4.1. By the *size* of $P_{i,j}$, we refer to the number of edges in $P_{i,j}$. Also, $d_{i,j}$ is the length of a shortest path from i to j.

Theorem 5.1: If $\tau_i > 0$, Algorithm REACH fires vertex i at least τ_i times.

Proof: We prove the theorem by showing that if there exists a directed negative-length path $P_{i,j}$, then Algorithm REACH fires i at least $|\ell(P_{i,j})|$ times. Proof is by induction on the size of $P_{i,j}$.

Clearly, the result is true if the size of $P_{i,j}$ is 1.

Assume the result to be true for all negative-length directed paths of size $\leq k$. Consider any negative-length directed path $P_{i,j}$ of size $k+1$. Let (j',j) be the last edge in $P_{i,j}$.

Case 1: Marking $M(j',j) \geq 0$.

In this case, $\ell(P_{i,j'}) = \ell(P_{i,j}) - M(j',j) < 0$. But, $P_{i,j'}$ is a path of size k. Hence, by the induction hypothesis, vertex i will be fired at least

$$|\ell(P_{i,j'})| > |\ell(P_{i,j})|$$

times.

Case 2: Marking $M(j',j) < 0$.

In this case, at some step during the execution of Algorithm REACH, vertex j' will have been fired at least $|M(j',j)|$ times. Let M' be the marking at that step. Then, $M'(j',j) \geq 0$.

Assume that the vertices i and j' have been fired σ_i and $\sigma_{j'}$ times in reaching M' from M. We shall denote the length of $P_{i,j}$ under the marking M' with $\ell'(P_{i,j})$. Clearly, $\sigma_{j'} \geq |M(j',j)|$. There is nothing to prove if $\sigma_i \geq |\ell(P_{i,j})|$. So, assume that $\sigma_i < |\ell(P_{i,j})|$. Then,

$$\begin{aligned}\ell'(P_{i,j'}) &= \ell(P_{i,j'}) + \sigma_i - \sigma_{j'} \\ &\leq \ell(P_{i,j'}) + \sigma_i - |M(j',j)| \\ &= \ell(P_{i,j}) + \sigma_i < 0.\end{aligned}$$

So, under the marking M', the length of $P_{i,j}$ is negative. But, $P_{i,j}$ is of size k. Hence, invoking the induction hypothesis, we find that vertex i will be fired at least $|\ell(P_{i,j})| - \sigma_i$ times after M' has been reached. Thus, the algorithm will fire i at least $|\ell(P_{i,j})|$ times starting from M.■

Recall that a vertex i is called a datum in G if under the initial marking M there exists no negative-length directed path originating at i. In other words, $\gamma_i = 0$ if i is a datum.

Next, we note that, at each *step*, Algorithm REACH examines an edge and performs an appropriate vertex firing operation. So, this algorithm may be considered as performing a sequence of vertex-firing operations. Let M_i denote the marking of G at the end of the i^{th} step or i^{th} firing operation. Thus, Algorithm REACH, starting at M, takes G through a sequence of markings $M_1, M_2, \cdots, M_k, \cdots$. In the following, $d_{i,j}^{(k)}$ will refer to the length of a shortest path from i to j in G under marking M_k. Thus,

$$\gamma_i^{(k)} = \max\{0, -\min_j\{d_{i,j}^{(k)}\}\}.$$

Similarly, $\ell^k(P_{i,j})$ will refer to the length of $P_{i,j}$ under M_k.

Consider any vertex i for which $\gamma_i > 0$. Then, let i' denote a vertex such that $\gamma_i = |d_{i,i'}|$. As we have seen in

Section 4.4, each such i' is a datum vertex. Finally, we note that firing a vertex i affects only the value of r_i in the new marking. So, if the vertices $v_{i_1}, \cdots, v_{i_n}$ have been fired $\sigma_{i_1}, \sigma_{i_2}, \cdots, \sigma_{i_n}$ times to take G to the marking M_k, then under M_k,

$$r_i^{(k)} = r_i - \sigma_i, \quad \forall\ i \in V. \tag{5.3}$$

Furthermore, for each i, vertex i' continues to be a datum of i under all the markings generated by Algorithm REACH. Thus, if

$$r_i^{(k)} > 0,$$

then

$$r_i^{(k)} = d_{ii'}^{(k)} \tag{5.4}$$

If $r_i^{(k)} = 0$, then vertex i is a datum under marking M_k. These crucial properties of Algorithm REACH prove the following.

Theorem 5.2: Algorithm REACH never fires a datum.■

Theorem 5.3: If G has no negative-length directed circuits under marking M, then Algorithm REACH terminates in a finite number of steps after firing every vertex i exactly r_i times.

Proof: By Theorem 5.1, Algorithm REACH fires each vertex i at least r_i times. By Theorem 5.2, a datum is never fired in this algorithm. This means that each step results in

reducing the value of exactly one r_i. Thus, all the r_i's will eventually be reduced to zero, or equivalently, every vertex i will be fired exactly a total of r_i times in no more than $\sum_{i=1}^{n} r_i$ steps. When all the r_i's reduce to zero, then in the resulting marking there will be no edges with negative tokens and Algorithm REACH will terminate.■

5.2.2 Complexity Analysis of Algorithm REACH

To bound the number of computational steps required by Algorithm REACH, we implement the algorithm as follows.

First, order the edges as $e_1, e_2, \cdots, e_m$, where m is the number of edges in G. Now, execute the algorithm by first examining e_1, then e_2 and so on, and firing the vertices the appropriate number of times. After the first such *sweep*, perform additional sweeps until an entire sweep results in no firings.

Theorem 5.4: Assume that G has no negative-length directed circuits under the initial marking M. Consider any vertex i for which $r_i > 0$. If the size of the path $P_{ii'}$ is k, then the vertex i will have been fired r_i times at the end of the k^{th} sweep. (Note: i' is a datum vertex and $|\ell(P_{ii'})| = r_i$.)

Proof: Let $P_{ii'} = i, i_1, i_2, \cdots, i_{k-1}, i'$. At the beginning of the first sweep, the marking on the edge (i_{k-1}, i') is $-r_{i_{k-1}}$. Also, i' is never fired. So, at the end of the

first sweep, vertex i_{k-1} will have been fired $\tau_{i_{k-1}}$ times. At the beginning of the second sweep i_{k-1} is a datum and the marking on the edge (i_{k-2},i_{k-1}) will be $-\tau_{i_{k-2}}$. So, during this sweep vertex i_{k-2} will be fired $\tau_{i_{k-2}}$ times. Repeating these arguments, we can see that at the end of the k^{th} sweep, vertex i will have been fired τ_i times.■

Theorem 5.5: If G has no negative-length directed circuits under M, then Algorithm REACH will terminate in no more than n sweeps, where n is the number of vertices in G.

Proof: Each directed path P_{i1} in G is of size $\leq$ n-1. So, by Theorem 5.4, each vertex i will be fired a total of τ_i times in no more than n-1 sweeps and the theorem follows.■

During each sweep, m edges are examined. Examining an edge requires computing the value of its current marking and firing a vertex. These operations take constant time. Thus, each sweep takes O(m) time and, hence, we have the following theorem.

Theorem 5.6: Complexity of Algorithm REACH is O(mn), if G has no negative-length directed circuits under the initial marking M.■

As an example, it may be verified that Algorithm REACH when applied to the graph G of Figure 4.4 terminates in the feasible marking shown in Figure 4.5. The number of times each vertex is fired by Algorithm REACH are shown in Figure

4.4.

We now show how to incorporate, in Algorithm REACH, a mechanism to detect the presence of a negative-length directed circuit in the graph under a given initial marking.

We associate with each vertex i a label denoted as LABEL(i). To begin with, the label of every vertex is set equal to zero. Every time an edge (i,j) with a negative marking is encountered and, as a result, vertex i is fired, we update LABEL(i) setting it equal to j. If, during the n^{th} sweep, the label of any vertex, say i, changes, then it indicates the presence of a negative-length directed circuit. This directed circuit can be obtained by tracing the label values starting at i.

5.3 Summary

In this chapter, we have pointed out the equivalence between the submarking-reachability problem and the problem of testing feasibility of the dual transshipment problem. We have presented an algorithm called Algorithm REACH to solve this problem. We have also established the correctness, termination and complexity of the algorithm. It is possible to incorporate in Algorithm REACH a mechanism to detect infeasibility of the given problem. Starting from a feasible solution obtained by Algorithm REACH, we can get a basic feasible solution using the algorithm presented in

Section 3.1.5. Since the complexity of the latter algorithm is $O(n^2)$, we conclude that the complexity of testing feasibility of a dual transshipment problem is $O(mn)$. This is in contrast to the complexity $O(n^3)$ of testing the feasibility of a transshipment problem by the maximum-flow algorithm presented in [71].

Chapter 6

DISTRIBUTED ALGORITHM REACH

The advances in computers and communications have made feasible the design of a network of processors coupled together with communication links. Such systems are referred to as *distributed systems*. The interconnection of the processors in a distributed system is often modeled by a graph. Thus, associated with each distributed system is a graph whose vertices represent the processors and whose edges represent communication links interconnecting the processors. Each processor has information about the links attached to it. The global properties that are to be computed in the system can be formally expressed as graph problems which should be solved in the above model of computation. Algorithms for computing the properties of graphs in the above model of computation are referred to as *distributed algorithms*. In short, *distributed computing* is concerned with the design of algorithms in which the processors associated with the vertices make use of the local information and cooperate with their neighbours to compute the desired properties. Distributed algorithms for several graph problems have been reported in the literature [67], [72] -- [78].

In this chapter, we distribute Algorithm REACH (of Chapter 5), that is, we obtain a distributed version of

REACH. We shall use the terms "vertex" and "node" interchangeably since this is a common practice in distributed computing literature.

6.1 On Distibuting Algorithm REACH

We may recall that the problem that Algorithm REACH solves is the following. Given a directed graph G with a given initial marking M which may not be nonnegative, take G to a nonnegative marking using a sequence of vertex firing operations. Algorithm REACH will produce a solution, whenever it exists, which requires a minimum number of firing operations. As we have pointed out in Section 5.2.2, we can also include in this algorithm a mechanism to detect the presence of a negative-length directed circuit in G which implies infeasibility of the problem. To distribute this algorithm, we pose the problem on an intelligent graph. An *intelligent graph* is a graph whose nodes have processing capabilities (intelligence) and can communicate with each adjacent node by passing messages over the links. We assume that transmission over the links is error-free or, at least, that there is a retransmission strategy which ensures message transmission and reception. Each input edge of node i is assigned a number from 1 through $d^-(i)$, where $d^-(i)$ is the in-degree of node i. Similarly, each output edge of node i is numbered from 1 through $d^+(i)$, where $d^+(i)$ is the out-degree of node i. With this numbering scheme, In_Edge(k) and Out_Edge(k)

denote the k^{th} input or output edge of a node, respectively. Also, a node j adjacent to node i will be referred to as an *out-neighbour* of i if the edge connecting i and j is directed from i to j. The *in-neighbour* of i is similarly defined.

The independence property of the vertex firing operation of Algorithm REACH implies that this algorithm is highly amenable to distribution. The condition under which a vertex is eligible for firing is determined solely by the markings on its output edges. This local condition means that an intelligent node can trigger its own firing by examining its output edge markings. Having done so, an intelligent node must then communicate this information to its neighbouring nodes so that they can react accordingly. Let us examine this idea a little more closely. To illustrate the degree of concurrency present, consider the unrestricted version of Algorithm REACH, that is, when there are no negative-length directed circuits present in G under M. It is clear that nodes need only keep track of the markings on their output edges. The collective data structures form a partition of the edge set of G and, hence, all markings are accounted for in a distributed sense. Since feasibility is guaranteed then each node simply fires whenever it detects a negative marking on one of its output edges. In doing so, it updates its own data structures and then informs each of its neighbouring nodes at the other end of its input edges that it has fired.

Theorem 5.4 guarantees that firing will eventually stop and when it does, we have a solution (in a distributed sense). The operation of the algorithm is totally asynchronous. To avoid unnecessary communication, an intelligent node i will determine

$$\delta \triangleq \max\{0, -\min_{e \in E_i^+} \{M(e)\}\},$$

where $E_i^+ \triangleq \{e = (i,j) | (i,j) \in E\}$ is the set of output edges of node i. If $\delta \leq 0$, node i will not react but will respond to messages from neighbours. Alternatively, if $\delta > 0$, then node i is eligible for firing δ times. Node i then fires by accumulating the firing count and adding δ tokens to each of its output edges. It then broadcasts the fact that it has fired δ times to each of its in-neighbours. When a node i receives such a message from one of its out-neighbours, say node j, then node i must update its record of the marking on edge (i,j) by simply subtracting δ tokens from edge (i,j), where δ is in the message from node j, telling node i how many times node j fired. Thus, messages always flow against the edge directions. It does not matter which of a node's in-neighbours is first informed of a node's firing nor does it matter in which sequence the in-neighbours are informed. All that matters is that they are all eventually informed of the node's firing. Remarkably, this chaos will always terminate in a solution if there are no negative-length directed circuits in the graph. This

is simply a property of the algorithm. It is independent of any distributed architecture. To obtain the general distributed version of Algorithm REACH, that is, with infeasibility detection capability, we must enforce some global firing control mechanism so as to invoke a cessation of activity whenever certain conditions are met, indicating the presence of a negative-length directed circuit.

In order to devise such a control, we shall first consider, in the following section, a sequential implementation of Algorithm REACH which incorporates a mechanism to detect a negative-length directed circuit.

6.2 Negative-Length Circuit Detection in Algorithm REACH

In Section 5.2, we pointed out how we can incorporate, in Algorithm REACH, a mechanism to detect the presence of a negative-length directed circuit. This was based on an implementation of REACH which uses the concept of a *sweep*. However, this implementation is not an appropriate choice if we wish to distribute REACH. We now describe another implementation which is quite amenable for distributing. This implementation requires searching the given graph in a depth-first fashion and is described below.

To begin with, we label all the vertices "new". We then select a new vertex, say, vertex r such that it has at least one input edge (i,r) with negative token count. We label vertex r "old" and start the search traversing the

edge (i,r) in the direction from r to i. In other words, our traversal of an edge is in a direction opposite to its orientation. After reaching i, we fire it so that the token count on edge (i,r) becomes zero. The vertex i is then labelled "old". The search now continues from i, looking for an input edge with a negative token count. If no such edge is present, we label i "new" and return (backtrack) to r along the edge (i,r). The search resumes at r and looks for an input edge with negative token count. Note that as we traverse the graph from r, we create a directed path of zero length terminating at r. The general step of the algorithm is as follows.

Suppose we have reached a vertex $v \neq r$ after traversing an edge (v,y) with negative token count. Then, we first fire v so that the token count on (v,y) becomes zero. One of the following cases will now arise.

i) vertex v is old;

In this case, we have detected a negative-length directed circuit. Algorithm REACH terminates here.

ii) vertex v is new;

In this case, we label v "old". The search then continues at v, looking for an input edge with negative token count. If no such edge is present, the search labels vertex v "new" and returns (backtracks) to y. Search now resumes

at y looking for an input edge with negative token count.

Suppose the search returns to r and finds no input edge with negative token count. In this case, Algorithm REACH terminates if all the edges in the graph have nonnegative token counts. Otherwise, we pick another vertex, say r', and start the search at this vertex.

The above implementation of REACH is quite simple and is very useful in designing a distributed version of this algorithm. However, we should point out that this implementation does not help us in getting a good bound on the number of steps required to execute REACH. This was the reason we preferred the implementation presented in Section 5.2 for the sequential case.

6.3 Distributed Algorithm REACH

In this section, we present a distributed version of Algorithm REACH.

Let G = (V,E) be the directed graph respresenting the given problem and let M be the initial infeasible marking of G. A node i ∈ V with

$$\min_{e \in E_i^+} \{M(e)\} < 0$$

is called an *activator*. Otherwise, node i is called a *nonactivator*. Each activator initiates a *firing wave* consisting of information relating to its initial firing and

all the subsequent firings induced by this firing throughout the graph. It is clear that all firing waves eventually subside when G contains no negative-length directed circuit. From this point of view, the distributed Algorithm REACH can be visualized as a number of concurrent firing waves, each initiated by one of the activators. From the independence property of firings, it should be clear that these concurrent firing waves interfere with one another constructively, from the point of view of getting closer to the answer, simply because the total number of firings executed by Algorithm REACH is independent of the order in which nodes fire. There is no real need to associate or identify a wave with any particular activator in the unrestricted case because after activators fire, we simply obtain a new group of activators, some of which may have been previous activators and since termination is guaranteed, then eventually, all nodes become nonactivators. Identifying a wave with an activator is an easy way to incorporate some distributed control in our distributed algorithm which would help us detect a negative-length directed circuit. It may be noted that the extent of a wave's propagation is not unique for an instance of a problem. The extent of a particular wave's propagation through G is defined only through the actual sequence of events which occur in each execution of the algorithm, for an instance of a problem. The waves interfere with one another as they propagate. It is this interaction between

the different waves that makes complexity analysis of the algorithm very difficult. However, an interesting aspect of this interaction is that it is a beautiful example of a blind cooperation toward a common goal.

In order to incorporate a negative-length directed circuit detection mechanism into our distributed algorithm, consider the possibility of guiding the propagation of each firing wave emanating from an activator in such a way that a node can, at some point, detect the existence of a negative-length directed circuit. In the sequential case, we guided Algorithm REACH in a depth-first fashion. We now present a distributed implementation of this technique. In our implementation, we incorporate an echo mechanism to cooperate with the wave mechanism.

Let $S \subseteq V$ be the set of all firing-wave activators in G under M. As we have stated before, each activator $w \in S$ activates wave w. Each wave propagates via a single wave message. Cooperating with each wave is a unique *echo* which propagates via a single echo message in a direction opposite to the direction of its associated wave. The echo is simply an acknowledgement scheme. Simply put, an echo is expected in return for each wave that is transmitted. By controlling the way in which the wave propagates and the way in which the echo returns, we can incorporate our depth-guided negative-length directed-circuit detection mechanism. Echo w is said to be *pending on* edge $(i,j) \in E$,

at node j, whenever node j is expecting the echo w message from node i in return for the wave w message transmitted from node j to node i along link (i,j). Echo w is said to be *pending* at node i if echo w is pending on any input edge of node i. Echo w is said to be *pending from* node i if echo w is pending on any output edge of node i.

Suppose that node w activates a firing wave. A node i ≠ w is said to be *participating* in wave w if echo w is pending from node i. An activator w participates in its own wave from its creation until its cessation. A node can participate in up to n firing waves simultaneously. A node i is said to be *actively* participating in wave w whenever echo w is pending at node i. An actively participating node i is one which fires in response to a wave that it is participating in. Wave w is said to *ebb* at node i whenever node i does not fire in response to wave w upon reception of the wave w message from some out-neighbour. Hence, we say that a node is *passively* participating in wave w when it is participating in wave w and wave w ebbs at node i. A node passively participating in wave w reflects wave w by emitting an echo w message. Echo messages always originate at a passively participating node and are repeated at actively participating nodes, ultimately terminating at the corresponding wave activator. The details of the depth-guided wave propagation are described below. Note that a node can be actively participating in some waves and pas-

sively participating in other waves simultaneously. Also, a node may actively or passively participate in the same wave more than once. Indeed, this is why the complexity of this technique is difficult to track. But leaving complexity analysis aside for now, it is the simplicity of this technique and its correctness which is interesting. Wave w is said to be in stage k at node i if echo w is pending on the k^{th} input edge of node i. Note that wave w can be in stage k at node i more than once. Clearly, if a node i receives a wave w message when wave w is currently in some stage k at node i, then node i has detected the presence of a directed circuit. The wave w message that node i sent out on its k^{th} input edge has been transmitted around a directed circuit back to node i on one of its output edges (i,j). Upon reception of the wave w message at node i while node i is currently participating in wave w, node i tests for the presence of a negative-length directed circuit by examining the marking on edge (i,j) after reacting to node j's request for wave w. If node i's record of the marking on edge (i,j) is nonnegative, then all is well so far and node i emits an echo w message to node j which it will later receive again as the echo w pending on its k^{th} input edge. If node i determines that its record of the marking on edge (i,j) is negative then node i has detected a negative-length directed circuit. This is true even if other waves interfere with this wave along the directed circuit around which the wave has propagated. If anything

is profound here, it is this property of Algorithm REACH. When a node i detects a negative-length directed circuit, a solution is impossible and the activity should begin to cease. This can be accomplished by flooding the network with a message informing all nodes of this condition. We shall take another approach by propagating this ill-condition information in the echo messages.

The wave initiated by node w is said to have *successfully terminated* when echo w is not pending on any input edge of w. The algorithm is said to have *terminated successfully* when all the waves have terminated successfully. If the wave w terminates successfully, then node w may broadcast this information to all other nodes. This additional broadcast from each node will keep all the nodes informed of the status of the algorithm, that is, whether or not it has terminated.

6.4 Node Protocols for Distributed REACH

In this section, we sketch an implementation of the distributed Algorithm REACH described in Section 6.3. The distributed algorithm is presented as a homogeneous collection of communicating sequential procedures which is executed by the nodes of an intelligent graph. In the presentation, we abstract away from the details of point-to-point link control by incorporating an assumed *send* and *receive* protocol pair at each node.

The order in which received messages are processed is irrelevent from the point of view of correctness. Thus, we may and will assume that messages received at a node are queued into a "received" buffer where they are processed sequentially by the node processor. We implement the protocol by having messages act as procedure invocations. An administrative procedure (which is not described here in detail) processes messages in the "received" buffer by examining a message's type and invoking the appropriate procedure. We define three types of messages, namely *wave*, *echo* and *tilt* messages, and we define three procedures for each node, namely a *wave procedure*, an *echo procedure* and a *tilt procedure*. When a node receives, say, a wave message, then it simply invokes its resident wave procedure with the parameters carried in the message. Note that a tilt message will be generated whenever infeasibility of the problem is detected.

In our protocol description, we assume that each node maintains the following data structures.

(Concurrent Depth-First-Search Version)

N is the number of nodes in the network.

Node_Id $\in \{1,2,\cdots,N\}$ is this node's identification number.

Out_Degree is the number of edges incident out of this node.

Out_Edge(k), $\forall\, k \in \{1,2,\cdots,Out_Degree\}$ identifies the k^{th} edge incident out of this node.

In_Degree is the number of edges incident into this

node.

In_Edge(k), $\forall$ k $\in$ {1,2,•••,In_Degree} identifies the k^{th} edge incident into this node.

y $\in$ {1,2,•••} is this node's accumulated firing count.

M(k) $\in$ {1,2,•••}, $\forall$ k $\in$ {1,2,•••,Out_Degree} is the current marking on the k^{th} edge incident out of this node.

Wave_Edge(w) $\in$ {1,2,•••,Out_Degree}, $\forall$ w $\in$ {1,2,•••,N} is the edge incident out of this node on which the most recent w wave message was received.

Bit(w) $\in$ {True, False} $\forall$ w $\in$ {1,2,•••,N} indicates whether or not this node is currently participating in the w wave.

To begin with, M(k), $\forall$ k $\in$ {1,2,•••,Out_Degree} is the initial marking on the k^{th} output edge.

To "ignite" the firing process, each initiation of a problem invokes a Wake_Up procedure at each node. A node must not respond to messages from other nodes until it has executed its Wake_Up procedure. Procedure Wake_Up is listed below. Procedure Wake_Up initiates a node's firing count (y ← 0) and the bit map (Bit ← (False, False, •••, False)) indicating which waves a node is participating in for a new problem and wakes up by making a transition to the Awake state. Then, the minimum marking on a node's output edges is determined. If that marking is negative then the node fires by recording y and updating the markings on its output edges. It then identifies itself as participating in its own firing wave (Bit(Node_Id) ← True), and sends a wave message carrying its node identification and the number of times it fired upon awakening along its logically-

first input edge. If a node determines that all of its output edges are nonnegatively marked, then Wake_Up terminates leaving the node in the Awake state to respond to incoming messages.

```
Procedure Wake_Up;
  Begin
    y ← 0;
    For w ← 1 to N do Bit(w) ← False;
    State ← Awake;
    min ← 0;
    For k ← 1 to Out_Degree do If M(k)<min Then min ← M(k);
    If min < 0
      Then
        Begin
          y ← -min; For k←1 to Out_Degree do M(k)← M(k) + y
          Bit(Node_Id) ← True;
          Send(Wave(Node_Id, y), In_Edge(1));
        End
  End {Wake_Up}
```

We are assuming that each node has at least one input edge and one output edge. That is, there are no sources and sinks in G. Also, G is strongly connected. Note that this is not a restriction of Algorithm REACH. We are assuming a strongly-connected topology for simplicity. We can permit any topology. If a node is a source then it can participate in waves including its own but all waves ebb at a source simply because a source represents an infinite token supply. Alternatively, a sink cannot be an activator or a participator because it has no output edges. This is consistent with our understanding of a sink as a datum of the graph. So, without loss of generality, we shall assume that each node can activate a nontrivial wave as well as participate in waves activated by other nodes. In other

words, we assume that our intelligent graph is strongly connected.

Each wave message is a packet carrying the fields Wave_Number and Delta. Wave_Number is a static field carrying the Node_Id of the node which activated the wave and is fixed for the duration of its existence. Delta is a dynamic field which indicates the number of times a node has fired in participating in the wave. We assume that the receiver appends the identification of the edge on which a wave message was received to the wave message as a third field Wave_Edge. When the administrator processes a wave message, it simply calls Procedure Wave listed below.

```
Procedure Wave(Wave_Number, Delta, Wave_Edge);
  Begin
    M(Wave_Edge) ← M(Wave_Edge) - Delta;
      If M(Wave_Edge) < 0
        Then
          If NOT Bit(Wave_Number)
            Then
              Begin
                Delta ← -M(Wave_Edge);
                For k ← 1 to Out_Degree do
                  M(k) ← M(k) + Delta;
                y ← y + Delta;
                Bit(Wave_Number) ← True;
                State ← Awake;
                Wave_Edge(Wave_Number) ← Wave_Edge;
                Delta(Wave_Number) ← Delta;
                Send(Wave(Wave_Number,Delta), In_Edge(1))
              End
            Else
              Begin
                State ← Tilted;
                If Wave_Number ≠ Node_Id
                  Then Send(Tilt(Wave_Number,Wave_Edge))
              End
        Else Send(Echo(Wave_Number), Wave_Edge)
  End {Wave}
```

Procedure Wave first decrements the marking on the edge on which the message was received and then checks whether or not this has made the marking on that edge negative. If so, then it examines the bit map to determine whether or not it is already participating in the wave. If it is not already participating in the wave, then it proceeds to participate in the wave by firing just enough to maintain nonnegativity of the marking on its output edge and identifying itself as a participant in the wave. It then records the number of times it fired in response to the wave message and the edge on which the wave message arrived for future reference and then it repeats the wave message by sending the wave message with its updated Delta field to the node adjacent to it on In_Edge(1). If Procedure Wave determines that the node is currently participating in the wave identified by Wave_Number then it has detected the presence of a negative-length directed circuit, as described in Section 6.3, and makes a transition to the Tilted state. If the wave was activated by some other node, then procedure wave sends a tilt message back to the adjacent node from which the wave message was sent along Wave_Edge.

If the marking on Wave_Edge remains nonnegative when procedure wave responds to a wave message, then that wave ebbs at the node and procedure wave sends an echo message back to the adjacent node which sent the wave message along Wave_Edge.

Echo messages are processed with Procedure Echo listed below. An echo message carries only the Wave_Number field identifying the wave associated with the echo. We assmume that the receiver appends the indentification of the edge on which the echo was received to the echo message as a second field Echo_Edge. Procedure Echo first determines whether or not the associated wave has completed all stages at the node by simply examining the Echo_Edge field of the echo message. If an echo is received on the last input In_Edge(In_Degree) then that wave has completed all stages. Otherwise, the wave associated with this echo must enter the next stage at the node.

```
Procedure Echo(Wave_Number, Echo_Edge);
  Begin
    If Echo_Edge ≠ In_Edge(In_Degree)
      Then
        Begin
          Delta ← Delta(Wave_Number);
          Send(Wave(Wave_Number,Delta),In_Edge(Echo_Edge+1))
        End
      Else
        Begin
          Bit(Wave_Number) ← False;
          If Wave_Number ≠ Node_Id
            Then
              Send(Echo(Wave_Number),Wave_Edge(Wave_Number))
            Else
              Begin
                Flag ← False;
                For w ← 1 to N do Flag ← Flag OR Bit(w);
                If NOT Flag Then State ← Finished
              End
        End
  End {Echo}
```

When Procedure Echo determines that a wave must enter the next stage it recalls the number of times that it fired

for that wave (Delta(Wave_Number)) and sends a wave message carrying this recalled information to the adjacent node along the logically-next input edge In_Edge(Echo_Edge + 1). When Procedure Echo determines that a wave has completed all stages at the node, then it identifies itself as not participating in the wave (Bit(Wave_Number) ← False) and if it is not the activator of the wave, then it sends an echo message carrying the associated wave number to the adjacent node along the edge upon which this wave message was received (Wave_Edge(Wave_Number)). Otherwise, the wave terminates and it makes a transition to the Finished state.

Tilt messages carry only the identification of the wave which resulted in the detection of a negative-length directed circuit. Upon invocation, Procedure Tilt makes a transition to the Tilted state and if it is not the activator of the associated wave, then it simply sends the tilt message to the adjacent node along the edge on which the wave message arrived (Wave_Edge(Wave_Number)). Thus, instead of pursuing the remaining stages of a tilted wave, the tilt message simply propagates to the activator of the tilted wave where it terminates.

```
Procedure Tilt(Wave_Number);
  Begin
    State ← Tilted;
    If Wave_Number ≠ Node_Id
      Then Send(Tilt(Wave_Number),Wave_Edge(Wave_Number))
  End {Tilt}
```

Algorithm REACH terminates when all nodes reach the

Finished state or when any node reaches the Tilted state and informs all the others.

6.5 Summary

In this chapter, we have given a distributed version of Algorithm REACH of Chapter 5. We have described the different node protocols which constitute Distributed REACH.

A number of distributed algorithms for several graph problems have been studied in the literature. Among these is the problem of computing a maximum flow in a transport network which arises as a subproblem in several network optimization problems. Whereas the maximum-flow problem is useful in testing the feasibility of the transshipment problem, the submarking-reachability problem is useful in testing the feasibility of the dual transshipment problem. It is in this context that Distributed REACH, presented in this chapter, derives its importance. An important open problem is to study the message complexity of this algorithm. We expect further study of this algorithm will throw much light on distributing the simplex method for the dual transshipment problem.

Chapter 7

REACHABILITY AND SEQUENCING IN STATE GRAPHS

In this chapter, we study certain problems on state graphs dual to some of those considered in the previous chapters on marked graphs. More specifically, we give a reachability theorem for state graphs. We study the extreme-execution problem. We also introduce and study the subsequence-executability problem.

7.1 Reachability in State Graphs

Recall that a ***finite-state graph***, is a directed graph $G = (V,E)$ with vertex set V, edge set E, a nonnegative-integer state vector M, associated with V, called the ***marking*** or ***token distribution*** of G, and a ***state-transition function*** $\delta_e(M)$, mapping M into a new marking M', resulting from firing edge $e \in E$. Vertices of G are called ***places*** or ***states*** and edges of G are called ***transitions***. The transition function removes one token from the initial place of edge e and puts one token on the terminal place of edge e. Since the resulting marking must be nonnegative, the only enabled transitions under a marking M are those whose initial places contain at least one token. Thus, tokens simply move about from place to place and, therefore, the marking of a finite-state graph is ***conservative***. The finite-state graph is equivalent to a closed queueing

network with a single customer class circulating within it.

In this section, we first show that the set of all firing-count vectors between an initial marking and a reachable final marking is characterized by a system of network equations. We then define the *extreme-execution problem* and demonstrate its equivalence to the *transshipment problem* of operations research.

7.1.1 Reachability and Extreme Executions

Let A denote the incidence matrix of G, Σ denote a *firing-count vector* of the edges of G, whose components record the numbers of times the corresponding transitions fire, and M_o be an initial marking of the vertices of G. A firing-count vector Σ is considered *executable* from a marking M_o if and only if there exists a legal sequence of firings from M_o, whose firing-count vector is Σ. Every executable Σ from M_o results in a marking M defined by the state equation

$$M = M_o - A\Sigma. \tag{7.1}$$

The marking M, then, is said to be *reachable* from the initial marking M_o. Letting Δ_M denote the differential marking $M - M_o$, we obtain the equations characterizing the set of all firing-count vectors transforming M_o into M, namely

$$A\Sigma = -\Delta_M \tag{7.2}$$

As an example, a state graph with two markings M_o and M_1 is shown in Figure 7.1. For this example, $-\Delta_M = M_o - M_1 =$

$$\begin{matrix} a & b & c & d & e \\ [3 & 2 & -1 & -2 & -2]. \end{matrix}$$

Equation 7.2 is a mathematical statement of the intuitive observation that transition firings in a state graph behave like a flow. In fact, viewing the state-transition process in a state graph as a token flow provides good insight into the execution properties of finite-state graphs. We will pursue this approach in what follows.

Let $n = |V|$ and let $m = |E|$. The system of equations in (7.2) is a dependent one since each equation in this system can be expressed as minus the sum of the remaining n-1 equations. The remaining n-1 equations then constitute an independent system. Hence, we need only delete one equation from (7.2) to obtain an independent system. However, before deleting an equation, we must ascertain an equivalent dependence in the right-hand side. If we sum all equations in (7.2), the left-hand side vanishes. Clearly, consistency requires that the right-hand side must also vanish. This is always the case in the finite-state graph since its marking is conservative. In other words, G can neither gain tokens nor lose tokens and, hence, the net token increase at any place in G is equal to the net token decrease in the rest of G. Thus, in viewing a finite-state graph as a token-flow network, we interpret places with a

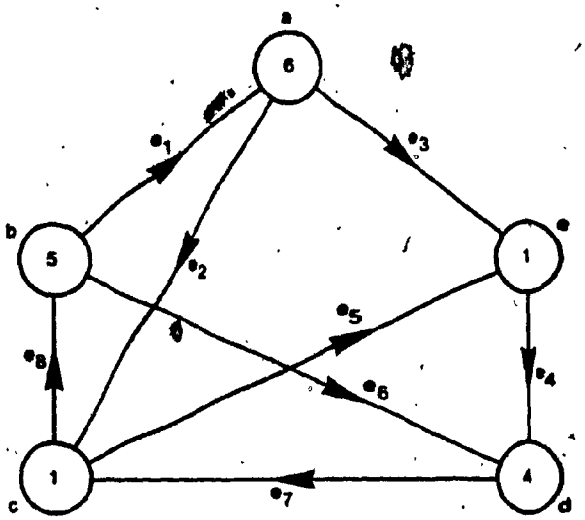


Figure 7.1 (a)

A state graph with marking M_o

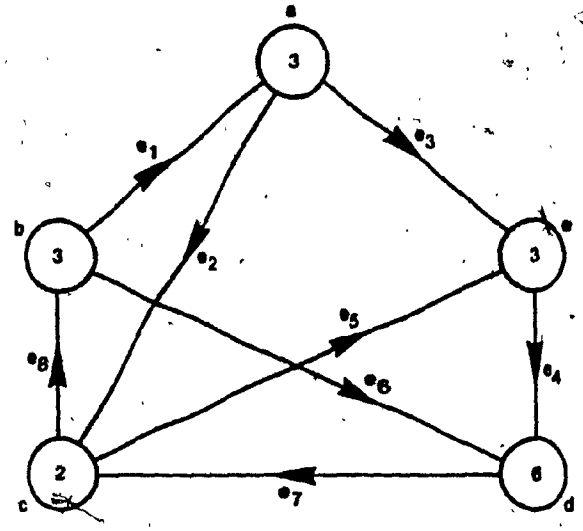


Figure 7.1(b)

A state graph with marking M_1

positive, zero, or negative Δ_M entry as a *token sink*, *intermediate place*, or *token source*, respectively. Conservation ensures that the total token supply equals the total token demand. Hence, let us assume that this condition holds for Δ_M and choose any vertex as a reference, delete the corresponding row from the incidence matrix A and denote the resulting *truncated incidence matrix* of G with the same symbol A. Also, let Δ denote the differential marking of the remaining n-1 places obtained by deleting the reference entry from Δ_M.

We may associate a per unit *cost* or *weight* W(e) with each edge $e \in E$, representing the cost of firing transition e once. In formulating the problem, we are assuming that the same cost W(e) is incurred each time transition e fires. Under this assumption, clearly, the expression $\sum_e W(e)\Sigma(e)$ represents the cost of executing the firing-count vector Σ. The *extreme-execution* problem is to determine a firing-count vector Σ which has minimum or maximum cost, among all such vectors leading G from an initial marking M_o to a final marking M. Note that $\sum_e W(e)\Sigma(e)$ is not, in general, the total execution time since this expression cannot account for concurrency among firings. In the sequential case of the state graph, namely the *state machine* (a state graph with a single token), the objective $\sum_e W(e)\Sigma(e)$ is exactly the execution time for any execution of Σ from M_o when each transition $e \in E$ takes W(e)

seconds to fire, every time it does.

Let W be a row vector of *transition weights*. We may now state the extreme-execution problem for a state graph, G as the equivalent linear program

$$\begin{aligned} &\text{minimize or maximize } W\Sigma \\ &\quad \text{subject to } A\Sigma = -\Delta, \\ &\qquad \Sigma \geq 0. \end{aligned} \tag{7.3}$$

The minimization problem in (7.3) is equivalent to the transshipment problem or minimum-cost single-commodity flow problem of operations research. This is exactly dual to the extreme-marking problem of Chapter 3, for a marked graph, and the network simplex method may be used to solve this problem.

The reachability problem for finite-state graphs reduces to the problem of obtaining a feasible solution to the constraints in (7.3). The justification for this statement follows easily from the token-flow network interpretation. That is, if a feasible solution to the constraints in (7.3) does not exist, then clearly M is not reachable from M_0. On the other hand, if a feasible solution does exist, then the answer to the reachability question is in the affirmative since a flow schedule satisfying the transshipment constraints is always implementable. Thus, we state, without proof, the reachability theorem for state graphs. Proof follows from the network simplex method [70].

Theorem 7.1 (Reachability Theorem for State Graphs): A marking M is reachable from an initial marking M_o of a state graph G = (V,E) if and only if G does not contain a directed cutset $(S,\bar{S})$, $S \subseteq V$, with

$$\sum_{i \in S} M(i) \quad) \quad \sum_{i \in S} M_o(i). \blacksquare \tag{7.4}$$

7.1.2 Basic Legal Executions

Let T and $\bar{T}$ denote a spanning tree of G and its corresponding cospanning tree, respectively. Arrange the columns of A so that the first n-1 columns correspond to the branches of T and the remaining m - n + 1 columns to the chords of $\bar{T}$. Arrange W and Σ in the same fashion. Then, Program 7.3 partitions into the following problem

$$\begin{aligned} &\textbf{minimize or maximize } W\Sigma = W_T\Sigma_T + W_{\bar{T}}\Sigma_{\bar{T}} \\ &\quad \textbf{subject to } A_T\Sigma_T + A_{\bar{T}}\Sigma_{\bar{T}} = -\Delta, \\ &\qquad \Sigma = (\Sigma_T^t, \Sigma_{\bar{T}}^t)^t \geq 0. \end{aligned} \tag{7.5}$$

The matrix A_T is nonsingular, has a triangular permutation and is unimodular. We may solve the matrix constraint in (7.5) for the tree variables in terms of the cospanning variables. Thus, dual to the extreme-marking problem, in the extreme-execution problem the spanning-tree variables become the ***basic variables*** and the cospanning-tree variables become the ***nonbasic variables***. First, multiplying both sides of the matrix constraint in (7.5) by A_T^{-1} leads to an equivalent canonical statement of this constraint, namely

$$A_T^{-1}A_T\Sigma_T + A_T^{-1}A_{\bar{T}}\Sigma_{\bar{T}} = -A_T^{-1}\Delta, \qquad (7.6)$$

$$U_T\Sigma_T + Q_f{}_{\bar{T}}\Sigma_{\bar{T}} = -\Delta_T, \qquad (7.7)$$

$$Q_f\Sigma = -\Delta_T, \qquad (7.8)$$

where Q_f is the fundamental-cutset matrix of G with respect to the spanning tree T. We shall call $\Delta_T \triangleq A_T^{-1}\Delta$ the ***fundamental differential marking*** with respect to T. Each entry in Δ_T is the differential marking of a subset of places defined by the corresponding branch in T. Equation 7.8 is simply a transformed version of the matrix constraint $A\Sigma = -\Delta$. Solving (7.7) for the tree variables in terms of the cospanning variables yields the ***dictionary***

$$\Sigma_T = -\Delta_T - Q_f{}_{\bar{T}}\Sigma_{\bar{T}}. \qquad (7.9)$$

A ***basic legal execution*** of a finite-state graph is any legal transition-firing sequence, leading from an initial marking to a reachable final marking, in which all firings are confined to the branches of some spanning tree of the graph. Let us call the subgraph of G consisting of all and only those edges which fire in a legal execution, the ***executed subgraph***. Thus, the executed subgraph of a basic legal execution is a forest.

Now, if $\Delta_T \leq 0$, then we get a basic feasible solution to the constraints in (7.5) by simply setting $\Sigma_{\bar{T}} = 0$ in the Dictionary 7.9 to obtain the tree solution $\Sigma_T = -\Delta_T$. The set of all legal firing sequences executing this basic feasible firing-count vector constitutes a class of basic

legal executions of the finite-state graph with respect to the spanning tree T. If $\Delta_T \not\leq 0$, then we are faced with the problem of obtaining a basic feasible solution to (7.5). That is, we must determine a spanning tree T of G for which $A_T^{-1}\Delta \leq 0$. This initialization problem can be formulated as a similar transshipment problem and, as such, may be solved in the same fashion as the original problem is solved. Thus, we will assume that a basic feasible solution is at hand.

Returning to the markings M_0 and M_1 in Figure 7.1, a basic feasible solution for (7.5) is given as

$$\Sigma(e_1) = 2,$$
$$\Sigma(e_2) = 1,$$
$$\Sigma(e_3) = 4,$$
$$\Sigma(e_4) = 2.$$

Firing the edges e_1, e_2, e_3 and e_4 in this order, the required number of times, takes G from M_0 to M_1.

Substituting the basic dictionary for the tree variables in the objective of (7.5), we get

$$\begin{aligned} W\Sigma &= W_T(-\Delta_T - Q_{+\bar{T}}\Sigma_{\bar{T}}) + W_{\bar{T}}\Sigma_{\bar{T}}, \\ &= -W_T\Delta_T + (W_{\bar{T}} - W_TQ_{+\bar{T}})\Sigma_{\bar{T}}, \\ &= -W_T\Delta_T + \tilde{W}_{\bar{T}}\Sigma_{\bar{T}}. \end{aligned} \tag{7.10}$$

The development from this point on is exactly dual to that for the extreme-marking problem and, thus, it will not be

elaborated upon here, except to highlight the dual concepts. First, we note that in this problem, $\tilde{W}_{\bar{T}}$ is the *relative-cost* vector of the cospanning tree $\bar{T}$ and $-W_T\Delta_T$ is the cost of our basic feasible solution. Hence, we have the following definition and theorem.

The *cost* or *weight* of a fundamental circuit C is the algebraic sum of the weights in C, when traversed in the direction of its defining chord. In other words, the circuit weight is the sum of the forward weights minus the sum of the backward weights.

Theorem 7.2: The relative chord costs are the fundamental-circuit weights.■

The proof of this theorem can be established in the same fashion as the proof of the dual theorem, namely Theorem 3.4, for marked graphs.

The important implication of the network-flow interpretation of the extreme-execution problem is that there exists a basic legal execution of a finite-state graph between an initial marking and any reachable final marking. Also, note that in a basic legal execution, firings are all confined to the branches of a spanning tree.

We conclude this section by again pointing out that the extreme-execution problem for the state graph is dual to the extreme-marking problem for the marked graph. Fur-

thermore, whereas in the case of a marked graph no more than (n-1) firing counts are nonzero in the minimum nonnegative executable firing-count vector, in the case of a state graph, no more than (n-1) edges need be fired to reach any reachable state from any initial state.

7.2 The Subsequence-Executability Problem

In the *subsequence-executability problem*, we are given a finite-state graph with an initial marking M_o and a reachable final marking M. The firing count Σ_c is specified on a subset of edges E_c called the *controlled set*. The remaining edges $E - E_c$ constitute the *free set* E_F. The problem is to determine the minimum firing-count vector of E, if it exists, leading from M_o to M which fires the controlled set E_c as prescribed by Σ_c.

As in the submarking-reachability problem, the incidence matrix A of G partitions into

$$A = \begin{array}{c} \begin{matrix} E_c & E_F \end{matrix} \\ \begin{bmatrix} A_{cc} & A_{cF} \\ 0 & A_{FF} \end{bmatrix} \begin{matrix} V_c \\ V_F \end{matrix} \end{array}$$

and, hence, the state equation becomes

$$A_{cc}\Sigma_c + A_{cF}\Sigma_F = -\Delta_c, \qquad (7.11)$$

$$A_{FF}\Sigma_F = -\Delta_F, \qquad (7.12)$$

where the notation should be obvious. Since Σ_c is specif-

ied, we must then, obtain a solution to the following linear program, if one exists,

$$\begin{aligned} &\text{minimize } W_F\Sigma_F \\ &\quad \text{subject to } A_{CF}\Sigma_F = -\Delta_C - A_{CC}\Sigma_C, \\ &\qquad\qquad A_{FF}\Sigma_F = -\Delta_F, \\ &\qquad\qquad \Sigma_{F_{\oplus}} \geq 0. \end{aligned} \tag{7.13}$$

The solution to this problem may be obtained by reducing the finite-state graph in a dual fashion to that used to reduce the submarking-reachability problem. We fire the controlled edges E_C by the prescribed number of times and take G to the not necessarily feasible intermediate state $\tilde{M}$, defined by the transformation

$$\tilde{M}_C = M_{Co} - A_{CC}\Sigma_C \tag{7.14}$$

$$\tilde{M}_F = M_{Fo}. \tag{7.15}$$

Clearly, from state $\tilde{M}$, edges in E_C must not be fired in reaching M since they have been completely fired in moving to $\tilde{M}$. Therefore, in the state $\tilde{M}$, we may open-circuit all controlled edges and remove them from G. Also, the differential marking associated with the controlled places V_C is modified to $\tilde{\Delta}_C = \Delta_C + A_{CC}\Sigma_C$. The differential marking of the free places does not change, i.e., $\tilde{\Delta}_F = \Delta_F$. So, the subsequence-reachability problem reduces to the following linear program

$$\begin{aligned} &\text{minimize } W_F\Sigma_F \\ &\quad \text{subject to } \tilde{A}\Sigma_F = -\tilde{\Delta}, \\ &\qquad\qquad \Sigma_F \geq 0, \end{aligned} \tag{7.16}$$

where $\tilde{A}$ is the incidence matrix of the reduced graph, $\tilde{\Delta}$ is the *intermediate differential marking* and W_r is the cost vector associated with the free transition firings. Program 7.16 is a transshipment problem. Thus, the subsequence-executability problem reduces to the extreme-execution problem which may be solved by the network simplex method.

7.3 Summary

In this chapter we have presented formulations of certain problems on state graphs which highlight the duality between the state graph and the marked graph. We have shown that the reachability problem on the state graph is equivalent to the problem of testing feasibility of a transshipment problem. We have defined the extreme-execution problem for a state graph and formulated it as a transshipment problem. This formulation has led us to define a basic legal execution and establish that a state graph can be taken from one state to a reachable state by firing only the edges of an appropriate tree. Finally, we have defined the subsequence-executability problem and have shown that this problem is also equivalent to a transshipment problem.

Chapter 8

SUMMARY AND PROBLEMS FOR FURTHER STUDY

In this chapter, we summarize the main contributions of this thesis and point out a few problems for future study.

8.1 Summary

In this thesis, we have been concerned with an algorithmic study of certain issues relating to the reachability and sequencing problems for marked graphs and state graphs. Most of the results presented in this thesis are based on linear-programming formulations of the relevant problems. Our study of problems related to marked graphs has been presented in Chapters 2 to 6. We have studied state graphs in Chapter 7.

In Chapter 2, we first gave an algorithmic proof to the reachability theorem on marked graphs. We have also extended this proof to cover the capacitated case. We then introduced the concept of scatter in a firing sequence. Using the notion of a greedy firing policy, we have presented algorithms for generating minimum-scatter firing sequences for different classes of graphs. More specifically, we have considered three cases: acyclic directed graphs, directed cirtuits and graphs in which all directed circuits are vertex disjoint. We have pointed out that in

the general case this problem reduces to that of determining a minimum-scatter firing sequence for a strongly-connected graph. Finally, we have presented a purely graph-theoretic characterization of the reachability problem on (0,1)-capacitated marked graphs. The relationship between this work and the results presented in [66] and [67] has been pointed out. This relationship suggests the possibility of designing efficient distributed algorithms for the reachability problem on (0,1)-capacitated marked graphs.

In Chapter 3, we first presented a linear-programming formulation of the maximum-weight marking problem on live, marked graphs. We have described the details of an algorithm (based on the simplex method) to obtain a maximum-weight marking. The concepts of a basic marking and diakoptic firings have been defined. We have shown that each pivot in the simplex method corresponds to a diakoptic firing. An algorithm requiring only vertex firings has been given to construct a basic feasible marking reachable from a given initial marking. In addition to constructing a maximum-weight marking, our algorithm constructs a firing sequence leading from the initial marking to a maximum-weight marking. We have also established the diakoptic reachability theorem. We then presented details of an algorithm to construct a maximum-weight marking in the case of capacitated marked graphs. Finally, a formulation of the problem has been given in terms of firing counts only.

Using this formulation, we have studied the maximum-weight marking problem for the nonlive class of problems. We have shown that the maximum-weight marking problem has the same structure in the cases of both live and nonlive graphs.

One important advantage of the linear-programming formulation of a problem is that it makes sensitivity analysis of the problem easy. Thus, our formulations in Chapter 3 would facilitate the study of the effects of small changes in the initial markings on the optimal solution. We have also pointed out, in Chapter 3, that the problem of determining the maximum resource requirements in the computation graph model of Karp and Miller [3] reduces to the maximum-weight marking problem in the case where the input and the output quanta as well as the threshold of each edge of the computation graph are equal.

In Chapter 4, we studied the structure of the submarking-reachability problem on marked graphs. Using a linear-programming formulation, we have presented an approach which exposes the structure of the problem and unifies the study of this problem for both capacitated and uncapacitated graphs. We have shown that the submarking-reachability problem reduces to the problem of taking a graph from an infeasible marking to a feasible marking through a sequence of, possibly illegal, vertex firings.

In Chapter 5, we first pointed out the equivalence

between the submarking-reachability problem and the problem of testing feasibility of the dual transshipment problem. We then gave an algorithm, called Algorithm REACH, to solve this feasibility testing problem and, hence, the submarking-reachability problem. Proof of correctness and termination of this algorithm have been presented. We have shown that the complexity of the algorithm is O(mn), where m is the number of edges and n is the number of vertices in the given graph. This is in contrast to the complexity $O(n^3)$ of the well-known MPM algorithm [71] which can be used to test the feasibility of the transshipment problem.

In Chapter 6, we distributed Algorithm REACH. We presented several procedures constituting this distributed algorithm. The place of this distributed algorithm in the context of distributed algorithms for other graph problems as well as the possibilities it opens up for distributing general network optimization problems have been pointed out.

In Chapter 7, we studied certain problems on state graphs dual to some of those discussed in the earlier chapters on marked graphs. We have shown that the reachability problem in the case of state graphs reduces to the problem of testing feasibility of the transshipment problem. We have defined the extreme-execution problem for a state graph and formulated it as a transshipment problem. This formulation has led us to define a basic legal

execution and establish that a state graph can be taken from one state to a reachable state through a sequence of firings which are all confined to the edges of an appropriate spanning tree. Finally, we defined the subsequence-reachability problem and showed that it is also equivalent to the transshipment problem.

8.2 Problems For Further Study

We now point out a few problems which require further study. These problems arise as a result of our investigations in this thesis.

1. In Chapter 1, we have given polynomial algorithms to design minimum-scatter firing sequences for certain classes of marked graphs. We have shown that in the general case the problem reduces to that of determining minimum-scatter sequences for strongly-connected graphs. It appears that the general problem is of exponential complexity. An important open problem is to determine whether a polynomial algorithm exists for the minimum-scatter problem in the case of strongly-connected graphs and, if not, prove that the problem is NP-Complete.

2. We have shown in Chapter 1 that the reachability problem on (0,1)-capacitated marked graphs is equivalent to the problem of transforming a given acyclic graph into another one using a sequence of source vertex firings. An easy problem in this context is to determine and characterize

the exact sequence of vertex firings which might transform one acyclic graph into another. It seems that such a sequence can be constructed using a depth-first-search of the given graph. We have also pointed out in Chapter 2 that similar problems, possibly less general ones, have arisen in seemingly different contexts [66] and [67]. In [67] a distributed algorithm has been given to transform a destination-disoriented acyclic graph to a destination-oriented acyclic graph. This work has been motivated by an application to a routing problem in computer communication networks. A related and interesting open problem is to design a distributed algorithm to transform an acyclic graph into another one using appropriate vertex firing protocols. It appears that study of this problem along the lines of [67] might provide considerable insight into this problem and also prove useful in generalizing the results in [67].

3. Avoiding the cycling phenomenon which results from degeneracy is an important problem in linear programming. Several anticycling rules have been reported in the literature. For the transshipment problem, a very ingenious anticycling rule due to Cunningham [70] which takes advantage of the structure of the problem is now available. However, such a simple rule is not available for the dual transshipment problem. Thus, an important open problem is to devise an elegant anticycling rule for the dual transshipment problem. A solution to this problem will be a major funda-

mental contribution to network optimization theory.

4. Computing a maximum flow in a transport network is a very important problem in operations research. Its importance is due to the fact that this problem arises as a subproblem in several network optimization problems. Also, several graph problems can be shown to be equivalent to the maximum-flow problem on (0,1)-capacitated transport networks. In fact, as we have pointed out earlier in Chapters 5 and 6, testing feasibility of a transshipment problem reduces to computing a maximum flow in an appropriate transport network. In view of these general applications, the maximum-flow problem has attracted considerable attention in the literature. The MPM algorithm [71] for this problem is known to have the best complexity of $O(n^3)$. Current efforts are to obtain efficient distributed protocols for this problem. Algorithm REACH, presented in Chapter 5, is of complexity $O(mn)$ and can be used to test feasibility of the dual transshipment problem. The distributed version of this algorithm has been presented in Chapter 6. An important problem which requires further study is to determine the message complexity of this distributed algorithm. A further study of this problem is expected to provide more insight into the nature of the dual transshipment problem which has not received much attention in the operations research literature.

REFERENCES

[1] J. L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981.

[2] J. L. Peterson, "Petri nets", Computing Surveys, Vol. 9, pp. 223-252, 1977.

[3] R. M. Karp and R. E. Miller, "Properties of a model for parallel computations: determinacy, termination and queueing", SIAM J. Appl. Math., Vol. 14, pp. 1390-1411, 1966.

[4] F. Commoner, A. W. Holt, S Even and A. Pneuli, "Marked directed graphs" J. Computer and System Science, Vol. 5, pp. 511-523, 1971.

[5] T. Murata, "Circuit-theoretic analysis and synthesis of marked graphs", IEEE Trans. Circuits and Systems, Vol. CAS-24, pp. 400-405, 1977.

[6] S. Kumagai, S. Kodama and M. Kitagawa, "Submarking reachability of marked graphs", IEEE Trans. Circuits and Systems, Vol. CAS-31, pp. 159-164, 1984.

[7] S. Kumagai, S. Kodama and M. Kitagawa, "Submarking reachability of marked graphs with token capacity constraints", Trans. IECE, Japan, Vol. E67, pp. 373-378, 1984.

[8] T. Murata, "State equation, controllability and maximal matchings of Petri nets", IEEE Trans. Automatic Control, Vol. AC-22, pp. 412-416, 1977.

[9] Y. Kajitani and T. Kawakami, "Subgraph reachability of marked graphs", Proc. 18th Annual Allerton Conf. on Communication, Control and Computing, pp. 507-513, 1980.

[10] S. Kumagai, S. Kodama, T. Naito and T. Sawai, "On the structural properties of extended marked graphs", Proc. IEEE International Symposium on Circuits and Systems, pp. 1386-1389, 1984.

[11] K. Tsuji, S. Kumagai, S. Kodama and S. Takeda, "On the liveness of extended marked graphs", Proc. IEEE International symposium on Circuits and Systems, pp. 471-479, 1985.

[12] K. Onaga, "Analysis of cyclic behaviours of multi-PERT Petri nets via extended marked graphs", Proc. IEEE International Symposium on Circuits and Systems, pp. 475-478, 1985.

[13] T. Murata and J. Y. Koh, "Reduction and expansion of live and safe marked graphs", IEEE Trans. Circuits and Systems, Vol. CAS-27, pp. 68-70, 1980.

[14] R. Johnsonbaugh and T. Murata, "Additional methods for reduction and expansion of marked graphs", IEEE Trans. Circuits and Systems, Vol. CAS-28, pp. 1009-1014, 1981.

[15] R. Vallette, "Analysis of Petri nets by step-wise refinements", J. Computer and System Science, Vol. 18, 35-46, 1979.

[16] I. Suzuki and T. Murata, "A method for step-wise

refinement and abstraction of Petri nets", J. Computer and Systems Science, Vol. 27, pp. 51-76, 1983.

[17] R. Reiter, "Scheduling parallel computations", J. Assoc. Compu. Machinery, Vol. 15, pp. 590-599, 1968.

[18] A. T. Amin and T. Murata, "A characterization of live and safe markings of a marked directed graph", Proc. Conference on Info. Sci. and Syst., John Hopkins University, pp. 295-299, 1976.

[19] T. Murata, "Petri nets, marked graphs and circuit-system theory", IEEE Circuits and Systems Society, Newsletter, Vol. 11, pp. 2-12, 1977.

[20] L. H. Landweber and E. L. Robertson, "Properties of conflict-free and persistent Petri nets", J. Assoc. Comp. Machinery, Vol. 25, pp. 352-364, 1978.

[21] T. Agerwala, "Putting Petri nets to work", Computer, Vol. 12, pp. 85-94, 1979.

[22] W. Brauer, B. Randell and C. A. Petri, (Editors), Net Theory and Applications, Lecture Notes in Computer Science, Vol. 84, Springer-Verlag, New York, 1980.

[23] T. Murata, "Relevance of network theory to models of distributed/parallel processing", J. Franklin Institute, Vol. 310, pp. 41-50, 1980.

[24] H. J. Genrich, K. Lautenbach and P. S. Thiagarajan, "Elements of general net theory" in Net Theory and Applications, W. Brauer et al. (Editors), Lecture Notes in Computer Science, Vol. 84, Springer-Verlag, pp. 21-164, 1980.

[25] C. Andre, M. Diaz, C. Girault and J. Sifakis, "Survey of French research and applications based on Petri nets", in W. Brauer et al. (Editors), Net Theory and Applications, Lecture Notes in Computer Science, Vol. 84, Springer-Verlag, New York,, pp. 321-346, 1980.

[26] E. W. Mayr, "An algorithm for the general Petri net reachability problem", in Proc. 13th Annual ACM Symp. on Theory of Computing, pp. 238-246, 1981.

[27] J. L. Johnson and T. Murata, "On the maximum number of concurrently enabled vertices in a marked graph", Proc. IEEE International Symposium on Circuits and Systems, pp. 806-809, 1981.

[28] K. Jensen, "Colored Petri nets and the invariant method", Theoretical Computer Science, Vol. 14, pp. 317-336, 1981.

[29] J. L. Johnson and T. Murata, "Conflict, confluence and precedence matrices for Petri nets", Proc. 25th Midwest Symp. on Circuits and Systems, pp. 230-234, 1982.

[30] Z. Wu and T. Murata, "Fair Petri nets and weighted synchronic distance", Proc. 26th Midwest Symp. on Circuits and Systems, pp. 129-133, 1983.

[31] M. A. Comeau and K. Thulasiraman, "Algorithms on marked directed graphs", Canadian Electrical Engg. Journal, Vol. 9, pp. 72-79, 1984.

[32] Di-Jen Leu and T. Murata, "Properties and applications of the token distance matrix of a marked graph", Proc.

IEEE International Symp. on Circuits and Systems, pp. 1381-1385, 1984.

[33] K. Thulasiraman and M. A. Comeau, "Maximum markings in marked graphs: An algorithm and interpretations based on the network simplex method", Proc. IEEE International Sym.on Circuits and Systems, pp. 479-482, 1985.

[34] J. L. Johnson and T. Murata, "Structure matrices for Petri nets and their applications", J. Franklin Institute, pp. 299-309, 1985.

[35] T. Murata and Z. Wu, "Some considerations on synchronic distances in a Petri net", Proc. IEEE International Symp. on Circuits and Systems, pp. 483-486, 1985.

[36] T. Murata, "Synthesis of decision-free concurrent systems for prescribed resources and performance", IEEE Trans. Software Engineering, Vol. SE-6, pp. 525-530, 1980.

[37] T. Murata, Van Ban Le and Di-Jen Leu, "A method for realizing the synchronic distance matrix as a marked graph", Proc. IEEE International Symp. on Circuits and Systems, pp. 1-4, 1982.

[38] A. Datta and S. Ghosh, "Synthesis of a class of deadlock free Petri nets",J. Assoc. Computing Machinery, Vol. 31, pp. 486-506, 1984.

[39] J. R. Jump, "Asynchronous control arrays", IEEE Trans. Computers, Vol. C-23, pp. 1020-1029, 1974.

[40] P. M. Merlin, "A methodology for the design and

implementation of communiction protocols", IEEE Trans. Communications, Vol. COM-24, pp. 614-621, 1976.

[41] P. M. Merlin and D. J. Farber, "Recoverability of communication protocols: Implications of a theoretical study", IEEE Trans. Communications, Vol. COM-24, pp. 1036-1043, 1976.

[42] J. L. Baer and C. S. Ellis, "Model, design and evaluation of a compiler for a parallel processing environment", IEEE Trans. Software Engg., Vol. SE-3, pp. 394-405, 1977.

[43] J. Sifakis, "Use of Petri nets for performance evaluation", in Beilner and Gelenbe (Editors), Measuring, Modeling and Evaluation of Computer Systems, North-Holland, Amsterdam, pp.75-93, 1977.

[44] C. V. Ramamoorthy and H. H. So, "Software requirements and specifications: Status and perspectives",in Tutorial: Software Methodology, IEEE Catalog No. EHO-142-0, pp. 43-164, 1978.

[45] J. Sifakis, "Realization of fault-tolerant systems by coding Petri nets", J. design Automat. Fault-Tolerant Comput., Vol. 3, pp. 93-107, 1979.

[46] C. V. Ramamoorthy and G. S. Ho,"Performance evaluation of concurrent asynchronous systems using Petri nets", IEEE Trans. Software Engg., pp. 440-449, 1980.

[47] L. J. Mekly and S. S. Yau,"Software design representation using abstract process networks", IEEE Trans. Software Engg., Vol. SE-6, pp. 420-435, 1980.

[48] K. Voss, "Using predicate/transition nets to model and analyse distributed database systems", IEEE Trans. Software Engg., Vol. SE-6, pp. 539-544, 1980.

[49] W. E. Kluge and K. Lautenbach, "The orderly resolution of memory access conflicts", IEEE Trans. on Computers, Vol. C-31, pp. 194-207, 1982.

[50] M. Sowa and T. Murata, "A dataflow computer architecture with program and token memories", IEEE Trans. Computers, Vol. C-31, pp. 820-824, 1982.

[51] M. R. Malloy, "Performance analysis using stochastic Petri nets", IEEE Trans. Computers, September, 1982.

[52] C. Girault and W. Reisig (Editors), Applications and Theory of Petri Nets, Informatik-Fachberichte, No. 52, Springer-Verlag, Berlin, 1982.

[53] Carol A. Niznik and Peter Loh, "A relational Petri net model for reduced computational complexity in routing table up-dating", Proc. ICC, pp. 63.4.1-63.4.5, 1981.

[54] Z. Wu and T. Murata, "A Petri net model of a starvation-free solution to the dining philosophers' problem", in IEEE Workshop on Languages for Automation, pp. 192-195, 1983.

[55] T. Murata, "Modeling and analysis of concurrent systems", in Handbook of Software Engg., C. R. Vick and C. V. Ramamoorthy, Van Nostrand, New York, 1984.

[56] Ajmone Marsan, M. G. Balbo and G. Conte, "A class of generalized stochastic Petri nets for the performance

evaluation of multi-processor systems", ACM Trans. Comp. Systems, Vol. 2, pp. 93-122, 1984.

[57] S. M. Shatz and W. K. Cheng, "Static analysis of ADA programs using the Petri net model", Proc. IEEE International Symposium on Circuits and Systems, pp. 719-722, 1985.

[58] R. Kujansuu, M. Lindqvist, L. Ojla and M. Tiusanen, "Petri net based production environment supporting software production", Proc. IEEE International Symp. on Circuits and Systems, pp. 727-730, 1985.

[59] A. K. Datta, S. Ghosh and D. Harms, "An analytical model for resource sharing problems in concurrent systems", Proc. IEEE International Symposium on Circuits and Systems, pp. 731-734, 1985.

[60] H. Ammar, F. Y. Huang and R. W. Liu, "Analysis by aggregation of the generalized stochastic Petri nets with applications to reachability/maintainability and fault diagnosis", Proc. IEEE International Symposium on Circuits and Systems, pp. 743-746, 1985.

[61] A. Pagnoni, "Planning, evaluation, dynamic control of complex activities", Proc. IEEE International Symposium on Circuits and systems, pp. 735-738, 1985.

[62] M. Ajmone Marsan and G. Chiola, "Modeling discrete event systems with stochastic Petri nets", Proc. IEEE International Symposium on Circuits and Systems, pp. 739-742, 1985.

[63] R. Vallette, M. Courvoisier, H. Demmou, J. M. Bigou

and C. Desclaux, "Putting Petri nets to work for controlling flexible manufacturing systems", Proc. IEEE International Symposium on Circuits and Systems, pp. 929-932, 1985.

[64] N. Kodama, Tomohiro Murata and K. Matsumoto, "Petri net based conroller: SCR and its application in factory automation", Proc. IEEE International Symposium on Circuits and Systems, pp. 937-940, 1985.

[65] H Alayan and R. W. Newcomb, "Petri net models for robot networks", Proc. IEEE International Symp. Circuits and Systems, pp. 996-999, 1986.

[66] K. Thulasiraman, M. G. G. Naidu and P. S. Reddy, "Similarity of graphs and enumeration of dissimilar n^{th} order symmetric patterns", Canadian Electrical Engineering Journal, pp. 9-14, 1985.

[67] E. M. Gafni and D. P. Bertsekas, "Distributed algorithms for generating loop-free routing in networks with frequently changing topology", IEEE Trans. Communications, Vol. COM-29, pp. 11-18, 1981.

[68] M. Mesures, Patricia A. Carr and Bruce D. Shriver, "A distributed operating system kernel based on dataflow principles", IEEE, 1982.

[69] M. N. S. Swamy and K. Thulasiraman, Graphs, Networks and Algorithms, Wiley-Interscience, 1981.

[70] V. Chvátal, Linear Programming, W. H. Freeman and Company, New York, 1983.

[71] V. M. Malhotra, M. P. Kumar and S. N. Maheshwari, "An

$O(|V|^3)$ algorithm for maximum flows in networks", Information Processing Letters, Vol. 7: pp. 277-278

[72] E. J. H. Chang, "Echo algorithms: Depth parallel operations on general graphs", IEEE Trans. Software Engineering, Vol. SE-8, pp.391-401, 1982.

[73] K. M. Chandy and J. Misra, "Distributed computation on graphs: shortest path problems", Comm. ACM, Vol. 25, pp. 833-837, 1982.

[74] A. Segal, "Decentralized maximum-flow protocols", Networks, Vol. 12, pp. 213-230, 1982.

[75] To-Yat Cheung, "Graph traversal techniques and the maximum flow problems in distributed computation", IEEE Trans. Software Engineering, Vol. SE-9, pp.504-512, 1983.

[76] A. Segal, "Distributed protocols", IEEE Trans on Information theory", Vol. IT-29, pp.23-35, 1983.

[77] R. G. Gallagher, P. A. Humblet and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees", ACM Trans. Prog. Lang. Syst., Vol. 5, pp. 66-77, 1983.

[78] E. Korach, D. Rotem and N. Santoro, "Distributed algorithms for finding centers and medians in networks", ACM Trans. Prog. Lang. Syst., Vol. 6, pp. 380-401, 1984.