HIERARCHICAL CONTROL FOR FINITE STATE MACHINES

Gang Shen

Departement of Electrical and Computer Engineering
McGill University, Montreal

August 1999

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

0-612-64668-8

ABSTRACT

We base the notion of state aggregation for finite state machines (FSM) on the *dynamical consistency (DC)* relation ([**17**]) between the blocks of states in any given state space partition π. In this framework, we present the new notion of *ST dynamical consistency (ST-DC))* for *source-target (ST)* FSMs where there is a preferred sense of flow from a set of source states (S) to a set of target states (T). It is proven that if a partition π is *ST in-block controllable (ST-IBC)*, the partition machine of an ST FSM M based on π, M^{π} (i.e. high level abstraction of M based on π), is controllable if and only if M itself is controllable. We also prove that all ST-IBC partition machines of M form a lattice and any chain from the top to the bottom of this lattice provides a *hierarchical feedback control structure.*

This methodology is next extended to optimal control problems for discrete event systems (DES) modelled by finite state machines. A partition machines based scheme called *hierarchically accelerated dynamic programming (HADP)* is introduced which significantly speeds up the standard dynamic programming procedure (up to several orders of magnitude) at the cost of a certain degree of sub-optimality. We present necessary and sufficient conditions for the HADP procedure to generate globally optimal solutions and, further, give bounds on the degree of sub-optimality. An example called the Broken Manhattan Grid (BMG) system is used to illustrate the implementation of HADP, and flexible and generalisable code for this example is described.

Many complex systems appear in the form of the product of multiple interacting sub-systems. A formulation of multi-agent systems is presented where the dynamics of the agents are described by default specifications, of a sets of forbidden state-event relational pairs, denoted $\mathcal{R}$. Such systems are called *relational multi-agent product* *systems (MA($\mathcal{R}$))*. The application of the HADP methodology to relational multi-agent product systems is analysed. A multi-machine system consisting of a time counter and agents called a *timed multi-agent relational product (TMA($\mathcal{R}$))* is formulated.

To apply hierarchical control to the routing problem for networks, we consider two conceptual classes of networks: first, *link network systems (LN)*, and, second, *buffer network systems (BN)*. The notions of *dynamical costs* and *network states* are introduced. In particular, the notion of *throughput-independent ST-IBC (TI-ST-IBC)* partitions is used to formulate the *incremental HADP (IHADP)* methodology. For the multiple objective optimisation problem of LNs, a notion of (vector) network state is introduced to carry the information describing the available transmission capacity of each link. For buffer network systems, the notion of *(matrix) network states* is given.

RÉSUMÉ

On fonde la notion d'aggrégation d'état pour les machines à état fini (FSM). introduites dans la relation *dynamiques consistantes* ([**17**]) entre les blocs des états d'une partition d'état π. Dans cette thèse. on présente la nouvelle notion de ST. *consistance dynamique. (ST-DC)* pour *source-déstination* (ST FSMs) ou le sens de la circulation de l'état source (S) à l'état cible (T) est préferée. Ceci donne naissance à une définition de la dynamique à haut niveau sur la machine finie correspondante à la partition blocs donnée. Si une partition π est *ST controllable en bloc (ST-IBC).* une partition machine d'une ST FSM M basée sur π. M^{π}. est controllable si et seulement si M est controllable. Cette définition prouvée. nous avons prouvé que si tous les états dans M sont co-accessibles à T. toutes les partitions machines ST-IBC de M provenant de la tréllis et de toute chaine de haut en bas de cette tréllis donne une structure *hiérarchique de commande à retour.*

Cette méthodologie est élargie aux problems de commande optimale pour les systèmes à évènement discret (DES) modelisés par des machines à état fini. Une partition machine basée sur le schema applé *programmation dynamique à hiérarchie accelerée (HADP).* qui accélère sensiblement la procedure de la programmation dynamique standard au coût d'un certain degré de sous-optimalité. est introduite. On présente une condition suffisante et necessaire pour la procédure (HADP) afin de générer des solution globales et optimales. mieux encore. donner des limites du degrée de sous-optimalité. Un exemple d'illustration d'une implémentation (HADP) est aussi donné.

Plusieurs systèmes complexes apparaisant sous la forme de sous-systèmes multiples interagis. Une formulation de systèmes multi-agents est presentée où les dynamiques des agents sont décrites. par des spécifications à default. notée $\mathcal{R}$. Ce genre de système est appelé *systèmes de* produit multi-agent relationnel (MA($\mathcal{R}$)). L'application d'une méthodolgie HADP à ces systèmes est analysée. Un système à multi-machines á base de compteurs et agents, appelé *produit relationnel multi-agent temporel (TMA($\mathcal{R}$))*, est formulé.

Afin d'appliquer une commande hiérarchique. aux problemes de réseaux, on considère deux classes conceptuelles de reseaux: la première est les *systèmes à réseau lié (LN)* est la deuxième est les systèmes à *réseau temporaire (BN)*. Les notions de côutes dynamique et états réseau. sont introduites. En particulier la notion *throughput-independent ST-IBC (TI-ST-IBC) partitions* est utilisée pour formuler la méthodologie (HADP) incrémentale pour des problems multiples d'optimisation de LNs. Une notion d'*état réseau (vector)* est introduite afin de transporter l'information décrirant la capacité disposible de transmission de chaque næud: pour les systèmes à reseau temporaire. la notion d'*états à réseau (matrice)* est donnée.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis supervisor, Professor Peter E. Caines, for his constant encouragement and generous sponsorship. Without his original inspiration and technical guidance, this thesis would never have begun. I must thank him for his rigorous reading of this manuscript, and for his suggestions on both style and the conceptual framework. Like all of Peter's graduate students, I will miss his famous Sunday morning calls, his commitment to scientific accuracy, and his foresight in new disciplines. My discussions with Professor Caines have been an enjoyable experience for me.

NSERC, the Greville Smith McGill Major Fellowship Foundation (1998-1999), FCAR (1999) and NASA-Ames Research Centre (Mountainview, CA) all deserve my gratitude for the financial support I have received from them.

I would also like to extend my appreciations to Dr. Charalambos Charalambous, Paul Hubbard, Lemch Ekaterina, and other CIM fellow students who have always been a challenging and pro-active audience for the presentations of my research. Their constructive and valuable comments led to the improvement of my work.

Thanks should go to the CIM management and technical staff who have provided a pleasant environment for me to concentrate on writing this thesis.

I would like to thank Nabil Aouf, who did the translation of the abstract into French. Thanks to Carlos Martínez-Mascarúa, who helped me struggle with Latex, and let me use his rich library of style files free of charge.

Finally, but most importantly, I thank my beloved wife, Hongbo, to whom I owe my entire life. Throughout this strenuous period of our life, her support has been my source of momentum to never give up hope. Unfortunately, I cannot find appropriate words to express my thanks to my parents; perhaps no words would do. They never hesitate to lend me their forgiveness when I disappoint them, for which I have no means to compensate them.

Gang Shen

Montréal, Québec, August 1999

CLAIMS OF ORIGINALITY

This thesis contains the following original contributions:

- A theory of hierarchical control for finite systems with a preferred sense of flow. The notions of ST-dynamical consistency and ST-in-block controllability. [Chapter 2]
- A theoretical framework of hierarchically accelerated dynamic programming (HADP) for trajectory optimisation in very complex systems, including the notions of optimality consistency, convex high level trajectories and semi-dual high level graphs. [Chapter 3]
- Algorithms for the implementation of HADP and Incremental HADP. [Chapters 3 and 5]
- The notion of relational multi-agent product finite state machines ($MA(\mathcal{R})$ FSMs). [Chapter 4]
- The notion of vector and matrix network states and their applications in hierarchical network routing. [Chapter 5]

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Traditional control problems concern continuous time and discrete time vector state valued systems which are usually modelled by differential and difference equations respectively; on the other hand, discrete event systems (DES) are distinguished from these traditional models in that the evolution of the finite set valued state of a DES is driven by the occurrences of some sequence of events ([34], [21], [60]).

Due to its conceptual simplicity and its vast range of applications, the finite state machine model is one of the most basic formal mechanisms for DES models ([29], [12], [59]). Furthermore, principally because of the advances in and universal application of computer systems, discrete event systems are applied in almost every aspect of engineering ([40], [37], [50]). In particular, as a foundation for research in hybrid systems (i.e. systems which possess mixed continuous and discrete behaviours), it is certainly foreseeable that DES theory will find an ever growing domain of application ([46], [10], [11], [57], [38], [62]).

Progress in digital computing technology has made possible the implementation of many advanced algorithms. However, computational complexity due to high state dimension, and frequent information exchanges, make the control of large scale systems extremely difficult ([4], [30]); consequently, to manage the computational complexity of control synthesis, hierarchically structured information and control systems are frequently employed ([44], [3], [12]). By grouping together the states of the original system, which in many applications has a huge cardinality and simpler dynamics, state aggregation will generate a system with a smaller set of (aggregate) states ([7], [8], [2]). Thus, as a principal way to create an applicable hierarchy, state aggregation plays a significant role in the analysis and control of large scale systems ([**27**]). This observation motivated the research presented in this thesis.

The Hierarchical Control of ST FSMs

A notion of state aggregation for finite machines was introduced in [**17**] via the concept of the *dynamical consistency* (DC) relation between the blocks of states in any given state space partition π. This formulation results in a definition of higher level dynamics on the finite (partition) machine M^π whose states correspond to the given partition elements.

Chapter 2 of this thesis treats the more general case of *Source-Target systems* (ST-systems) where there is a preferred sense of flow from a set of source states (S) to a set of target states (T). A generalisation of the theory of [**17**] to ST-systems is given which includes the generalisation of the notions of dynamical consistency, in block controllability and hierarchical feedback control on the associated hierarchical lattices. The dynamics of a higher level machine M^π in a hierarchy $\{M, M^\pi\}$ are defined in accordance with the ST-DC relations over the partition blocks of the lower level model M.

A class of hierarchical control structures for ST-systems is presented based on the notion of *ST dynamical consistency* (ST-DC). It is shown that when a partition π of M possesses some particular properties, which are termed *ST in-block controllability* (ST-IBC), the higher level finite state machine defined based on π is ST-controllable if and only if M is ST-controllable. Further, if M is *T-trimmed*, i.e., every state of M is co-accessible to T, the set ST-IBC partition machines of M forms a lattice structure, and any chain from the top element to the bottom element of this lattice provides a *hierarchical control structure* for M.

The material in Chapter 2 follows that in [**14**] and [**15**], co-authored by P.E. Caines, V. Gupta and G. Shen.

Hierarchically Accelerated Dynamic Programming

Complex finite state systems arise in many contexts, in particular, optimal control problems for finite state machines have significant applications in transportation management, manufacturing systems and telecommunication networks. The main theoretical foundation and computational technique for finding optimal trajectories in finite state systems is that of *dynamic programming* (DP)([**45**], [**8**], [**51**]). Consequently, a large number of algorithms have been developed to solve DP problems. The time complexity of these algorithms naturally depends upon the size of the models involved: in general, complexity grows non-linearly (specifically, faster than quadratically) with the number of states in the system ([**25**]). Hence computational efficiency degrades significantly and, in particular, standard DP algorithms are not applicable to real-time problems of practical interest.

Chapter 3 presents a scheme called *hierarchically accelerated dynamic programming (HADP)* which significantly speeds up dynamic programming (by up to several orders of magnitude) for discrete event systems modelled by finite state machines at the cost of a certain degree of sub-optimality.

The HADP methodology is based upon (possibly iterated) dynamical abstraction of the given DES (by state aggregation) which generates a *control consistent* hierarchy of finite state machines. We discuss necessary and sufficient conditions for the HADP procedure to generate globally optimal solutions and, further, give bounds on the degree of sub-optimality which can occur. Various approaches are proposed to improve the accuracy of the sub-optimal solutions by using *semi-dual high level graph* while reducing the computational time via the use of *weakly ST-IBC partitions*. Finally, an example called the Broken Manhattan Grid (BMG) system is used to illustrate our software implementation of HADP.

Some contributions in this chapter appeared in [**55**], [**56**] and [**53**], co-authored by P.E. Caines and G. Shen.

Relational Multi-agent Systems

In a variety of situations, interacting agents are involved in an integrated environment ([41], [64], [61]). The state transition of each agent takes place when an event happens, and events in distinct agents may occur concurrently or non-simultaneously. The behaviour of the individual agents is regulated by the interaction between all of the agents which can be present in various forms.

Chapter 4 analyses the systems of interacting finite state machines via the proposed models of *multi-agent (MA)* product and *timed multi-agent (TMA)* product.

A notion of a *simultaneous product* of finite state machines for which the events have the same execution duration (*live time*) is proposed, and the necessary and sufficient conditions for simultaneous product systems to be controllable are discussed. When the interaction between agents is given in terms of a set of *forbidden (state-event) configurations* $\mathcal{R}$, the system of n interacting finite state machines M_i, $1 \leq i \leq n$, is modelled by $(\|_{\neg\mathcal{R}})_{i=1}^{n} M_i$, the *multi-agent (relational)* $(MA(\mathcal{R}))$ *product* of M_i. The application of the HADP algorithm to $(\|_{\neg\mathcal{R}})_{i=1}^{n} M_i$ is formulated in this chapter, and some preliminary results on the construction of the related control hierarchies are also given. A *time counter* is used to observe the state transitions of M_i, $1 \leq i \leq n$, in case distinct events in M_i have different live times. Finally, the product system of the time counter and M_i, $1 \leq i \leq n$, the so called *timed multi-agent (relational)* $TMA(\mathcal{R})$ *product* is formulated.

Hierarchical Network Routing

Dynamical routing in a network is a way of providing flexibility to adapt to changing and volatile traffic demands. Traditionally, for very large networks, a multi-level hierarchical routing is used to reduce the size of each routing table ([58]). With the recent strides in microprocessing technology, dynamic traffic management has become practical ([48]) and state-dependent routing is used to increase network utilisation. The key features of state-dependent routing include the explicit use of global network information and very short update cycle times.

In Chapter 5, we discuss the hierarchical routing methodologies within the context of several network models.

To cope with the changing traffic load. the notion of a *throughput ST-IBC* partition is used to form a stable hierarchical control structure. As each new *request* (*message*) arises, the high level costs are recalculated according the the feedback of the network state and HADP is applied for route optimisation. The resulting *incremental HADP (IHADP)* algorithm is formulated in the first part of this chapter.

Consider a network of links for which the capacity of all nodes (viewed as buffers) is assume to be infinite. If multiple requests arrive at the network at the same time instant. the overall optimal solution may differ from the optimal solutions for individual requests because of the constraints of the limited capacity of links. A *vector network state* carries the information describing the available transmission capacity of each link. For n requests. if we can derive a hierarchy consisting of the *throughput IBC* partition machines of the original network. HADP may be applied.

Next we consider networks with buffers at the nodes and for which the capacity of all links is assumed to be infinite. The notion of a *matrix network state* is given for such networks. By decomposing an IBC block of a partition of the network nodes into a set of intersecting rings. we obtain a method to ensure the *in-block controllability* of a network state. Finally. we prove that when high level network is in a controllable state and each of the partition blocks of the original network is in an in-block controllable state. the original network is in a controllable state.

CHAPTER 2

The Hierarchical Control of ST Finite State Machines

2.1. Introduction

Hierarchically structured information and control systems occur for at least two related reasons: first, the great complexity of many natural and designed systems limits the ability of humans and machines to describe and comprehend them, and, second, the inherent limitations on the information processing capacity of feedback regulators result in the regulators (and possibly the controlled systems) being organised in special, in particular hierarchical, configurations.

Many mathematical theories and engineering methodologies have been developed as a response to these problems. Examples are readily found in the power distribution industry and in large scale manufacturing industries. Further important examples are found in the organisation and control procedures of the communications, rail, road and air traffic systems. All of these examples involve some form of aggregation in state space, and most involve some form of hierarchically structured flow of information and control signals. Large social organisations such as governments and corporations provide sociological examples of such structures, while systems which probably do not display these features are idealised markets and the unorganised target-seeking behaviour of messages on the Internet as currently organised.

Theoretical work on hierarchical control has a large literature and has connections to, among other subjects, game theory, mathematical programming and optimal resource allocation. These topics were presented together with their connections to control theory in [**30**]. More recently, formulations of hierarchical control have appeared in stochastic control [**52**], automated highway system studies [**63**] and within the supervisory control formulation of discrete event system theory (e.g. [**42**], [**49**], [**68**], [**70**]).

The work in this chapter follows that in [**66**], [**17**] and [**65**], where a new notion of state aggregation is introduced via the concept of the dynamical consistency (DC) relation between the sets of states constituting the members of any given partition π of the state space. This formulation results in a definition of high level dynamics on the finite (partition) machine M^{π} whose states correspond to the given partition elements. The DC relation is then similarly defined on any further partition of the state set of M^{π}; and so on. The theoretical development in [**17**] gives the lattice structure of the class of so-called *in block controllable (IBC)* partition machines. The notion of state aggregation given by the concept of the DC dynamics of a partition machine permits a natural construction of a large class of hierarchical control structures on any given finite state machine. It is to be noted that in the purely graph theoretic setting, without any controlled dynamics, an idea related to that of DC dynamics is to be found in [**28**].

Since there is a natural parallel between the formulation of levels in hierarchical system theory and their definition in hybrid system theory, where discrete systems play the roles of both models and controllers for finer continuous state systems. In [**19**] and [**18**], the theory of [**17**] is generalised to the hybrid case, and to [**11**].

In the analysis and design of hierarchical control systems, one is often interested in the reachability of a set of terminal states from a initial (or start) set of states. Many examples of systems with such a preferred sense of flow are to be found among natural and designed systems. As a result, in this chapter, we consider a generalisation of the theory of hierarchical control initiated in [**17**] to systems in which there are distinguished source and target sets for the controlled flow.

2.2. The ST-Dynamical Consistency Relation and ST-IBC Partition Machines

Consider a finite state machine, $M = \{X, \Sigma, \delta\}$, where X is a finite set of states, Σ is a finite set of (forced) events, and $\delta : X \times \Sigma \to X$ is the (partial) state transition function of the system. We shall use the standard notion Σ^* for the set of all finite sequences (including the empty string ϵ) of elements of Σ.

A *partition* π of the state space X of M is a collection of subsets of X, namely, $\pi = \{X_1, X_2, \ldots, X_{|\pi|}\}$ satisfies (1) $X = \bigcup_{i=1}^{|\pi|} X_i$; (2) $X_i \cap X_j = \emptyset$ for $1 \leq i \neq j \leq |\pi|$; (3) $X_i \neq \emptyset$, $1 \leq i \leq |\pi|$. A *partition machine* M^{π} takes π as its state space and its elements are called *blocks*.

Let $(v < u)$ $v \leq u$ denote that a string v is a (proper) prefix of u.

Denote a distinguished subset of states called *source states* and a distinguished subset called *target states* with $S \subseteq X$ and $T \subseteq X$ respectively. We shall term finite systems with such distinguished subsets *ST-systems*.

We note that in the case where the sets S, T and X are identical all the definitions and results below become consistent with their counterparts in [**17**]. In this sense the work in this chapter generalises [**17**].

Definition 2.1. (ST-Controllability)
(1) (Strong ST-Controllable) M is said to be *strongly ST-controllable* if and only if for every $x \in S$ and for every $y \in T$, there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$.
(2) (Weak ST-Controllability) M is said to be *weakly ST-controllable* if and only if for every $x \in S$ there exists $y \in T$ and there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$. $\square$

Clearly, strong ST-controllability (and hence weak ST- controllability) is a significantly weaker property than standard controllability because the latter requires the accessibility of every state from every other state.

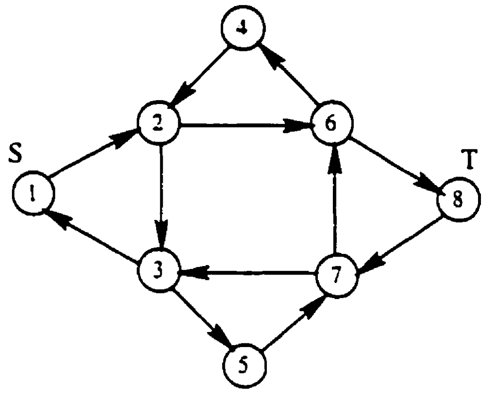

FIGURE 2.1. An 8-state machine M_8

Example 2.1. In the 8-state machine M_8 shown in Figure 2.1. $S = \{1\}$ and $T = \{8\}$. Clearly, this ST-system is both strongly ST-Controllable and weakly ST-Controllable. □

We recall that a *partition* π of a finite set X is a collection of pairwise disjoint subsets called blocks, $X_i \subseteq X$, $1 \le i \le |\pi|$, such that $X_i \cap X_j = \emptyset$ for $i \ne j$ and $X = \cup_{i=1}^{|\pi|} X_i$. A *partial order* relation $\preceq$ *(finer than)* on partitions of X is defined such that for two partitions, $\pi_1 = \{X_1^1, X_2^1, \ldots, X_{|\pi_1|}^1\}$ and $\pi_2 = \{X_1^2, X_2^2, \ldots, X_{|\pi_2|}^2\}$, $\pi_1 \preceq \pi_2$ if and only if for each block $X_i^1 \in \pi_1$, there is an $X_j^2 \in \pi_2$ such that $X_i^1 \subseteq X_j^2$, where $1 \le i \le |\pi_1|$, $1 \le j \le |\pi_2|$, i.e. π_1 is a refinement of π_2.

In the example above, for the two partitions $\pi_1 = \{\{1\}, \{2, 4\}, \{3\}, \{5, 7\}, \{6\}, \{8\}\}$, $\pi_2 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}\}$, we see that $\pi_1 \preceq \pi_2$.

Definition 2.2. ($I(X_i, S, T)$, $O(X_i, S, T)$) Consider a partition $\pi = \{X_1, X_2, \ldots, X_{|\pi|}\}$, of the state set X of a finite state machine M. In each block $X_i \in \pi$, $1 \le i \le |\pi|$, we specify two subsets, respectively $I(X_i, S, T)$ and $O(X_i, S, T)$, which are termed the *local entries* (or *in-set*) and *local exits* (or *out-set*); these are defined respectively as follows:
$x \in I(X_i, S, T) \Longleftrightarrow x \in S \cap X_i$ or there exists $x' \in (X - X_i)$, i.e. the complement of X_i in X, and there exists $u \in \Sigma$ such that $\delta(x', u) = x$;
$y \in O(X_i, S, T) \Longleftrightarrow y \in T \cap X_i$ or there exists $y' \in (X - X_i)$ and there exists $u \in \Sigma$ such that $\delta(y, u) = y'$. □

In the partition $\pi = \{\{1,2,3,4\},\{5,6,7,8\}\}$ of Example 2.1, for $X_1 = \{1,2,3,4\}$, $I(X_1,S,T) = \{1,3,4\}$ and $O(X_1,S,T) = \{2,3\}$; for $X_2 = \{5,6,7,8\}$, $I(X_2,S,T) = \{5,6\}$ and $O(X_2,S,T) = \{6,7,8\}$.

We now shall define the appropriate generalisation of the notions of dynamical consistency and partition machines to ST-systems.

Definition 2.3. (ST-Dynamical Consistency (ST-DC)) The relation of ST-dynamical consistency for an ordered pair of blocks $\langle X_i, X_j\rangle$ in a partition π is defined as follows:
$\langle X_i, X_j\rangle \in \pi \times \pi$ is called *ST-dynamically consistent (ST-DC)* if one of the following cases holds:
(a) $i \neq j$. For each $x \in I(X_i,S,T)$

> (1) there exists $y \in O(X_i,S,T)$ and there exists $u_{i,i} \in \Sigma^*$ such that $\delta(x,u'_{i,i}) \in X_i$ for all $u'_{i,i} \leq u_{i,i}$, and $\delta(x,u_{i,i}) = y$; and
> (2) for at least one such y, there exists $z \in I(X_j,S,T)$ and there exists $u_{i,j} \underset{\Sigma}{\boxed{\in U}}$ such that $\delta(y,u_{i,j}) = z$.

We write $u_{i,i} \cdot u_{i,j}$ as u_i^j, where $\cdot$ denotes concatenation.
(b) $i = j$

> For every $x \in I(X_i,S,T)$ there exists $y \in I(X_i,S,T)$, and there exists some non-null $u_{i,i} \in \Sigma^*$ such that $\delta(x,u'_{i,i}) \in X_i$ for all $u'_{i,i} \leq u_{i,i}$ and $\delta(x,u_{i,i}) = y$. □

In the case $i = j$ the condition above simply requires that each input state of X_i should have a non-empty controlled path within X_i which makes it return to the in-set so as to form a high level *pseudo-cycle*. This is postulated to obtain desirable properties for the formal language of high level transitions as defined below.

Example 2.2. In Example 2.1, $\langle\{1,2,3\},\{4,6\}\rangle$ is ST-DC. Here the first partition block is such that its in-set is equal to the whole block and the out-set is the pair of elements $\{2,3\}$. The second block is such that its in-set is the element $\{6\}$, which is accessible in one step from $\{2\}$, and the out-set of $\{4,6\}$ is the whole set. But $\langle\{1,2,3\},\{4\}\rangle$, where $\{4\}$ is both an input and and out-set, is not ST-DC since $\{4\}$ is not accessible in one step from the first set. A general ST-DC relation is represented in Fig. 2.2, in which we see displayed I-unreachable states (i.e those not reachable

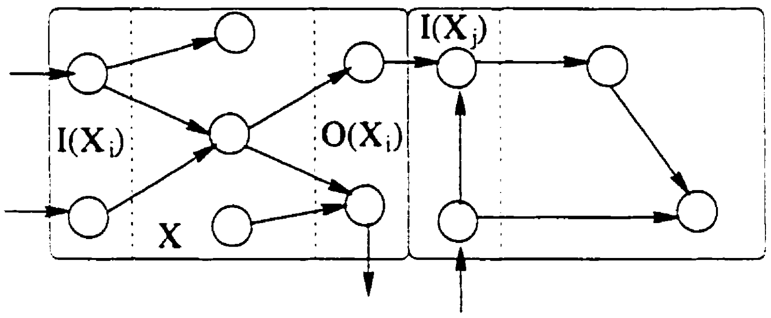

FIGURE 2.2. $\langle X_i, X_j \rangle$ is ST-DC but not DC

from the input states of X_i) and O-inaccessible states (i.e. those from which the output states of X_i are not reachable) in the block X_i. □

A *high level transition (input) event* U_i^j is defined, and denoted by U_i^j, if and only if $\langle X_i, X_j \rangle$ is ST-DC; in other words, for any pair i, j, U_i^j is defined if and only if the conditions of Definition 2.3 hold. The term high level transition is to be taken relative to the base machine M, but no mention shall be made of this whenever the context is clear. Let Σ^π denote the collection of all such U_i^j for which we note that the high level null string ϵ is an element of $(\Sigma^\pi)^*$.

In order to define the *partition machine* $M^\pi = \{\pi, \Sigma^\pi, \delta^\pi\}$ (based upon the partition π of M), we define the state transition function $\delta^\pi : \pi \times \Sigma^\pi \to \pi$ by $\delta^\pi(X_i, U_i^j) = X_j$, $X_i, X_j \in \pi, 1 \leq i, j \leq |\pi|$, whenever $\langle X_i, X_j \rangle$ is ST-DC. We may now define $\delta^\pi : \pi \times (\mathcal{U}^\pi)^* \to \pi$ recursively as follows: first set $\delta^\pi(X_i, \epsilon) = X_i$; then, for strings in $(\Sigma^\pi)^*$ of length one, the definition of δ^π gives $\delta^\pi(X_i, U_i^j) = X_j$ whenever $\langle X_i, X_j \rangle$ is ST-DC; finally, for strings U in $(\Sigma^\pi)^*$ of length greater than one we set $\delta^\pi(X_i, U_i^j \cdot U) = \delta^\pi(X_j, U)$ if $\delta^\pi(X_i, U_i^j) = X_j$ and if $\delta^\pi(X_j, U)$ is defined. Here $\cdot$ indicates the concatenation of two strings. From this recursive definition we immediately obtain the following fact as a special case.

why the font change?

Lemma 2.1. (Semi-Group Property) $\delta^\pi(X_i, U_1 \cdot U_2) = \delta^\pi(\delta^\pi(X_i, U_1), U_2)$ as long as $\delta^\pi(X_i, U_1)$ and $\delta^\pi(\delta^\pi(X_i, U_1), U_2)$ are defined, where $\cdot$ means the concatenation of two sets. □

It may be verified that when a chain of high level transitions is defined, it is the case that the appropriate generalisation of Definition 2.3 holds; that is to say, for every state in the I-set of the initial block there exists a path through a chain of O and I state sets in the successive blocks which terminates in the I states of the final block. It is to be noted that the definition of one step pseudo-cycles in Definition 2.3 permits well defined chains of high level transition event strings to contain sets of pseudo-cycles, i.e. high level identity elements.

For a given state space X and partition π, let S^π denote the elements of π containing states lying in S, i.e., $X_i \in S^\pi$ if $X_i \cap S \neq \emptyset$, and similarly let T^π denote the elements of π containing states in T, i.e., $Y_j \in T^\pi$ if $Y_j \cap T \neq \emptyset$. Then we may give the following definition of the controllability of the partition machine $M^\pi \triangleq (\pi, \Sigma^\pi, \delta^\pi)$.

Definition 2.4. (ST-Between Block Controllability) A partition machine $M^\pi = (\pi, \Sigma^\pi, \delta^\pi)$ is
(1) *strongly ST between block controllable (strongly ST-BBC)* if it is the case that for every $X_i \in S^\pi$ and $Y_j \in T^\pi$, there exists a $U \in (\Sigma^\pi)^*$ such that $\delta^\pi(X_i, U) = Y_j$; and
(2) *weakly ST between block controllable (weakly ST-BBC)* if it is the case that for every $X_i \in S^\pi$, there exists a $Y_j \in T^\pi$ and there exists a $U \in (\Sigma^\pi)^*$ such that $\delta^\pi(X_i, U) = Y_j$. □

inconsistent fonts.

We note that we may obtain weak ST-BBC from strong ST-BBC by simply exchanging a universal quantifier for an existential one. As this applies to the results below we henceforth only discuss strong properties in detail and leave the development of the analogous weak properties to the reader.

Definition 2.5. (ST-In-Block Controllability) A block $X_i, 1 \leq i \leq |\pi|$ is *ST-in block controllable (ST-IBC)* if and only if either $I(X_i, S, T) = \emptyset$ or $O(X_i, S, T) = \emptyset$ or the following two conditions hold conjointly:
(1) For every $x \in I(X_i, S, T)$ there exists $y \in O(X_i, S, T)$, and there exists $u \in U^*$ such that $\delta(x, u') \in X_i$ for each $u' < u$, and $\delta(x, u) = y$.
(2) For every $y \in O(X_i, S, T)$ and for every $z \in O(X_i, S, T)$ with $z \neq y$, there exists $v \in U^*$ such that $\delta(y, v') \in X_i$ for each $v' < v$, and $\delta(y, v) = z$, i.e., the states in $O(X_i, S, T)$ are mutually accessible with respect to X_i.
A partition π is said to be *ST-in-block controllable (ST-IBC)* if every block of π is ST-IBC. □

In other words, any in-set state of an ST-IBC block $X_i \in \pi$ must have an internal trajectory going to an out-set state of X_i, and the exit states of X_i must be mutually accessible, i.e., X_i is weakly ST-controllable and $O(X_i, S, T)$ is mutually accessible with respect to X_i. In the 8-state machine of Example 2.1, the block $\{1, 2, 3, 4\}$ is ST-IBC, although it is not IBC in the sense defined in [**17**].

We observe that in the standard case where $S = T = X$, ST-in block controllability specialises to the standard IBC property because in this case all elements of any block X_j are mutually accessible. Further, in case $S = T = X$, ST-between block controllability clearly implies that the standard BBC property holds.

We let $\pi_{ST}^{IBC}(X)$ represent the collection of all ST-IBC partitions of X and $M_{ST}^{IBC}(M)$ denote the collection of all partition machines of M corresponding to partitions in $\pi_{ST}^{IBC}(X)$.

Let $\rightarrow$ represent a one-step state (block) transition in M and M^π.

Theorem 2.1. *If M^π is an ST-in-block controllable partition machine of M, then M^π is strongly (respectively, weakly) ST-between block controllable if and only if M is strongly (respectively, weakly) ST-controllable.*

Proof:

We only prove the implications concerning strong ST-controllability as the weak case follows by an analogous argument.

$\Longrightarrow$:

Given $M^\pi \in M_{ST}^{IBC}(M)$. Assume M is strongly ST-controllable and let us look at arbitrary blocks X_i in S^π and $X_j \in T^\pi$. Now consider any $x \in X_i \cap S$ and $y \in X_j \cap T$; then by Definition 2.1, we know there is a trajectory from x to y. Suppose this trajectory traverses a chain of blocks in π, in the order $Y_1, Y_2, \ldots, Y_k$, $Y_1 = X_i$, $Y_k = X_j$. Because all of the blocks are ST-IBC it follows that the block pairs $\langle Y_1, Y_2\rangle, \langle Y_2, Y_3\rangle, \ldots, \langle Y_{k-1}, Y_k\rangle$ satisfy the conditions to be ST-DC. Thus, we have well-defined high level transitions, $\delta^\pi(Y_1, U_2^1) = Y_2, \delta^\pi(Y_2, U_3^2) = Y_3, \ldots, \delta^\pi(Y_{k-1}, U_k^{k-1}) =$

Y_k, and hence $\delta^\pi(Y_1, U_2^1 \cdot U_3^2 \cdot ... \cdot U_k^{k-1}) = Y_k$, i.e., M^π is strongly ST-between block controllable.

$\Longleftarrow$:

Let M^π be strongly ST-between block controllable. Consider an $x \in S$ and $y \in T$. Then there must be some $X_i \in S^\pi$ and $X_j \in T^\pi$ such that $x \in X_i$ and $y \in X_j$. By Definition 2.4, we know, there is a (finite) sequence of events in M^π, $U_2^1 \cdot U_3^2 \cdot ... \cdot U_k^{k-1}$ from X_i to X_j. Let this sequence of block transitions correspond to a trajectory of blocks, $Y_1 \to Y_2 ... \to Y_k$, $Y_1 = X_i$ and $Y_k = X_j$. Because each pair of adjacent blocks Y_l and Y_{l+1} on the above trajectory of blocks are ST-DC, by Definition 2.3, there exists a trajectory from x to some input state of X_k. Because of the ST-in-block controllability of X_k, this input state must have an in-block trajectory leading to $y \in O(X_k, S, T)$. Thus, we get a complete trajectory from x to y. Hence, M is strongly ST-controllable. $\square$

2.3. The T-trimmed ST-Machine and Its Associated ST-IBC Lattice

For any given ST-system modelled by $M = \{X, \Sigma, \delta\}$, we term the states in X which are not reachable from S *S-inaccessible states*, and term the states in X from which T is not reachable *T-co-inaccessible states*.

The S-inaccessible and T-co-inaccessible states of FSM M are irrelevant to the ST-control problem: this is because any path from a state in S to a state in T cannot pass through either of the S-inaccessible and T-co-inaccessible states. From the point of view of ST-controllability, all the S-inaccessible or T-co-inaccessible states may be deleted before we investigate the ST-control problem. This would yield a minimal realisation of M, M', by which we mean every state of M' is S-accessible and T-co-accessible, and hence there are no redundant states with respect to the ST-controllability problem for the resulting ST-system. In this chapter, however, it is sufficient to eliminate the T-co-inaccessible states of a finite state machine M and we denote the finite state machine obtained after this T-trimming process by $M_t = (X_t, U_t, \delta_t)$. M_t is said to be the *T-trimmed* FSM of M.

Definition 2.6. (Chain Union $\cup^C$) The chain union of two partitions π_1 and π_2 of X, denoted $\pi_1 \cup^C \pi_2$, is the least upper bound of π_1 and π_2 with respect to $\preceq$ in the set of partitions of X. ← This actually forms a complete lattice and the lub is what you call "chain union". □

The following algorithm may be used to calculate the chain union of two partitions. Each of the distinct blocks Z_i of the partition $\pi_1 \cup^C \pi_2, \pi_1, \pi_2 \in \pi_{ST}^{IBC}(X)$, can be constructed recursively by setting $Z_i = \bigcup_{n=1}^{N} Z_{i,n}$, $N = \max\{|\pi_1|, |\pi_2|\}$, where $Z_{i,n}$ is given by the following algorithm: Set $Z_{i,1} = X_i$ for some $X_i \in \pi_i$, then for all $n, 1 \leq n < N$,

$$Z_{i,n+1} = \begin{cases} Z_{i,n} \cup \{X' \in \pi_1 : Z_{i,n} \cap X' \neq \delta\} & n \text{ odd,} \\ Z_{i,n} \cup \{Y' \in \pi_2 : Z_{i,n} \cap Y' \neq \delta\} & n \text{ even} \end{cases}$$

Alternatively, let us define $x \sim x'$ if either $x, x' \in X_i^{\pi_1} \in \pi_1$, $1 \leq i \leq |\pi_1|$, or $x, x' \in X_j^{\pi_2} \in \pi_2$, $1 \leq j \leq |\pi_2|$. Then the equivalence classes of the transition closure of this relation are the blocks of $\pi_1 \cup^C \pi_2$.

The property of ST-in-block controllability which we have defined above restricts the set of partitions in such a way that it is preserved under chain union.

Theorem 2.2. *For $\pi_1, \pi_2 \in \pi_{ST}^{IBC}(M_t)$, the chain union of π_1 and π_2 is ST-IBC, i.e., $\pi_1 \cup^C \pi_2 \in \pi_{ST}^{IBC}(M_t)$.*

Proof:

Suppose π_1 and π_2 are two ST-IBC partitions of X_t, the T-trimmed finite state machine of M, let us look at their chain union $\pi_1 \cup^C \pi_2 = \{Z_1, Z_2, \ldots, Z_r\}$. First, we prove that if A and B are ST-IBC, $A \cup B$ is ST-IBC whenever $A \cap B \neq \emptyset$. It is clear that $I(A \cup B, S, T) \subseteq I(A, S, T) \cup I(B, S, T)$ and $O(A \cup B, S, T) \subseteq O(A, S, T) \cup O(B, S, T)$ so that we need only consider the nontrivial cases when $O(A, S, T) \cup O(B, S, T) \neq \emptyset$.

After eliminating all states which are not T-co-accessible, at least one of the output states of A or B is in $A \cap B$. This is shown as follows. Since we may suppose $O(A, S, T) \neq \emptyset$ (recall $O(A, S, T) \cup O(B, S, T) \neq \emptyset$), without loss of generality, take $x \in A \cap B$, $y \in O(A, S, T)$ then there is an internal path in A from x to y (since X_t

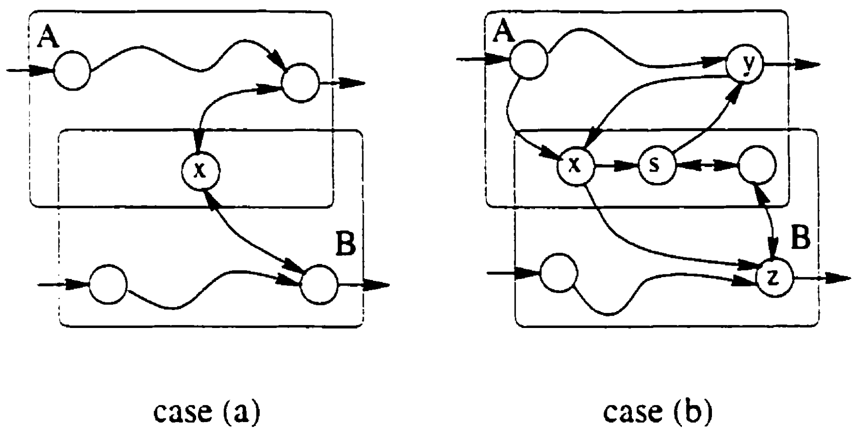

FIGURE 2.3. ST-IBC is closed under chain union

has been thinned, i.e., all T-co-inaccessible states in X have been eliminated, such that the states left in X_d must have a trajectory to T via an output state of its block; and all output states of A in an ST-IBC partition, are mutually reachable). If (1) $y \in A \cap B$, the above claim holds; otherwise if (2) $y \in A - (A \cap B)$, then the path within A from x to y of the form $x \to x_1 \to \ldots \to x_k \to y$ must have a one step transition leaving B of the form $b \to a$, where $a \in A - (A \cap B), b \in A \cap B$ and $b, a \in \{x, x_1, \ldots x_k, y\}$. Thus, $b \in O(B, S, T)$, so the conclusion follows again.

Now consider the case $I(A \cup B, S, T) \neq \emptyset$ and $O(A \cup B, S, T) \neq \emptyset$ (otherwise, $A \cup B$ is trivially ST-IBC), therefore, we have two cases to analyse as follows (see Fig. 2.3):
Case (a) $(O(A, S, T) \cap O(B, S, T)) \neq \emptyset$. Let $x \in A \cap B$ be such a common output state of both A and B (see Fig. 2.3(a)). Then, by the mutual accessibility of the output states in ST-IBC blocks, all output states of $O(A, S, T)$ and $O(B, S, T)$ can communicate with each other through x. Hence, the second condition of Definition 2.5 holds. Since $I(A \cup B, S, T) \subseteq I(A, S, T) \cup I(B, S, T)$, every state in $I(A \cup B, S, T)$ can be driven to some state in $O(A, S, T)$ or $O(B, S, T)$ because A and B are ST-IBC; moreover, it may be then driven to some state in $O(A \cup B, S, T) \neq \emptyset$. Hence, it follows that $A \cup B$ is ST-IBC.
Case (b) $(O(A, S, T) \cap O(B, S, T)) = \emptyset$. Without loss of generality, by symmetry we may assume there exists $x \in O(A, S, T) \cap (A \cap B)$ (we have shown above $(O(A, S, T) \cup O(B, S, T)) \cap (A \cap B) \neq \emptyset$) such that $x \notin O(B, S, T)$ (see Fig. 2.3(b)). Now, since the states in $O(A \cup B, S, T) \cap (A \cap B)$ are output states of both A and B,

$x \notin O(A \cup B, S, T)$ by the current hypothesis. Because we have already assumed that all T-co-inaccessible states have been eliminated, $x \in B$ is T-co-accessible implies that there is a trajectory in B from x to some output state of B, say z. Now we need to consider the following two alternative situations:

(b)(1) If all output states of A are inside $A \cap B$, then the current hypothesis of case (b) implies that none of the output states of A can be an output state of $A \cup B$, since such a state would be a common output state of A and B. Because $O(A \cup B, S, T) \neq \emptyset$, $O(A \cup B, S, T) \subseteq O(B, S, T)$. But by the ST-IBC property of A, every input state of A has a trajectory to $x \in O(A, S, T) \cap A \cap B$ and hence to $z \in O(B, S, T)$. Then by the ST-IBC property of B, z has a trajectory to an element of $O(A \cup B, S, T)$. Hence, $A \cup B$ is ST-IBC.

(b)(2) If (b)(1) does not hold, there exist $y \in O(A, S, T) \cap (A - B)$. Clearly, all output states of A are accessible to the elements of $O(B, S, T)$ through x and z, since all output states of A, including x, are mutually reachable by Definition 2.5(b) (see Fig. 2.3(b)). But, by the mutual accessibility of $O(A, S, T)$, it is evident that $x \in O(A, S, T)$ must have a trajectory connecting it to $y \in O(A, S, T) \cap (A - B)$ through some $s \in O(B, S, T) \cap (A \cap B)$. Thus, all output states of B are accessible to y through $s \in O(B, S, T)$ by the mutual accessibility of the output states of a ST-IBC block. Hence, all output states of A and output states of B are mutually accessible with respect to $A \cup B$. Furthermore, since A and B all are ST-IBC, their input states have in-block trajectories leading to their output states and hence to a state in $O(A \cup B, S, T) \neq \emptyset$. It follows again that $A \cup B$ is ST-IBC.

By induction on n, the chain union $\pi_1 \cup^C \pi_2$ defined in Definition 2.6 is ST-IBC.

Furthermore, $\pi_1 \cup^C \pi_2$ is the least upper bound of π_1 and π_2 in the collection of ordinary partitions ordered by $\preceq$, and, since it is ST-IBC, it is also the least upper bound of π_1 and π_2 in the $\pi_{ST}^{IBC}(X)$. □

In the light of Definition 2.5, a block containing only one state must be ST-IBC, since that state is either an input and output state, or one (or both) of the input or output (singleton) sets of this (singleton) block is the empty set. Thus, the partition $\pi^{id} = X$ must be an ST-IBC partition. This partition π^{id} acts as a lower bound of all ST-IBC partitions.

Theorem 2.3. *For two ST-IBC partitions $\pi_1, \pi_2 \in \pi_{ST}^{IBC}(M_t)$ of an ST finite state machine, the greatest lower bound $\pi_1 \sqcap \pi_2 \in \pi_{ST}^{IBC}(M_t)$ exists.* □

The proof is obtained by use of Theorem 2.2, in particular, the existence of the greatest lower bound of π_1 and π_2 (π_1 and π_2 are ST-IBC) is established by noting that (i) the trivial partition (i.e., the partition of singletons) lies in the set of ST-IBC partitions, and (ii) $\pi_1 \cup^C \pi_2$ is the least upper bound of π_1 and π_2 ([**22**]).

We extend the partial order of partitions to their corresponding partition machines by defining $M^{\pi_1} \preceq M^{\pi_2}$ if and only if $\pi_1 \preceq \pi_2$. The following theorem is a straightforward result of Theorem 2.2 and Theorem 2.3.

Theorem 2.4. *All ST-IBC partition machines of an ST finite state machine M_t, ordered by $\preceq$, form a lattice $\langle M_{ST}^{IBC}(M_t), \preceq, \cup^C, \sqcap\rangle$, denoted by $HIBC_{ST}$, which takes the machine $M_t^{id} = M_t$ as its bottom element. In case M_t is ST-IBC, then $HIBC_{ST}$ has as top element the trivial partition M_t^{tr}.* □

2.4. Hierarchical Control for ST-Systems

As stated in the introduction, a feature of the ST-IBC lattice structure for any ST-system M is that it permits the construction of all possible sets of hierarchical feedback control systems (for ST-controllability problems) for the given machine.

To be specific, parallel to the standard IBC hierarchical control problem (see [**17**]), once the underlying ST-IBC lattice of the thinned machine M_t has been constructed, one may select any chain C from the base element M_t^{id} to the top element in the ST-IBC lattice $HIBC_{ST}$. (This may or may not be the trivial partition machine depending on whether T is a mutually accessible set.). Then any set of partition machines lying along such a chain is called an *ST-hierarchical control structure.* Concerning such a control structure we have the following theorem which is readily verified using the results established above.

Theorem 2.5. *For an ST-Controllable finite state machine M_t, consider any pair of distinct elements $M_t^{\pi_1}$ and $M_t^{\pi_2}$, $M_t^{\pi_1} \preceq M_t^{\pi_2}$, in a hierarchical control structure $\langle M_{ST}^{IBC}(M_t), \preceq, \cup^C, \sqcap\rangle$; then $M_t^{\pi_2}$ is ST-BBC and ST-IBC with respect to $M_t^{\pi_1}$. Further, any (necessarily solvable) state to state ST-controllability problem for M_t has a decomposition into a set of recursively defined, solvable, block to block ST-controllability problems for a sequence of machine pairs $M_t^{\pi_n}$ and $M_t^{\pi_{n+1}}$, $1 \leq n \leq$*

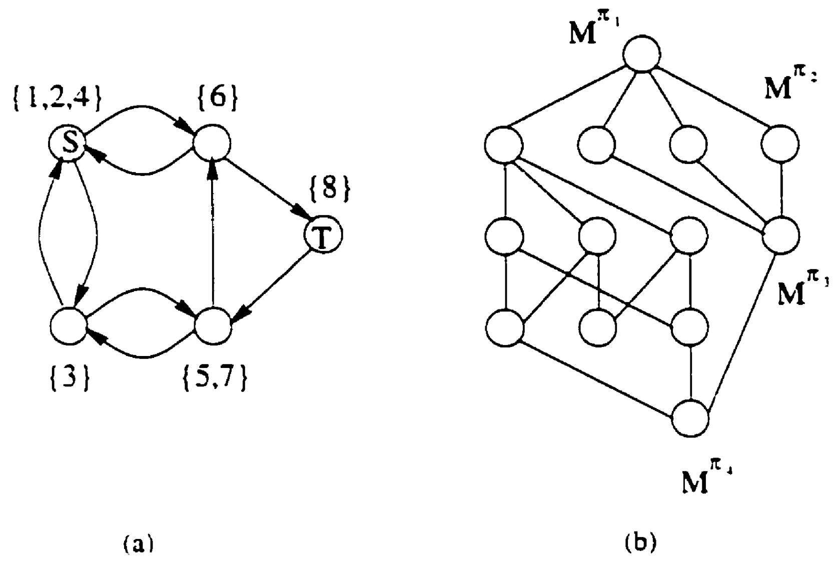

FIGURE 2.4. The 5-state partition machine M_5^π of M_8 and its ST-IBC lattice

$N - 1$, *corresponding to the elements of the N-level hierarchical control structure* $(M_{ST}^{IBC}(M_t), \preceq, \sqcup^C, \sqcap)$. *A (hierarchical family of) solution (trajectories) to this set of problems gives a solution to the original ST-controllability problem.* □

Theorem 2.5 shows that any ST-controllability problem may be decomposed into a sequence of hierarchical control problems: these are such that the feedback controller at any level steers the level-n aggregated state (i.e. level-n partition machine state, containing the base level system state) along a trajectory solving the level-n partition T-reachability control problem.

We observe that there may well be a wide choice of chains in any given ST-IBC lattice and that this consequently facilitates the design of a hierarchical control system: on the other hand, any given machine M has only one ST-IBC lattice and to alter it the dynamics of the base machine M must themselves be altered.

Example 2.3. Let us examine the 5-state machine shown in Figure 2.4.(a): this is a partition machine of the model M_8, in Example 2.1, and we choose $S = \{1, 2, 4\} \in M_5, T = \{8\}$. We notice that M_8 is already trimmed with respect to T and that its corresponding ST-IBC lattice is given in Figure 2.4.(b).

In this lattice the rightmost chain of partitions from the top to the bottom of the lattice is given by:
$\pi_1 = X_{tr} = \{\{\{1,2,4\},\{3\},\{5,7\},\{6\},\{8\}\}\}$; $\pi_2 = \{\{\{1,2,4\},\{3\}\},\{\{5,7\},\{6\},\{8\}\}\}$;
$\pi_3 = \{\{1,2,4,\},\{3\},\{\{5,7\},\{6\},\{8\}\}\}$; $\pi_4 = X_5 = \{\{1,2,4\},\{3\},\{5,7\},\{6\},\{8\}\}$.

In this chain we shall choose the sub-chain π_1, π_2, π_4 as the hierarchical control structure. In π_1, the trivial partition, we know that $S^{\pi_1} = T^{\pi_1} = \pi_1$. Noting the containment relations, we have $S = S^{\pi_4} = \{1,2,4\} \subset S^{\pi_2} = \{\{1,2,4\},3\} \subset S^{\pi_1} = \pi_1$ and $T = T^{\pi_4} = \{8\} \subset T^{\pi_2} \subset T^{\pi_1} = \pi_1$. Since the partition π_1 is ST-IBC, there is an internal trajectory from S^{π_1} to T^{π_1}.

For the partition machine of π_2, we know $\langle\{\{1,2,4\},3\},\{\{5,7\},6,8\}\rangle$ is DC (relative to π_4), $\{\{1,2,4\},3\} \rightarrow \{\{5,7\},6,8\}$. Now the $\mathcal{M}^{\pi_2}$ controller chooses the unique one step control event to drive S^{π_2} to T^{π_2}. This is realized (at the next level of the control hierarchy) by the $\mathcal{M}^{\pi_4}$ controller which has the choice of driving $S^{\pi_2} = \{\{1,2,4\}\}$ to $\{6\}$ in one step, or S^{π_2} to $\{\{5,7\}\}$ via $\{3\}$. As far as the realization of this hierarchical control law is concerned, the choice is arbitrary and may be determined by any well defined rule.

At the finest (π_4) level, the controller terminates the path to T^{π_4} by finding a path from $\{5,7\}$ to $T = \{8\}$. Finally, expressing the corresponding ST-DC relations as the state to state one step transitions at the finest (π_4) level, we obtain $\{1,2,4\} \rightarrow \{3\} \rightarrow \{5,7\} \rightarrow \{6\} \rightarrow \{8\}$ that solves the ST-reachability problem. □

It is worth noting that all the results in this chapter apply to those machines and partitions in which the state set and in-set of the base machine are countably infinite and the set of blocks of any partition has finite cardinality.

CHAPTER 3

Hierarchically Accelerated Dynamic Programming

3.1. Introduction

We consider finite state machines with transition costs $M = \{X, \Sigma, \delta, l\}$, where $X = \{x_0, x_1, x_2, x_3, \ldots x_T\}$ is a finite state space. Σ denotes a finite alphabet of events (controls). δ is a (partial) state transition function defined on $X \times \Sigma$, and the cost function $l : x \times u \to (0, \infty)$ associates each state and control action with a strictly positive real value. The minimisation of the additive cost along all possible paths (i.e. trajectories) between any two given states is a basic problem in many contexts, and dynamic programming (DP) is well known to be a fundamental technique for its solution.

Let the cost index $d(s, t; u, l)$ be defined by

$$d(s, t; u, l) = \sum_{i=1}^{m-1} l(x_i, u_i),$$

where $s = x_1, t = x_m, x_{i+1} = \delta(x_i, u_i), 1 \leq i \leq m-1$. We shall denote by u^{00} any optimal control minimising the cost index $d(s, t; u, l)$ over all control sequences u of all lengths m such that t is accessible from s. If we represent a given finite state machine as a directed graph, this problem is a (weighted edge) shortest path problem. A large number of algorithms have been developed to solve shortest path problems, including Dijkstra's algorithm, Ford's algorithm and Dijkstra's Two-tree algorithm (see [**25**]), to name a few. The time complexity of these algorithms naturally depends upon the size of the model: in general, the complexity grows non-linearly with the number of states in the system, and so the computational efficiency degrades significantly and in particular, they are not applicable to real-time problems. We propose a hierarchical approach to dynamic programming problems which, at the cost of a degree of sub-optimality, and subject to an initial investment in constructing a control hierarchy, may reduce the computational complexity of solving any given shortest path problem.

The work here is founded upon that in [**17**], where hierarchical control systems are formulated in terms of the construction of *partition machines* via the notion of *dynamically consistent (DC)* state aggregation, and on that in [**15**], where the generalisation of the previous results to source-target systems was presented. The HADP methodology proceeds by first decomposing a finite state dynamical system into hierarchical layers of partition machines. Each trajectory optimisation problem (between a source-target pair in the base system with respect to the additive cost function) is represented at the next higher aggregated level using a specifically constructed cost function: this process is iterated up to the highest defined aggregation level (for simplicity of exposition, we only consider two hierarchical layers in Sections 3.2, 3.3 and 3.4 of this chapter). Tangibly related work can be found in [**7**], where the authors proposed an aggregate DP for acyclic networks without discussing the consistency of high and low level models. Finally, the dynamic programming (DP) solution to the resulting highest level problem is then passed down to the next lower layer. In that layer, a set of corresponding DP problems is solved, one in each of the blocks lying along the previously derived optimal high level path. This process terminates at the bottom level of the hierarchy. The analysis in Section 3.3 of this chapter gives conditions to ensure that the procedure yields optimal, or near optimal, solutions to the original base level trajectory optimisation problem. In Section 3.4, we provide estimates of the sub-optimality of the HADP method when the optimality conditions of Section 3.3 fail.

3.2. Hierarchical Control and Control Consistency

In this section, we give formal definitions of the notions of dynamical consistency ([17], [15]) and control consistency (analogous to that defined in [68] and [70]) for a class of hierarchical finite state machines.

Definition 3.1. (Dynamical Consistency (DC)) An ordered pair of partition elements (blocks) $\langle X_1, X_2 \rangle$ is said to be *dynamically consistent (DC)* (with respect to π) if for all $x \in X_1$, there exists $u \in \Sigma^*$ such that $\delta(x, u) \in X_2$ and for all $v < u$, $\delta(x, v) \in X_1$. □

A two level hierarchy formed by M and its abstraction model $M_h = \{X_h, U_h, \delta_h\}$, where $X_h = \pi$, U_h is a set of high level control symbols, δ_h is the abstract high level state (block) transition function, is denoted by $\{M, M_h\}$.

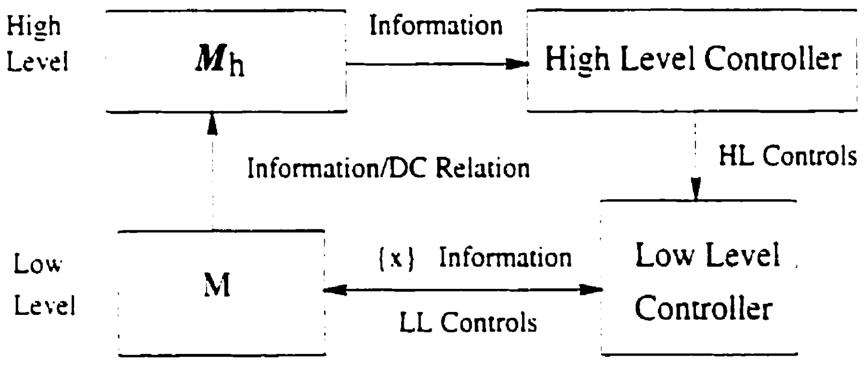

FIGURE 3.1. Hierarchical control structure

Assume that we have an abstraction $M_h = \{\pi, \Sigma_h, \delta_h\}$ of the given base model M. In the two level structure shown in Figure 3.1, the hierarchical control is carried out in a top-down fashion layer by layer. The high level controller employs the information provided by M_h. The function of M_h is to simplify M while preserving its critical behavioural properties. This hierarchical control configuration achieves an objective in the following way: first, the low level control task is communicated to the high level, then the high level control is performed and the corresponding high level (abstract) control commands are passed to the low level controller. The function of low level controllers is simply to realise the high level abstract commands in terms of specific low level state transitions.

For simplicity, we denote a trajectory from x driven by u as $Trj(x, u)$ and a high level trajectory from $X_i \in \pi$ driven by U as $Trj(X_i, U)$. If (1) for all $v < w < u$, there exist $V \leq W \leq U$ such that $\delta(x, v) \in \delta_h(X_1, V)$ and $\delta(x, w) \in \delta_h(X_1, W)$, and (2) for all $V \leq W \leq U$, there exist $v < w < u$ such that $\delta(x, v) \in \delta_h(X_1, V)$ and $\delta(x, w) \in \delta_h(X_1, W)$, the trajectory $Trj(x, u)$ is said to be contained in $Trj(X_i, U)$.

Definition 3.2. (Control Consistency) A two level hierarchy $\{M, M_h\}$ is said to be *control consistent (CC)* if and only if the following two accessibility conditions hold:
(1) For any $x \in X_i \in \pi$ and $y \in X_j \in \pi$ if there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$ then there exists $U \in \Sigma_h^*$ such that $\delta_h(X_i, U) = X_j$ and $Trj(x, u)$ is contained in $Trj(X_i, U)$.
(2) For all $X_i, X_j \in \pi$, if there exists $U \in \Sigma_h^*$ such that $\delta_h(X_i, U) = X_j$, then for all $x \in X_i$, there exists $y \in X_j$ and $u \in \Sigma^*$ such that $\delta(x, u) = y$ and $Trj(x, u)$ is contained in $Trj(X_i, U)$. $\square$

This property is analogous to that defined in [70] in the hierarchical supervisory control context: it ensures that if a low level task can be completed by a sequence of transitions, then the controlled dynamics of the high level system are consistent with these transitions: conversely, if the high level controller steers a high level state to its target, then the high level commands can be translated into realisable low level transitions.

Theorem 3.1. *A two level hierarchy $\{M, M_h\}$ is control consistent (CC) if and only the following hypotheses (H) are true:*
(1) If there exist $x \in X_i \in \pi, y \in X_j \in \pi, u \in \Sigma$ such that $\delta(x, u) = y$ and $Trj(x, u)$ is contained in $X_i X_j$, then $\langle X_i, X_j\rangle$ is dynamically consistent.
(2) For all $X_i, X_j \in \pi$, if there exists $U \in U_h$, such that $\delta_h(X_i, U) = X_j$, then $\langle X_i, X_j\rangle$ is dynamically consistent.
(3) For all $X_i, X_j \in \pi$, if $\langle X_i, X_j\rangle$ is dynamically consistent, then there exists $U \in U_h$, such that $\delta_h(X_i, U) = X_j$.

Proof:

$H \Rightarrow CC$
Suppose the statements (1)-(3) hold, we shall prove that $\{M, M_h\}$ possesses the two properties of control consistency.

(1) For any $x \in X_j \in \pi$ and $y \in X_j \in \pi$, if there is $u \in \Sigma^*$ such that $\delta(x, u) = y$, denote $Trj(x, u) = z_{1,1}...z_{1,m_1}z_{2,1}...z_{n,1}...z_{n,m_n}$, where $z_{1,1} = x$, $z_{1,2}, ...z_{1,m_1} \in Y_1 = X_i$, $z_{2,1}, ..., z_{2,m_2} \in Y_2$, ..., and $z_{n,1}, ...z_{n,m_n} \in Y_n = X_j$. It is clear that u can be rewritten in terms of $u = u_1u_2...u_{n-1}$ with each $u_i \in \Sigma^*$ such that $\delta(x, u_1) = z_{2,1}$, $\delta(z_{2,1}, u_2) = z_{3,1}$, ..., and $\delta(z_{(n-1),1}, u_{n-1}) = y$. Moreover, we know that $Trj(x, u_1)$ is contained in X_iY_2, $Trj(z_{2,2}, u_2)$ is contained in Y_2Y_3, ..., $Trj(z_{n-1,1}, u_{n-1})$ is contained in $Y_{n-1}X_j$. By statement (1), $\langle X_i, Y_2\rangle, \langle Y_2, Y_3\rangle, ..., \langle Y_{n-1}, X_j\rangle$ are dynamically consistent and thus by statement (3), there are $U_1, U_2, ..., U_{n-1} \in \Sigma_h$ such that $\delta_h(Y_k, U_k) = Y_{k+1}$ for $1 \leq k \leq n-1$. Therefore, $\delta_h(X_i, U_1U_2 \cdots U_{n-1}) = X_j$, and $Trj_h(X_i, U_1U_2 \cdots U_{n-1}) = X_iY_2...X_j$ contains $Trj(x, u)$).

(2) For any $X_i, X_j \in \pi$, if there is $U \in \Sigma_h^*$ such that $\delta_h(X_i, U) = X_j$, then we denote $Trj_h(X_1, U) = Y_1Y_2...Y_{n-1}Y_n$ with $Y_1 = X_i$ and $Y_n = X_j$. So, U may be decomposed into $U_1U_2 \cdots U_{n-1}$ such that $\delta_h(Y_k, U_k) = Y_{k+1}$ for $1 \leq k \leq n-1$. By statement (2), $\langle Y_k, Y_{k+1}\rangle$ are dynamically consistent, $1 \leq k \leq n-1$. If $x \in X_i$ and $y \in X_j$ are arbitrarily given, according to the definition of dynamical consistency, there are respective $u_k \in \Sigma^*$ and $x_k \in Y_k$, $2 \leq k \leq n-1$ such that $\delta(x, u_1) = x_1, \delta(x_k, u_i) = x_{k+1}, 2 \leq k \leq n-2, \delta(x_{n-1}, u_{n-1}) = y$, and $Trj(x, u_1)$ is contained Y_1Y_2, $Trj(x_k, u_k)$ is contained in $Y_iY_{k+1}, 2 \leq k \leq n-1$. Hence, $\delta(x, u_1u_2 \cdots u_{n-1}) = y$ and $Trj(x, u_1u_2 \cdots u_{n-1})$ is contained in $Trj(X_i, U)$. In short, the two conditions for $\{M, M_h\}$ to be CC are satisfied.

CC ⇒ H

This part follows straightforwardly from the relevant definitions. □

On the basis of this theorem, we conclude that the dynamics of the high level model in a control consistent hierarchy are coincidental with the dynamical consistency relations on the partition of the state space of the base model. Thus, we define the dynamics of M^π based on the dynamical consistency relations on $\pi \times \pi$. To be specific, in $M^\pi = \{\pi, \Sigma^\pi, \delta^\pi\}$, if $\langle X_i, X_j\rangle$ is DC, then there exists $U_i^j \in \Sigma^\pi$ such that $\delta^\pi(X_i, U_i^j) = X_j$. It is to be noted that Σ^π is a set of high level abstract control symbols and is not a set of implementable (base level) control actions.

Definition 3.3. **(IBC Block)** A partition block $X_i \in \pi$ is *in-block controllable (IBC)* if for all $x, y \in X_i$ there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$ and for all $v < u$, $\delta(x, v) \in X_i$. □

A partition π is an IBC partition if all its blocks are IBC. A theory of the structure of partition machines for hierarchical control ([**65**], [**17**], [**15**]) and its extensions to the hybrid and supervisory control cases respectively are to be found in [**20**], [**16**], [**13**] and [**32**]. In particular, this theory asserts that all IBC partition machines of M constitute a lattice L and any chain from the top to the bottom of L provides a hierarchical control structure, where the terms base (level) and high level machine have an obvious implication.

Consider a base system M which may be represented by a directed graph. We observe that if X_i is an IBC block, then for any $x, y \in X_i$ there exists $uv \in \Sigma^*$ such that $\delta(x, u) = y, \delta(y, v) = x$ and for all $w < uv$, $\delta(x, w) \in X_i$. This fact reveals that IBC partitions may be generated locally, through a search for circuits. We may first partition M into m subgraphs G_i, $i = 1, 2, \ldots, m$. The following algorithm recursively gives the maximal IBC block which contains a given state s in a subgraph G_i.

Algorithm for generating the maximal IBC block $X(s)$ containing s in G_i
(1) Set $X(s) = \{s\}$.
(2) If there is a path $x_1x_2...x_n$ in G_i which originates from $x_1 \in X(s)$ and ends at $x_n \in X(s)$, and $x_2, x_3, \ldots, x_{n-1} \notin X(s)$, then $X(s) = X(s) \bigcup \{x_2, x_3, \ldots, x_{n-1}\}$;
(3) else, stop. □

This algorithm has the same time complexity as depth-first search ([**25**]).

Definition 3.4. **(Controllability)** A finite state machine $M = \{X, \Sigma, \delta\}$ is *controllable* if for any $x, y \in X$, there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$. □

The controllability of M^π is similarly defined.

Theorem 3.2. [**17**] *If π is IBC, then M^π is controllable if and only if M is controllable.* □

Finally, we conclude the following result.

Theorem 3.3. *If π is IBC, then $\{M, M^\pi\}$ is control consistent.*

Proof:
Suppose $\pi = \{X_1, X_2, \ldots, X_{|\pi|}\}$ is IBC and M^π is a partition machine of M based on π.
(1) Let $x \in X_i \in \pi$ and $y \in X_j \in \pi$ be two arbitrary states of M. If there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$, let $u = u_0 u_1 \ldots u_m$ ($u_i \in \Sigma$ for $0 \leq i \leq m$), then $Trj(x, u) = x z_1 z_2 \ldots z_m y$, with $\delta(x, u_0) = z_1$, $\delta(z_m, u_m) = y$ and $\delta(z_i, u_i) = z_{i+1}$, $1 \leq i \leq m-1$. Suppose $z_i \in X_{n_i}$, for some integers $1 \leq n_i \leq |\pi|$. For convenience, set $z_0 = x$, $z_{m+1} = y$ and $n_0 = i$, $n_{m+1} = j$.

Whenever $X_{n_i} \neq X_{n_{i+1}}$, it is the case that $z_i \in X_{n_i}$ and $\delta(z_i, u_i) = z_{i+1} \in X_{n_{i+1}}$. By the assumption that π is IBC, X_{n_i} is IBC, i.e. for any state $x' \in X_{n_i}$, there exists $v \in \Sigma^*$ such that $\delta(x', v) = z_i$ and for all $v' < v, \delta(x', v') \in X_{n_i}$. Let $w = v u_i$, then $\delta(x', w) = \delta(z_i, u_i) \in X_{n_{i+1}}$, and for all $w' < w, \delta(x', w') \in X_{n_i}$. In other words, $\langle X_{n_i}, X_{n_{i+1}} \rangle$ is DC and this holds for all n_i. Therefore, there is $U_i \in \Sigma^\pi$ such that $\delta^\pi(X_{n_i}, U_i) = X_{n_{i+1}}$.

If $X_{n_i} = X_{n_{i+1}}$, let $U_i = \epsilon$, the empty string over the Σ^π, we also have $\delta^\pi(X_{n_i}, U_i) = X_{n_{i+1}}$.

Hence, there exists a string in $(\Sigma^\pi)^*$, i.e. $U = U_0 U_1 \ldots U_m$ such that $\delta^\pi(X_i, U) = X_j$ and clearly, $Trj(x, u)$ is contained in $Trj(X_i, U)$.

(2) Let X_i, X_j be two arbitrary blocks in π. Suppose $U \in (\Sigma^\pi)^*$ is a string of high level control symbols such that $\delta^\pi(X_i, U) = X_j$. Denote $U = U_1 U_2 \ldots U_m$ with $U_k \in \Sigma$, $1 \leq k \leq m$. For simplicity, let $n_1 = i$, $n_{m+1} = j$, $\delta^\pi(X_{n_i}, U_i) = X_{n_{i+1}}$, where $X_{n_i} \in \pi$ and n_i is some integer, $1 \leq n_i \leq |\pi|$.

For any two states $x \in X_i$ and $y \in X_j$, we now show there is a $u \in \Sigma^*$ such that $\delta(x, u) = y$. Since $\delta^\pi(X_{n_i}, U_i) = X_{n_{i+1}}$, i.e. $\langle X_{n_i}, X_{n_{i+1}} \rangle$ is DC, then we know there exists $x_2 \in X_{n_2}$ and there is $u_1 \in \Sigma^*$ such that $\delta(x, u_1) = x_2$ and for all $u_2' < u_2$,

$\delta(x, u_2') \in X_{n_1} = X_i$. Subsequently, we may take $x_i \in X_{n_i}$ for $2 < i \leq m+1$ such that there exist $u_i \in \Sigma^*$, $\delta(x_i, u_i) = x_{i+1}$ and $Trj(x_i, u_i)$ is contained in $X_{n_i}X_{n_{i+1}}$. Let $v = u_1u_2...u_m$, then $\delta(x, v) = x_{m+1} \in X_{n_{m+1}} = X_j$. Moreover, because π is IBC, X_j is IBC. This implies that there exists $w \in \Sigma^*$ such that $\delta(x_{m+1}, w) = y$ and for all $w' < w$, $\delta(x_{m+1}, w') \in X_j$. Denote $u = vw$, clearly, $\delta(x, u) = y$ and $Trj(x, u)$ is contained in $Trj(X_i, U) = X_{n_1}X_{n_2}...X_{n_{m+1}}$.

□

In this chapter, unless otherwise indicated, all partitions π are assumed to be IBC partitions and $M_h = \{X_h, U_h, \delta_h\}$ will be used to denote the partition machine M^π corresponding to π.

3.3. Optimality Consistency

An optimal control with respect to a start state x and a target state y, $u^{00}(x, y)$ is a sequence of transitions such that $y = \delta(x, u^{00}(x, y))$ and for all $v \in \Sigma^*$ with $\delta(x, v) = y$, $d(x, y; u^{00}(x, y), l) \leq d(x, y; v, l)$. Denote the set of optimal controls with respect to the start state x and the target state y by $\mathbf{u}^0(x, y)$. Hence, if $v \notin \mathbf{u}^0(x, y)$, $d(x, y; u^{00}(x, y), l) < d(x, y; v, l)$.

Denote the set of high level optimal controls with respect to the start block $X_i \in \pi$ and the target block $X_j \in \pi$ by $\mathbf{U}^0(X_i, X_j)$.

Definition 3.5. (Optimality Consistency (OC(C_h))) A two level hierarchy $\{M, M_h\}$ with a given low level cost function l is *optimality consistent (OC)* if there exists a high level cost function $C_h : \pi \times \Sigma_h \mapsto R^+$ such that the following conditions hold:
(1) for any two states $x \in X_i \in \pi, y \in X_j \in \pi$, if $u \in \mathbf{u}^0(x, y)$, then there is $U \in \mathbf{U}^0(X_i, X_j)$ such that $Trj(x, u)$ is contained in $Trj(X_1, U)$.
(2) for any two blocks $X_i, X_j \in \pi$, if $U \in \mathbf{U}^0(X_i, X_j)$, then for every $x \in X_i$ and $y \in X_j$, there exists $u \in \mathbf{u}^0(x, y)$ such that $Trj(x, u)$ is contained in $Trj(X_i, U)$. □

Definition 3.6. (Convex High Level Trajectories) A high level trajectory $P(X_1^n) = X_1X_2 \cdots X_n$ is said to be *convex* if either:

(1) for all $x \in X_1$ and all $y \in X_n$, $P(X_1^n)$ does not contain any low level optimal path from x to y.
(2) or for all $x \in X_1$ and $y \in X_n$, $P(X_1^n)$ contains a low level optimal path from x to y.

A high level model M_h is said to be *convex* if all of its paths are convex. □

We note that the convexity property does not imply the low level optimal path between a pair of states is unique; nor does it imply that the high level path containing low level optimal paths between a pair of states is unique.

All states in a block X_i can be classified into two disjoint categories: (i) boundary states ∂X_i which have direct connections to or from elements in X^c, and (ii) interior states $\overset{o}{X}_i$ which are elements of X_i not in ∂X_i, i.e. $\overset{o}{X}_i = X_i - \partial X_i$. It turns out that the convexity of paths is exclusively determined by their boundary states. This is formulated by the theorem below.

Theorem 3.4. *A high level path $P(X_1^n) = X_1 X_2 ... X_n$ containing a low level globally optimal path from a state in X_1 to a state in X_n is convex if and only if for all $x \in O(X_1)$ and $y \in I(X_n)$, there exists a low level globally optimal path from x to y path-wise contained in $P(X_1^n)$.*

Proof:

First, we prove the sufficiency. Because π is IBC, the hierarchy $\{M, M_h\}$ is CC. Therefore, for any state $x' \in X_1$ and any state $y' \in X_n$, there is a low level path contained in $P(X_1^n)$ connecting x' and y'. Because, the set of controls with respect to x' and y' is not empty, the optimal control exists. Denote the low level global optimal path from x' to y' by $Trj(x', u)$. There is a state $x'' \in O(X_1)$ such that $Trj(x', u)$ can be written as $x' \rightsquigarrow x'' \rightarrow z \rightsquigarrow y'$ where all states on $z \rightsquigarrow y'$ are not in X_1. Also, there is a state $y'' \in I(X_n)$ such that $Trj(x', u)$ can be written as $x' \rightsquigarrow z' \rightarrow y'' \rightsquigarrow y'$ where all states on $x' \rightsquigarrow z'$ are not in X_1. By the condition that there is a low level globally optimal path from x' to y' path-wise contained in $P(X_1^n)$, denote this path as $Trj(x'', v)$. Hence, $x' \rightsquigarrow Trj(x'', v) \rightsquigarrow y'$, which is path-wise contained in $P(X_1^n)$, has the same cost as $Trj(x', u)$ and thus is optimal. Since x' and y' are arbitrary, we conclude that $P(X_1^n)$ is convex.

"Only if" part is straightforward by the definition of convexity. □

Theorem 3.5. *If a two level hierarchy $\{M, M_h\}$ is optimality consistent then $\{M, M_h\}$ is convex.*

Proof:

Suppose a two level hierarchy $\{M, M_h\}$ is OC(C_h). Let $T = X_i X_{k_2} X_{k_3} \ldots X_j$ be an arbitrary trajectory of M_h. In the non-empty case, if $x \in X_i$ and $y \in X_j$ are two states of M and $u \in \mathbf{u}^0(x, y)$ is an optimal low level control sequence such that $Trj(x, u)$ is contained in T, then, by Definition 3.1(1), we know there exists $U \in \mathbf{U}^0(X_i, X_j)$ such that $Trj(x, u)$ is contained in $Trj(X_1, U)$. That is to say, $T = Trj(X_i, U)$ is an optimal high level path from X_i to X_j.

Since T is an optimal high level path, by Definition 3.8(2), for any two states $x' \in X_1$ and $y' \in X_2$, there exists $u' \in \mathbf{u}^0(x', y')$ such that $Trj(x', u')$ is contained in T. In other words, there is an optimal low level path from x' to y' contained in T. Since T is arbitrary, it follows that $\{M, M_h\}$ is convex. □

However, convexity does not imply optimality consistency, as is shown in the following example.

Example 3.1. In Figure 3.2 is shown a low level model M. Each edge is labelled by the cost of its corresponding transition. An IBC partition machine of M, M_h, is given by Figure 3.3. It is easy to verify that all (high level) paths of M_h are convex. However, as we shall show, no admissible high level cost function exists to make the hierarchy $\{M, M_h\}$ optimality consistent.

Suppose to the contrary, there was a high level cost function C_h such that $\{M, M_h\}$ is OC(C_h) and the cost of transition from X_i to X_j for a DC pair $\langle X_i, X_j \rangle$ in M_h is denoted by $C(X_i, X_j)$.

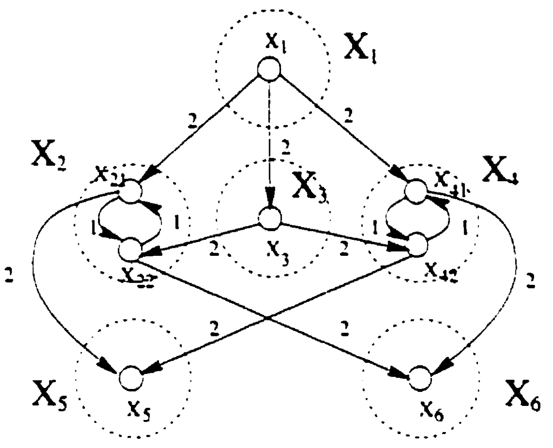

FIGURE 3.2. A finite state machine M

By the definition of optimality consistency, a high level path containing a low level optimal path from a state in its first block to a state in its last block is optimal with respect to C_h. To be specific, $x_1 \to x_{21} \to x_5$ (contained in $X_1 \to X_2 \to X_5$) is optimal and $x_1 \to x_{41} \to x_{42} \to x_5$ (contained in $X_1 \to X_4 \to X_5$) is non-optimal, hence

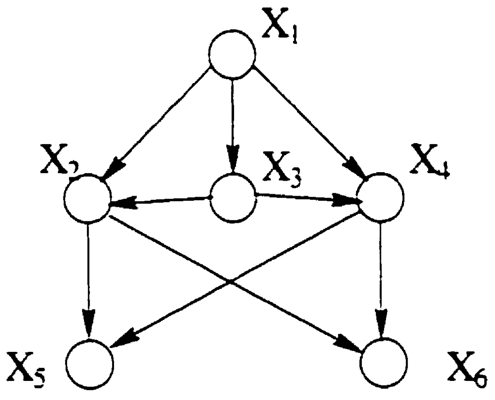

FIGURE 3.3. An IBC partition machine M_h

$$C(X_1, X_2) + C(X_2, X_5) < C(X_1, X_4) + C(X_4, X_5). \tag{1}$$

$x_1 \to x_{41} \to x_6$ (contained in $X_1 \to X_4 \to X_6$) is optimal and $x_1 \to x_{21} \to x_{22} \to x_6$ (contained in $X_1 \to X_2 \to X_6$) is non-optimal, thus,

$$C(X_1, X_4) + C(X_4, X_6) < C(X_1, X_2) + C(X_2, X_6). \tag{2}$$

(1) and (2) jointly give

$$C(X_2, X_5) + C(X_4, X_6) < C(X_4, X_5) + C(X_2, X_6). \tag{3}$$

Similarly. $x_3 \to x_{42} \to x_5$ (contained in $X_3 \to X_4 \to X_5$) is optimal and $x_3 \to x_{22} \to x_{21} \to x_5$ (contained in $X_3 \to X_2 \to x_5$) is non-optimal. hence

$$C(X_3, X_4) + C(X_4, X_5) < C(X_3, X_2) + C(X_2, X_5). \tag{4}$$

Now $x_3 \to x_{22} \to x_6$ (contained in $X_3 \to X_2 \to X_6$) is optimal and $x_3 \to x_{42} \to x_{41} \to x_6$ (contained in $X_3 \to X_4 \to X_6$) is non-optimal. thus.

$$C(X_3, X_2) + C(X_2, X_6) < C(X_3, X_4) + C(X_4, X_6). \tag{5}$$

(4) and (5) jointly give

$$C(X_4, X_5) + C(X_2, X_6) < C(X_2, X_5) + C(X_4, X_6). \tag{6}$$

Obviously. (6) contradicts (3). Thus C_h can not exist. □

For $x \in X_1 \in \pi$ and $y \in X_n \in \pi$ and a high level trajectory $T = X_1 X_2 \cdots X_n$. let $\mathbf{u}_T(x, y)$ be a set of sequences of low level control actions satisfying: $v \in \mathbf{u}_T(x, y) \Rightarrow \delta(x, v) = y$ and $Trj(x, v)$ is contained in T.

Definition 3.7. (**Minimal Cost** $d^0_T(x, y)$) For any $x, y \in X$ and a state and control cost function l. we may define
$d^0_T(x, y) = d^0_T(x, y; l) \triangleq \min_v \{d(x, y; v, l) : v \in \mathbf{u}_T(x, y)\}$. □

The optimal low level cost $d(x, y; u, l)$. where $u \in \mathbf{u}^0(x, y)$. shall be denoted by $d^{00}(x, y)$. and this shall be referred to as the *global optimum value* of $d(x, y; u, l)$.

We may define a less restrictive version of optimality consistency, which is that of optimality consistency with respect to an initial state s:

Definition 3.8. (Optimality Consistency w.r.t. an Initial State s) A two level hierarchy $\{M, M_h\}$ with a given low level cost function l is said to be *optimality consistent with respect to an initial state* $s \in X^s \in \pi$ if there exists a high level cost function $C_h : \pi \times \Sigma_h \mapsto R^+$ such that the following conditions hold:
(1) For all $t \in X_t \in \pi$, if $u \in \mathbf{u}^0(s,t)$, then there exists $U' \in \mathbf{U}^0(X^s, X_t)$ such that $Trj(s,u)$ is contained in $Trj(X^s, U')$, and
(2) For all $t \in X_t$, if $U' \in \mathbf{U}^0(X^s, X_t)$, then there exists $u \in \mathbf{u}^0(s,t)$ such that $Trj(s,u)$ is contained in $Trj(X^s, U')$. □

Definition 3.9. (Convexity w.r.t. an Initial State s) A high level model M_h is said to be *convex w.r.t. an initial state* s, where $s \in X^s \in \pi$, if for all paths $P(Y_1^n) = Y_1 Y_2 \cdots Y_n, \subset M_h$ with $Y_1 = X^s$,
$\exists x \in Y_n, \exists u^{00}(s,x) \in \mathbf{u}^0(s,x)$ s.t. $Trj(s,x) \subset P(Y_1^n)$
$\Longrightarrow \forall y \in Y_n, \exists u^{00}(s,y) \in \mathbf{u}^0(s,y)$ s.t. $Trj(s,y) \subset P(Y_1^n)$. □

With this weakened version of convexity, we have,

Theorem 3.6. *If all paths in M_h are convex with respect to s, then $\{M, M_h\}$ is optimality consistent with respect to s.*

Proof:

To prove the theorem, we introduce a graph called the *optimal high-level path graph w.r.t. s* (OPG(X^s)); OPG(X^s) consists of all high level nodes and all edges that are on some high level path from X^s to some node X_t, where these paths each contain a low level optimal path from s to a state in X_t. Given the convexity w.r.t. s of all high level paths starting from X^s, we construct the graph OPG(X^s) by arbitrarily picking a state, say z, in each block and then finding all low level optimal paths from s to z. The high level paths containing these low level optimal paths constitute OPG(X^s).

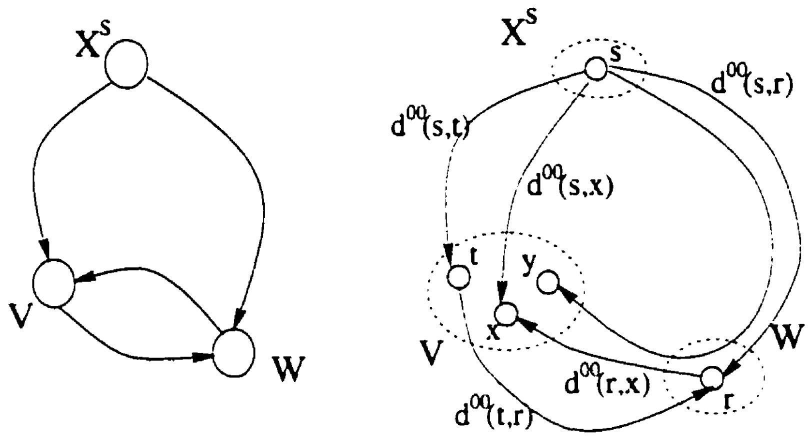

FIGURE 3.4. There are no circuits in OPG(X^s)

We shall prove that there are no circuits in the OPG(X^s) by obtaining a contradiction by use of the convexity hypothesis.

We let $\rightarrow$ represent a one-step transition and let $\rightsquigarrow$ represent a sequence of transitions. Suppose there were a circuit between two distinct nodes V and W in OPG(X^s), namely some path $V \rightsquigarrow W \rightsquigarrow V$. By the definition of OPG(X^s), $V \rightsquigarrow W$ and $W \rightsquigarrow V$ are the terminal parts of high level paths containing low level optimal paths from s to states in W and V respectively. Therefore, there must be low level optimal paths from s to a node $x \in V$ contained in $X^s \rightsquigarrow V$ and another from s to $y \in V$ contained in $X^s \rightsquigarrow W \rightsquigarrow V$ (see Figure 3.4.).

Let $x \in V$ be a state such that the cost from s to x is the minimum with respect to all states in V. By convexity w.r.t s, since $y \in V$, we also conclude that there is a low level optimal path from s via W to x contained in $X^s \rightsquigarrow W \rightsquigarrow V$. Denote a state in W on such a low level optimal path by r. By similar reasoning, we know there is a low level optimal path from s to r contained in $S \rightsquigarrow W$, and there is a low level optimal path from s to r contained in $S \rightsquigarrow V \rightsquigarrow W$. Denote a state in V on the latter path by t.

We know that all segments on a low level optimal path must also be globally optimal trajectories between their end points. Let $d^0_{S\rightsquigarrow V}(s,x)$ denote the cost of an optimal path $s \rightsquigarrow x$ with respect to all low level paths from s to x contained in

$S \leadsto V$, $d^0_{S\leadsto V}(s,x)$ equals to the global optimum $d^{00}(s,x)$. We also know that two optimal paths between the same pair of states have the same cost, i.e.,

$$d^{00}(s,x) = d^0_{X^s\leadsto V}(s,x) = d^0_{X^s\leadsto W\leadsto V}(s,x)$$
$$= d^0_{X^s\leadsto W}(s,r) + d^0_{V\leadsto W}(r,x) = d^{00}(s,r) + d^{00}(r,x) \tag{7}$$

and

$$d^{00}(s,r) = d^0_{X^s\leadsto W}(s,r) = d^0_{X^s\leadsto V\leadsto W}(s,r)$$
$$= d^0_{X^s\leadsto V}(s,t) + d^0_{V\leadsto W}(t,r) = d^{00}(s,t) + d^{00}(t,r) \tag{8}$$

Therefore, substituting (8) into (7), it follows that

$$d^{00}(s,x) = d^{00}(s,t) + d^{00}(t,r) + d^{00}(r,x).$$

Because we have made the assumption that $d^{00}(s,x)$ is minimal with respect to all states in V,

$$d^{00}(s,t) \geq d^{00}(s,x).$$

Hence

$$d^{00}(s,t) \geq d^{00}(s,x) = d^{00}(s,t) + d^{00}(t,r) + d^{00}(r,x),$$

which then gives

$$d^{00}(t,r) + d^{00}(r,x) = 0.$$

This implies $r = t = x$, contradicting $V \neq W$.

We consider two distinct cases where the high level paths that contain low level optimal paths from s to the states in any given block are respectively unique and not unique.

First, let us assume that the high level paths that contain low level optimal paths from s to the states in any given block are unique, then OPG(X^s) is a tree. In this case, circuits are clearly impossible in OPG(X^s). It is quite straightforward to see that by assigning each arc on this tree a cost 1, and all other edges of the high level graph which are not shown in OPG(X^s) with cost infinity (or a sufficiently large positive number), we obtain a high level cost function which preserves the partial order on blocks established by OC w.r.t.s.

Second, consider the general case in which the uniqueness of high level paths containing optimal low level paths does not hold, that is to say, there may be multiple high level paths containing low level optimal paths from s to some given node in their last block. In this case, we invoke the result proved above to conclude there are no circuits in OPG(X^s). To make the costs of the above mentioned high level path equal, we introduce a term *depth*.

Define the depth of each node on OPG(X^s). $dep(S) = 0$, $dep(W) = \max\{dep(V)+1$ for all V which is a direct predecessor of $W\}$. As we have proved, the depth of a node must be finite because no circuit exists in OPG(X^s).

Now we define the cost of the arc $V \to W$ by $C_h(V, V \to W) = dep(W) - dep(V)$, which we note is strictly greater than 0.

For all arcs $V \to W$ not appearing in OPG(X^s), let $C_h(V, V \to W) = \infty$.

It is clear that under the so-defined C_h, the set of optimal high level paths from X^s to $X_i \in \pi$ contains a low level optimal path from s to any state $x \in X_i$; and any low level optimal path from s to an arbitrary state $x \in X_i \in \pi$ is contained in a high level optimal path from X^s to X_i. Thus, we conclude that $\{M, M_h\}$ is optimality consistent with respect to s. □

Let us extend the notion of convexity with respect to a single start state s to that of *convexity with respect to* X^s: given a high level trajectory T in $\{M, M_h\}$, if for all $x \in X^s \in \pi$, T is convex with respect to x, then T is said to be convex with respect to X^s. Similarly, we extend the notion of optimality consistency with respect to s to *optimality consistency with respect to* X^s: a hierarchy $\{M, M_h\}$ is *optimality* *consistent with respect to* $X^s \in \pi$ if $\{M, M_h\}$ is optimality consistent with respect to x, for all $x \in X^s$.

Corollary 3.1. *If all paths in* M_h *are convex with respect to* X^s, *then* $\{M, M_h\}$ *is optimality consistent with respect to* X^s. □

3.4. P-HADP(C_h), HADP(C_h) and Their Sub-optimality Estimates

Recalling the treatment introduced in [**15**], we specify a subset called an in-set for each block of the high level model.

Definition 3.10. (**In-Sets**) For a partition block $X_i \in \pi$ in the partition π, its *input set*, or *in-set*, $I(X_i)$ is a subset of X_i satisfying the following condition:
$\forall x \in I(X_i)$, there exists $x' \notin X_i$, and there exists $u \in \Sigma$ such that $\delta(x', u) = x$.

$I(X_i)$ is further refined according to the DC relations: for a pair of DC blocks $\langle X_j, X_i \rangle$, we set $I_j(X_i) \triangleq \{x \in X_i : \exists x' \in X_j \exists u \in \Sigma . \delta(x', u) = x\}$. □

In particular, if the initial and target states $s \in X_s \in \pi$ and $t \in X_t \in \pi$ are identified, we designate $I(X_s) = \{s\}$ and $I(X_t) = \{t\}$.

A possible measure of the cost of a high level transition between blocks is the total cost in terms of the low level events required to traverse this block from its in-set to the next block. Define $d(x, y; u, l)$ with $x \in X_i$, $y \in I_i(X_j)$, and for all $v < u$, $\delta(x, u) \in X_i$, as a *traverse cost* from X_i to X_j. Because a block may have multiple in-set states, finding a single parameter representing the traverse costs may be impossible. Bearing this in mind, we define the bounds.

Definition 3.11. ($D^+(X_i, X_j)$, $D^{+/-}(X_i, X_j)$, $D^-(X_i, X_j)$) If $\langle X_i, X_j \rangle$ is DC,
$D^+(X_i, X_j) \triangleq \max\{d^0_{X_i X_j}(x, y)$ for all $x \in I(X_i)$ and $y \in I_i(X_j)\}$.
$D^{+/-}(X_i, X_j) \triangleq \max_x \min_y\{d^0_{X_i X_j}(x, y)$ for all $x \in I(X_i)$ and $y \in I_i(X_j)\}$.
$D^-(X_i, X_j) \triangleq \min\{d^0_{X_i X_j}(x, y)$ for all $x \in I(X_i)$ and $y \in I_i(X_j)\}$. □

Clearly, we know that $D^+(X_i, X_j) \geq D^{+/-}(X_i, X_j) \geq D^-(X_i, X_j)$. If $P(X_1^n) \triangleq X_1 X_2 \cdots X_n$ is a high level path, then for all $x \in I(X_1)$ and $y \in I_{n-1}(X_n)$, $\Sigma_{i=1}^{n-1} D^-(X_i, X_{i+1})$

is a lower bound for $d^0_{P(X_1^n)}(x,y)$, and $\Sigma_{i=1}^{n-1} D^+(X_i, X_{i+1})$ and $\Sigma_{1=1}^{n-1} D^{+/-}(X_i, X_{i+1})$ give upper bounds $d^0_{P(X_1^n)}(x,y)$. In this sense, $\Sigma_{1=1}^{n-1} D^{+/-}(X_i, X_{i+1})$ is a conservative estimate for the cost of the low level optimal control from any $x \in I(X_1)$ to $y \in X_n$.

Let $D(P(X_1^n); D^{+/-})$ represent the $D^{+/-}$ cost of $P(X_1^n)$, i.e., $D(P(X_1^n); D^{+/-}) = \sum_{i=1}^{n-1} D^{+/-}(X_i, X_{i+1})$ and let $D(P(X_1^n); D^-)$ represent the D^- cost of $P(X_1^n)$. A high level path is said to be $D^{+/-}$-optimal if it has the minimal $D^{+/-}$ cost over all high level paths from X_1 to X_n. Suppose we set the high level cost function in terms of $D^{+/-}$, i.e., let $C_h(X_i, U_i^j) = D^{+/-}(X_i, X_j)$. If we perform dynamic programming with respect to this high level cost function, the $D^{+/-}$-optimal path $P^0_-(X_1, X_n) = X_1 X_2 \cdots X_m$ contains a possibly sub-optimal low level solution between the start state s in $I(X_1)$ and the target state t in X_m. This is the case simply because any high level path contains a (possibly sub-optimal) low level path by the definition of the DC property.

Let the high level cost function $C_h : (\pi - X^s \cup X^t) \times \Sigma_h \mapsto R^+$ be given. For $s \in X^s \in \pi$ and $t \in X^t \in \pi$, let $I(X^s) = \{s\}$ and $I(X^t) = \{t\}$. By virtue of the special status of the start and terminal states s and t, the value of $C_h(X^s, U_s^j)$ and $C_h(X_i, U_i^t)$ is taken to depend upon s and t respectively. We observe that the definition of C_h entails that it is defined only for high level blocks and transitions corresponding to DC pairs.

After this preparatory step, we have the two distinct schemes presented below for seeking a possibly sub-optimal low level path between the given start state s and target state t. For each of these, we that the target state t is accessible from the start state s, but we note that the failure of these algorithms to converse provides a rapid test for precise accessibility of t from s.

The P-HADP(C_h) Algorithm

This algorithm seeks the low level optimal path with respect to the constraint of confinement in a high level optimal path (with respect to C_h) from X^s to X^t: this is called the *path-wise HADP* or *P-HADP* for short.

P-HADP(C_h) Algorithm:

(1) Set $Dis(X^s; X^s) = 0$ and $Dis(X_i; X^s) = \infty$ for all $X_i \in \pi$ such that $X_i \neq X^s$;

(2) $Dis(X_j; X^s) = \min\{Dis(X_i; X^s) + C_h(X_i, U_i^j)\}$ for all $X_i \in \pi$ such that $\delta_h(X_i, U_i^j) = X_i$ with some $U_i^j \in \Sigma_h$. Find $Dis(X_i; X^s)$ for all $X_i \in \pi$.

(3) Find an optimal path from X^s to X^t in M_h with respect to C_h. Denote this path by $P^0(X^s, X^t; C_h)$.

(4) Denote the set of all states in the blocks on $P^0(X^s, X^t; C_h)$ by X_{opt}. Set $dis(s; s) = 0$ and $dis(x; s) = \infty$ for all $x \in X_{opt}$ such that $x \neq s$.

(5) $dis(y; s) = \min\{dis(x; s) + l(x, u)\}$ for all $x \in X_{opt}$ such that $\delta(x, u) = y \in X_{opt}$ for some $u \in \Sigma$. Find $dis(x; s)$ for all $x \in X_{opt}$.

(6) With $dis(x; s)$, find a low level optimal path from s to t in X_{opt}. □

P-HADP shall stand for P-HADP($D^{-/-}$), i.e. the HADP procedure with $D^{-/-}$ taken as the high level cost function C_h. We note that if $\{M, M_h\}$ is OC with the high level cost function $D^{-/-}$, P-HADP($D^{-/-}$) will generate the true low level optimal solutions.

The HADP(C_h) Algorithm

This algorithm seeks a low level sub-optimal trajectory contained block-wise in a high level optimal path $P^0(X^s, X^t; C_h)$. The resulting algorithm is called *block-wise HADP*, or simply *HADP* for short.

HADP(C_h) Algorithm:

(1) Set $Dis(X^s; X^s) = 0$ and $Dis(X_i; X^s) = \infty$ for all $X_i \in \pi$ such that $X_i \neq X^s$;

(2) $Dis(X_j; X^s) = \min\{Dis(X_i; X^s) + C_h(X_i, U_i^j)\}$ for all $X_i \in \pi$ such that $\delta_h(X_i, U_i^j) = X_i$ with some $U_i^j \in \Sigma_h$. Find $Dis(X_i; X^s)$ for all $X_i \in \pi$.

(3) Find an optimal path from X^s to X^t in M_h with respect to C_h. Denote this path by $P^0(X^s, X^t; C_h)$.

(4) Set $P^0(X^s, X^t; C_h) = Y_1 Y_2 ... Y_n$ with $Y_1 = X^s$ and $Y_n = X^t$. Set $x_1 = s$. Start at $i = 1$.

(5) Set $dis(x_i; x_i) = 0$ and $dis(x; x_i) = \infty$ for all $x \in Y_i \cup I_i(Y_{i+1})$.

(6) $dis(y; x_i) = \min\{dis(x; x_i) + l(x, u)\}$ for all $x \in Y_i$ such that $\delta(x, u) = y \in Y_i \cup I_i(Y_{i+1})$ with some $u \in \Sigma$. Find all $dis(x; x_i)$ for $x \in Y_i \cup I_i(Y_{i+1})$.

(7) Set $x_{i+1} = \arg\min_{y \in I_i(Y_{i+1})} dis(y; x_i)$. Find a low level optimal path from x_i to

x_{i+1} path-wise contained in Y_iY_{i+1}. Denote this path by $x_i \rightsquigarrow x_{i+1}$.
(8) If $x_{i+1} = t$, $s \rightsquigarrow x_2 ... \rightsquigarrow x_{n-1} \rightsquigarrow t$ is the solution, stop; else set $i = i + 1$, repeat 5 to 8. □

Similarly, HADP shall denote HADP($D^{+/-}$), i.e. the scheme above with C_h taken to be $D^{+/-}$. When $D^{+/-}(Y_i, Y_{i+1}) = D^-(Y_i, Y_{i+1})$, $1 \le i < n$, HADP will generate the true low level optimal solutions. P-HADP(D^-) and HADP(D^-) are the similar processes with D^- as C_h in the above procedures.

The *diameter* of a block X_i with respect to its in-set $I(X_i)$ is defined as $\mathcal{D}(X_i) = max\{d^0_{X_i}(x, y) : x \in I(X_i)\ y \in X_i\}$.

Definition 3.12. (Condition MM$<_s$) For a two level hierarchy $\{M, M_h\}$, the condition MM$<_s$ holds if the following is true: for any two blocks $X_1, X_n \in \pi$, with $s \in X_1$, let $Y_1 = X_1$ and $Y_m = X_n$, then, whenever $P^0(X_1^n) = X_1X_2...X_n$ is a $D^{+/-}$-optimal path, and $P'(Y_1^m) = Y_1Y_2...Y_m$ is any path which is not $D^{+/-}$-optimal, it is the case that $D(P^0(X_1^n); D^{+/-}) + \mathcal{D}(X_n) < D(P(Y_1^m); D^-)$. □

Theorem 3.7. *Let $\{M, M_h\}$ be a two level IBC hierarchy. For any $X_1, X_n \in \pi$ with $s \in X_1$, assume that the high level $D^{+/-}$-optimal path from X_1 to X_n is unique. Then, MM$<_s$ implies the hierarchy $\{M, M_h\}$ is OC($D^{+/-}$) with respect to s.*

Proof:

Suppose for a hierarchy $\{M, M_h\}$, the high level cost function is given by $C_h(X_i, U_i^j)$ $= D^{+/-}(X_i, X_j)$ for every DC pair $\langle X_i, X_j \rangle$ with $X_j = \delta_h(X_i, U_i^j)$. Under the assumption that the condition MM$<_s$ holds, we prove $\{M, M_h\}$ is OC($D^{+/-}$) with respect to $s \in X_1 \in \pi$.

(1) Let $y \in X_j \in \pi$ be an arbitrary state accessible from s, and let $u \in \mathbf{u}^0(s, y)$. Denote the high level path containing $Trj(s, u)$ by $P(Y_1^m) = Y_1Y_2...Y_m$, with $Y_1 = X_1$ and $Y_m = X_j$. There is an entry state $x \in I(X_j)$, such that $u_1 \in \mathbf{u}^0(s, x)$ and $u_2 \in \mathbf{u}^0(x, y)$, where $u_1u_2 = u$. It is evident that $D(P(Y_1^m); D^-) \le d^{00}(s, x)$.

Next, to obtain a contradiction, assume $P(Y_1^m)$ is not $D^{+/-}$-optimal, i.e. there is $P(Z_1^l) = Z_1Z_2...Z_l$ with $Z_1 = X_1$ and $Z_l = X_j$ which differs from $P(Y_1^m)$ and is

$D^{+/-}$-optimal. We shall prove by (forward) induction (on k) that there exists z_k in $I_{k-1}(Z_k)$, $1 < k < l$, such that $d^0_{P(Z_1^k)}(s, z_k) \leq D(P(Z_1^k); D^{+/-})$.

When $k = 2$, the above inequality holds by the definition of $D^{+/-}(Z_1, Z_2)$. Suppose it holds for $k = p$, $2 < p < l$. By the property of IBC blocks, every state in Z_{p+1} is accessible from z_p. Denote $d_{Z_pZ_{p+1}}(z_p, z^*_{p+1}) = arg\min_{z \in I_p(Z_{p+1})} d^0_{Z_pZ_{p+1}}(z_p, z)$. It is clear $d_{Z_pZ_{p+1}}(z_p, z^*_{p+1}) \leq D^{+/-}(Z_p, Z_{p+1})$. Hence,

$$d^0_{P(Z_1^{p+1})}(s, z_{p+1}) \leq d^0_{P(Z_1^p)}(s, z^*_p) + d_{Z_pZ_{p+1}}(z_p, z^*_{p+1})$$

$$\leq D(P(Z_1^p); D^{+/-}) + D^{+/-}(Z_p, Z_{p+1}) = D(P(Z_1^{p+1}); D^{+/-}).$$

Therefore, it is straightforward to see that

$$d^0_{P(Z_1^l)}(s, x) \leq d^0_{P(Z_1^l)}(s, z^*_l) + d^0_{Z_l}(z^*_l, x) \leq D(P(Z_1^l); D^{+/-}) + \mathcal{D}(X_j) < D(P(Y_1^m); D^-).$$

Then, by the assumption that MM$<_s$ holds for $\{M, M_h\}$, $D(P(Z_1^l); D^{+/-}) + \mathcal{D}(X_j) < D(P(Y_1^m); D^-)$, thus $d^0_{P(Z_1^l)}(s, x) < D(P(Y_1^m); D^-)$. This leads to the contradiction that $d^0_{P(Z_1^l)}(s, x) < d^{00}(s, x)$. Hence, $P(Y_1^m)$ is optimal.

(2) Let $P(Y_1^m) = Y_1Y_2...Y_m$ be the unique high level $D^{+/-}$-optimal path from $Y_1 = X_1$, $s \in Y_1$, to $Y_m = X_j$ and let y be an arbitrary state in X_j. Assume that $u \in \mathbf{u}^0(s, y)$ and $Trj(s, u)$ is contained in $P(Z_1^l) = Z_1Z_2...Z_l$, with $Z_1 = X_1$ and $Z_l = X_j$, which differs from $P(Y_1^m)$. We may decompose u into $u = u_1u_2$, where $u_1 \in \mathbf{u}^0(s, x), u_2 \in \mathbf{u}^0(x, y)$ with $x \in I_{Z_{l-1}}(Z_l)$ is the last entry state into Z_l on $Trj(s, y)$. Clearly,

$$D(P(Z_1^l); D^-) \leq d^{00}(s, x) = d^0_{P(Z_1^l)}(s, x).$$

But by the condition MM$<_s$,

$$D(P(Y_1^m); D^{+/-}) + \mathcal{D}(X_j) < D(P(Z_1^l); D^-)$$

And clearly we have,

$$d^0_{P(Y_1^m)}(s, x) \leq D(P(Y_1^m); D^{+/-}) + \mathcal{D}(X_j).$$

Thus, we reach the contradiction $d^0_{P(Y_1^m)}(s, x) < d^{00}(s, x)$. Therefore, $P(Y_1^m)$ contains a low level optimal path $Trj(s, u)$ from s to y. Since y is an arbitrary state in X_j, the second part of the OC property is established. $\square$

In spite of the information given in the theorem above, there still remains the problem that if there exist two or more high level optimal paths from X_1 to X_n, then in an optimisation procedure we have to investigate which of them actually includes the optimal low level solution.

Let $\alpha_j(X_i) = D^-(X_i, X_j)/D^{+/-}(X_i, X_j)$ for all $X_i, X_j \in \pi$ such that $\langle X_i, X_j\rangle$ is DC.

Theorem 3.8. *For arbitrary start state $s \in X_i$ and target state $t \in X_j$, let $d^{00}(s,t)$ be the cost of the global low level optimal path from s to t, $P^0(Y_1, Y_n; D^{+/-})$ be the $D^{+/-}$-optimal high level path from $Y_1 = X_i$ to $Y_n = X_j$, and let $d^{HADP(D^{+/-})}(s,t)$ denote the cost of the HADP(D^+) solution. Then if the $D^{+/-}$-optimal high level path contains a global optimal solution,*

$$d^{HADP(D^+)}(s,t) - d^{00}(s,t) \leq \sum_{i=1}^{n-1}(1 - \alpha_{i-1}(Y_i))D^{+/-}(Y_i, Y_{i-1}).$$

Proof:

$$d^{HADP(D^+)}(s,t) \leq \sum_{i=1}^{n-1} D^{+/-}(Y_i, Y_{i-1}).$$

If the $D^{+/-}$-optimal high level path contains a global optimal solution,

$$d^{00}(s,t) \geq \sum_{i=1}^{n-1} D^-(Y_i, Y_{i-1}).$$

Because, $D^-(Y_i, Y_{i-1}) \geq \alpha_{i-1}(Y_i)D^{+/-}(Y_i, Y_{i-1})$,

$$d^{HADP(D^+)}(s,t) - d^{00}(s,t)$$

$$\leq \sum_{i=1}^{n-1}(D^{+/-}(Y_i, Y_{i+1}) - D^-(Y_i, Y_{i+1}))$$

$$\leq \sum_{i=1}^{n-1}(1 - \alpha_{i+1}(X_i))D^{+/-}(Y_i, Y_{i-1}).$$

□

If the $D^{+/-}$-optimal high level path does not contain a global optimal solution, we define a parameter ξ to represent its *closeness* to the D^--optimal high level path. Let $P^0(Y_1, Y_n; D^{+/-})$ be a $D^{+/-}$-optimal high level path from $Y_1 = X_i$ to $Y_n = X_j$, and $P^0(Z_1, Z_m; D^{-/-})$ be a D^--optimal high level path from $Z_1 = X_i$ to $Z_m = X_j$; then we define $\xi = \sum_{i=1}^{m-1} D^-(Z_i, Z_{i+1}) / \sum_{i=1}^{n-1} D^-(Y_i, Y_{i+1})$.

Corollary 3.2. *For arbitrary start state $s \in X_i$ and target state $t \in X_j$, let $d^{00}(s,t)$ be the cost of the global low level optimal path from s to t, $P^0(Y_1, Y_n; D^{+/-})$ be the $D^{+/-}$-optimal high level path from $Y_1 = X_i$ to $Y_n = X_j$, and let $d^{HADP(D^{+/-})}(s,t)$ denote the cost of the HADP(D^+) solution. Then*

$$d^{HADP(D^{+})}(s,t) - d^{00}(s,t) \leq \sum_{i=1}^{n-1} (1 - \xi \alpha_{i+1}(Y_i)) D^{+/-}(Y_i, Y_{i+1}).$$

□

We define the parameter $\alpha = \alpha(\pi)$ to be the supremum of the set of μ $(1 \geq \mu > 0)$ satisfying $D^-(X_i, X_j) \geq \mu D^{+/-}(X_i, X_j)$, for all i, j, such that $\langle X_i, X_j \rangle$ is DC.

Let us choose another version of closeness measure of $D^{+/-}$-optimal high level path P^0 to a high level path P^{00} which contains a low level optimal path from the start state to the target state, namely, $\eta \triangleq |P^0 \cap P^{00}| / |P^0|$, where $|P^0 \cap P^{00}|$ is the number of blocks on both P^0 and P^{00}, and $|P^0|$ is the total number of blocks on P^0. Denote $\alpha^0 = \min\{\alpha_j(X_i) : X_i X_j \text{ is a segment of } P^0\}$.

Corollary 3.3. *For arbitrary start state $s \in X_i$ and target state $t \in X_j$, let $d^{00}(s,t)$ be the cost of the global low level optimal path from s to t, $P^0(Y_1, Y_n; D^{+/-})$ be the $D^{+/-}$-optimal high level path from $Y_1 = X_i$ to $Y_n = X_j$, and let $d^{HADP(D^{+/-})}(s,t)$ denote the cost of the HADP(D^+) solution. Then*

$$d^{HADP(D^{+})}(s,t) - d^{00}(s,t) \leq (1 - \eta\alpha^0 - (1-\eta)\alpha) P^0(Y_1, Y_n; D^{+/-}).$$

□

It is natural to look for properties of a state space partition π which permit the associated HADP($D^{+/-}$) and P-HADP($D^{+/-}$) to generate near-optimal results. In particular, we seek partitions for which the D^- and $D^{+/-}$ parameters are as close as possible: this is in order to reduce the variability of costs of block traversals corresponding to different in-set states.

Theorem 3.9. *For arbitrary start state $s \in X_i$ and target state $t \in X_j$, let $d^{00}(s,t)$ be the cost of the global low level optimal path from s to t, $P^0(X_i, X_j; D^{+/-})$ be the $D^{+/-}$-optimal high level path from X_i to X_j, and let $d^{HADP(D^{+/-})}(s,t)$ denote the cost of the HADP($D^{+/-}$) solution. Then*

$$d^{HADP(D^{+/-})}(s,t) - d^{00}(s,t) \le \frac{1-\alpha}{\alpha} d^{00}(s,t),$$

or, equivalently,

$$d^{HADP(D^{+/-})}(s,t) \le \alpha^{-1} d^{00}(s,t).$$

Proof:
It is clear that

$$D(P^0(X_i, X_j; D^-); D^-) \le d^{00}(s,t)$$

and

$$d^{HADP(D^{+/-})}(s,t) \le D(P^0(X_i, X_j; D^{+/-}); D^{+/-}).$$

Because $P^0(X_i, X_j : D^{+/-})$ is $D^{+/-}$-optimal,

$$D(P^0(X_i, X_j; D^{+/-}); D^{+/-}) \le D(P^0(X_i, X_j; D^-); D^{+/-}).$$

By the definition of α,

$$D(P^0(X_i, X_j; D^-); D^{+/-}) \le \frac{1}{\alpha} D(P^0(X_i, X_j; D^-); D^-).$$

Thus,

$$d^{HADP(D^{+/-})}(s,t) - d^{00}(s,t)$$

$$\le D(P^0(X_i, X_j; D^{+/-}); D^{+/-}) - d^{00}(s,t)$$

$$\le D(P^0(X_i, X_j; D^-); D^{+/-}) - d^{00}(s,t)$$

$$\le \frac{1}{\alpha} D(P^0(X_i, X_j; D^-); D^-) - d^{00}(s,t)$$

$$\le \frac{1}{\alpha} d^{00}(s,t) - d^{00}(s,t) = \frac{1-\alpha}{\alpha} d^{00}(s,t)$$

□

We observe that in case $D(P(X_1^n); D^-)$ is not a tight estimate of $d_P^0(x, y)$ for $x \in I(X_1)$ and $y \in I_{n-1}(X_n)$, and if other knowledge of the least cost path or its value between X_1 to X_n is available, better estimation may be achieved by exploiting this information. For example, if we know $d_{P(X_1^n)}^0(x, y)$ for some $x \in X_1$ and $y \in X_n$, $d_{P(X_1^n)}^0(x', y') \geq d_{P(X_1^n)}^0(x, y) - d_{X_1}(x, x') - d_{X_2}(y', y)$ for any $x' \in X_1$ and $y' \in X_2$. All such information about lower bounds for $d_{P(X_1^n)}^0(x', y')$ can be used to evaluate an HADP solution.

3.5. Semi-dual System Graphs (M_h^{sd}, C_h^{sd})

In this section and the next, we discuss our approaches to the improvements of the performance of HADP. This is a two-fold problem: increasing the accuracy of sub-optimality estimation and decreasing the number of blocks of partition machines and thus reducing the computational time.

In Section 4, we have defined $D^-(X_i, X_j)$, $D^{-\,-}(X_i, X_j)$ and $D^-(X_i, X_j)$ as the bounds of the costs of low level control driving the system from states in $I(X_i)$ to states in $I_i(X_j)$, where $\langle X_i, X_j \rangle$ is a pair of DC blocks. As we pointed out in Section 4, the entry states of X_i, $I(X_i)$, may be differentiated into a collection of subsets according to the neighbouring blocks. If we identify the source and target block of a low level path to traverse a block X_i (i.e., use the costs from $I_k(X_i)$ to $I_i(X_j)$, where $\langle X_k, X_i \rangle$ and $\langle X_k, X_i \rangle$ are DC) to define those bounds, we are able to obtain more precise estimates of the low level optimal cost.

Definition 3.13. $(D_k^-(X_i, X_j), D_k^{-\,-}(X_i, X_j), D_k^-(X_i, X_j))$
If $\langle X_k, X_i \rangle$ and $\langle X_i, X_j \rangle$ are DC,
$D_k^-(X_i, X_j) \triangleq \max\{d_{X_i X_j}^0(x, y)$ for all $x \in I_k(X_i)$ and $y \in I_i(X_j)\}$,
$D_k^{-\,-}(X_i, X_j) \triangleq \max_x \min_y\{d_{X_i X_j}^0(x, y)$ for all $x \in I_k(X_i)$ and $y \in I_i(X_j)\}$,
$D_k^-(X_i, X_j) \triangleq \min\{d_{X_i X_j}^0(x, y)$ for all $x \in I_k(X_i)$ and $y \in I_i(X_j)\}$. □

Suppose there is more than one X_k block such that $\langle X_k, X_i \rangle$ is DC, then one needs to know which block is the predecessor of X_i on a high level path in order to choose the corresponding bounds to estimate the cost. As a result, the transition $X_i \rightarrow X_j$ cannot be labelled by a single value to carry out DP.

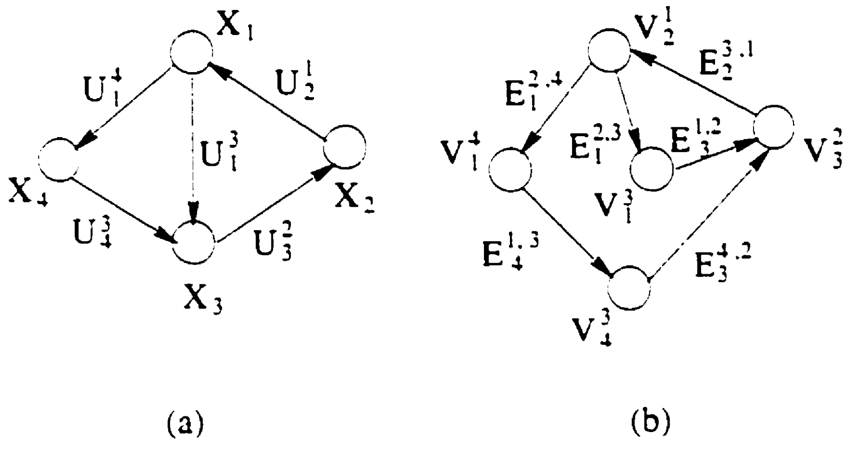

FIGURE 3.5. (a) M_h (b) M_h^{sd}

Using the notion of out-sets in Chapter 2, we may introduce a model based on M_h called the *semi-dual high level machine* M_h^{sd}. Whenever $\langle X_i, X_j \rangle$ is a DC pair in M_h, let us define the high level transition event symbol V_i^j, and denote the collection of all such symbols by V. By the standing IBC hypothesis, there exist paths from each element in $I_i(X_j)$ to each element in $O(X_j)$ and hence, in particular, to each element in the subset $O_k(X_j) \subset O(X_j)$ consisting of elements with one step transitions to X_k. Define the set of edges E to be the new collection of derived high level events $E_j^{i,k}$, each of which corresponds to a set of paths in X_j from $I_i(X_j)$ to $O_k(X_j)$ and then to $I_j(X_k)$. The elements $E_j^{i,k} \in E$ are in one-to-one correspondence with the pairs $(V_i^j, V_j^k) \in V \times V$ and the resulting finite state machine $\{V, E, \delta_h^{sd}\}$ is denoted by M_h^{sd}.

The algorithm below formulates the HADP procedure based on M_h^{sd}. Here the idea is, with M_h^{sd}, it is feasible to perform DP. After we find an optimal solution with respect to M_h^{sd}, we obtain a sequence of between block boundaries, and this in turn gives a sequence of high level blocks where a sub-optimal low level solution is contained. Taking between block boundary $\partial_{X_i}(X_j)$ as a node representing a DC pair $\langle X_i, X_j \rangle$, and taking the minimal cost between $\partial_{X_i}(X_j)$ and $\partial_{X_j}(X_k)$ as the high level event $E_j^{i,k}$, we carry out DP in the following fashion.

Pre-processing

(1) Convert M_h into M_h^{sd};

(2) Define the high level cost function: for all $1 \le i, j, k \le |V|$, if $\delta_h^{sd}(V_i^j, E_j^{i,k}) = V_j^k$,

$C_h^{sd}(V_i^{\cdot j}, E_j^{i,k}) = D_j^{+/-}(X_i, X_k)$;
(3) If the start state $s \in X_i \in \pi$ and the target state $t \in X_j \in \pi$, add two new elements, the source node $V_s^{\cdot i}$ and and the target node $V_j^{\cdot t}$ to $V^{\cdot}$: $V^{\cdot} = V^{\cdot} \cup \{V_s^{\cdot i}, V_j^{\cdot t}\}$;
(4) Add new transitions from $V_s^{\cdot i}$ and to $V_j^{\cdot t}$ to E in the following way:
for all $1 \leq k \leq |\pi|$, if $\langle X_i, X_k\rangle$ is DC, add $E_i^{s,k}$ to E, and $\delta_h^{sd}(V_s^{\cdot i}, E_i^{s,k}) = V_i^{\cdot k}$, the cost of this transition is $C_h^{sd}(V_s^{\cdot i}, E_i^{s,k}) = \min\{d^0_{X_i X_k}(s, y) \text{ for all } y \in I_i(X_k)\}$;
if $\langle X_l, X_j\rangle$ is DC, add a new control symbol $E_j^{l,t}$ to E, and $\delta_h^{sd}(V_l^{\cdot j}, E_j^{l,t}) = V_j^{\cdot t}$, the cost of this transition is $C_h^{sd}(V_l^{\cdot j}, E_j^{l,t}) = \max\{d^0_{X_j}(x, t) \text{ for all } x \in I_l(X_j)\}$;

Denote the resulting model with $V_s^{\cdot i}$ and $V_j^{\cdot t}$ added by M_h^{sd}. □

In other words, in the pre-processing we set the start and target nodes in accordance with the start state and the target state, and then set their corresponding transitions to the boundary of the DC pair $\langle X_i, X_k\rangle$ and from the boundary of the DC pair $\langle X_k, X_j\rangle$. With these new nodes, we are able to carry out DP with M_h^{sd}, in order to find a sequence of blocks containing a sub-optimal low level path between the start and target states.

The Semi-dual HADP($D^{+/-}$) algorithm
(1) Set $Dis(V_s^{\cdot i}; V_s^{\cdot i}) = 0$; Set $Dis(V_j^{\cdot k}; V_s^{\cdot i}) = \infty$ for all $V_j^{\cdot k} \neq V_s^{\cdot i}$;
(2) $Dis(V_j^{\cdot k}; V_s^{\cdot i}) = \min\{C_h^{sd}(V_l^{\cdot j}, E_j^{l,k}) + Dis(V_l^{\cdot j}; V_s^{\cdot i})\}$ for all $E_j^{l,k} \in E$; find all $Dis(V_j^{\cdot k}; V_s^{\cdot i})$ for $V_j^{\cdot k} \in V^{\cdot}$;
(3) If $Dis(V_j^{\cdot t}; V_s^{\cdot i}) = \infty$ (s is not reachable to t), stop; else, find a shortest path from $V_s^{\cdot i}$ to $V_j^{\cdot t}$ with respect to C_h^{sd} in M_h^{sd}. Denote this optimal path by $P^0(V^{\cdot s}, V^{\cdot t}; D^{+/-})$. Let $P^0(V^{\cdot s}, V^{\cdot t}; D^{+/-}) = V_{i_0}^{\cdot i_1} V_{i_1}^{\cdot i_2} \ldots V_{i_{n-1}}^{\cdot i_n}$ with $V_{i_0}^{\cdot i_1} = V_s^{\cdot i}$ and $V_{i_{n-1}}^{\cdot i_n} = V_j^{\cdot t}$;
(4) Let $x_1 = s$. Start from $k = 1$;
(5) If $X_{i_k} = X_j$, $I_{next} = \{t\}$; else $I_{next} = I_{i_k}(X_{i_{k+1}})$;
(6) Set $dis(x_i; x_i) = 0$; set $dis(x; x_i) = \infty$ for all $x \in X_{i_k} \cup I_{i_k}(X_{i_{k+1}})$ and $x \neq x_i$;
(7) $dis(x; x_i) = \min\{dis(y; x_i) + l(y, u)\}$ for all $y \in X_{i_k}$ such that there is $u \in \Sigma$ s.t. $\delta(y, u) = x$; find all $dis(x; x_i)$ for $x \in X_{i_k} \cup I_{i_k}(X_{i_{k+1}})$;
(8) Let $x_{k+1} = \arg\min_{x \in I_{i_k}(X_{i_{k+1}})} d^0_{X_{i_k} X_{i_{k+1}}}(x_k, x)$. Find a shortest path from x_1 to x_{k+1} path-wise contained in $X_{i_k} X_{i_{k+1}}$. Denote this path by $x_k \rightsquigarrow x_{k+1}$;
(9) If $x_{k+1} = t$, i.e. t is reached, $x_1 \rightsquigarrow x_2 \rightsquigarrow \ldots \rightsquigarrow x_{k+1}$ forms a low level solution,

stop; else, set $k = k + 1$, repeat step 5-9. □

If we find the low level optimal path from s to t with respect to the high level optimal path obtained in step 4, the result gives a P-HADP algorithm solution.

Theorem 3.10. *For a high level path $P(X_1^n) = X_1X_2\cdots X_n$, with arbitrary start state $s \in X_1$ and target state $t \in X_n$,*

$$d_P^0(s,t) \le \frac{D(P(X_1^n); D^+) + D(P(X_1^n); D^-)}{2}.$$

Proof:

Let $D^-(X_i, X_{i+1})$, $1 \le i \le n-1$ (the minimal between in-set cost) be realised by a sequence of (not necessarily connected) segments $S_1 \triangleq s \rightsquigarrow y_2$, $S_2 \triangleq z_2 \rightsquigarrow z_3$, $S_3 \triangleq y_3 \rightsquigarrow y_4$, $S_4 \triangleq z_4 \rightsquigarrow z_5, \ldots$, for $y_2, z_2 \in I(X_2)$; $y_3, z_3 \in I(X_3)$; ...; $y_{n-1}, z_{n-1} \in I(X_{n-1})$. Linking y_2, y_3; y_4, y_5;... and s, z_2;z_3, z_4;..., respectively by optimal paths, we construct two low level paths from s to t as:
$p_1 = s \rightsquigarrow y_2 \rightsquigarrow y_3 \cdots y_{n-1} \rightsquigarrow t$ and
$p_2 = s \rightsquigarrow z_2 \rightsquigarrow z_3 \cdots z_{n-1} \rightsquigarrow t$.

Hence we obtain two low levels paths with alternate realisations of $D^-(X_i, X_{i+1})$. Denote the low level costs of p_1 and p_2 by $d(p_1)$ and $d(p_2)$, then collecting the elements of S_i together to give the first sum below, we obtain,

$$d(p_1) + d(p_2) = (\sum_{i=1}^{n-1} D^-(X_i, X_{i+1})) + d^0_{X_1X_2}(s, z_2) + d^0_{X_2X_3}(y_2, y_3) + \ldots$$

$$\le \sum_{i=1}^{n-1} D^-(X_i, X_{i+1})) + \sum_{i=1}^{n-1} D^+(X_i, X_{i+1})$$

$$= D(P(X_1^n); D^+) + D(P(X_1^n); D^-).$$

Since $d_P^0(s,t) \le d(p_1)$ and $d_P^0(s,t) \le d(p_2)$, we have

$$d_P^0(s,t) \le \frac{D(P(X_1^n): D^+) + D(P(X_1^n): D^-)}{2}.$$

□

Let us define the parameter $\beta = \beta(\pi)$ to be the supremum of all μ $(1 \ge \mu > 0)$ satisfying $D_k^-(X_i, X_j) \ge \mu D_k^+(X_i, X_j)$ for all k, i, j such that $\langle X_k, X_i \rangle$ and $\langle X_i, X_j \rangle$ are DC.

Theorem 3.11. *For arbitrary start state $s \in X_i$ and target state $t \in X_j$, let $P^0(X_i, X_j : D^-)$ be the D^--optimal high level path from X_i to X_j, and $d^0_{P^0(X_i,X_j,D^-)}(s,t)$ be the cost of the P-HADP(D^-) solution, then*

$$d^0_{P^0(X_i,X_j,D^-)}(s,t) - d^{00}(s,t) \le \frac{1-\beta}{2\beta} d^{00}(s,t).$$

Proof:

By Theorem 3.10,

$$d^0_{P^0(X_i,X_j,D^-)}(s,t) - d^{00}(s,t)$$

$$\le \frac{1}{2}(D(P^0(X_i,X_j: D^-): D^+) + D(P^0(X_i,X_j: D^-): D^-)) - d^{00}(s,t)$$

Since $D_k^-(X_i, X_j) \ge \beta D_k^+(X_i, X_j)$ for all k, i, j,

$$D(P^0(X_i,X_j: D^-): D^-) \ge \beta D(P^0(X_i,X_j: D^-): D^+).$$

Hence,

$$d^0_{P^0(X_i,X_j:D^-)}(s,t) - d^{00}(s,t)$$

$$\le \frac{1+\beta}{2\beta} D(P^0(X_i,X_j: D^-): D^-) - d^{00}(s,t)$$

$$\le \frac{1+\beta}{2\beta} d^{00}(s,t) - d^{00}(s,t)$$

$$= \frac{1-\beta}{2\beta} d^{00}(s,t)$$

□

Definition 3.14. ($\overline{D}_k(X_i, X_j)$) *The* (arithmetic) average between boundary distance *is defined as*

$$\overline{D}_k(X_i, X_j) \triangleq \frac{\sum_{x \in I_k(X_i)} (\sum_{y \in I_i(X_j)} d^0_{X_i X_j}(x, y))}{|I_k(X_i)| \times |I_i(X_j)|}$$

□

Theorem 3.12. *Let $P(X_1^n) = X_1 X_2 ... X_n$ be a high level path with $s \in X_1$ and $t \in X_n$. then $d^0_P(s, t) \leq \sum_{i=0}^{n-1} \overline{D}_{i-1}(X_i, X_{i+1})$.*

Proof:

Let the elements of $I_{i-1}(X_i)$? be denoted $\{x^i_1, x^i_2, ..., x^i_{|I_{i-1}(X_i)|}\}$. $1 \leq i \leq n$. where $I_0(X_1) = \{s\}$ and $I_{n-1}(X_n) = \{t\}$. We connect x^i_j and x^{i+1}_k. $1 \leq j \leq |I_{i-1}(X_i)|$. $1 \leq k \leq |I_i(X_{i+1})|$ via the optimal low level path $x^i_j \overset{0}{\rightsquigarrow} x^{i+1}_k$ contained in $X_i X_{i+1}$ with the cost of $d^0_{X_i X_{i+1}}(x^i_j, x^{i+1}_k)$. This gives a low level path from s to t path-wise contained in $P(X_1^n)$ necessarily passing through $I_{i-1}(X_i)$. $1 < i < n$. Because there are $|I_i(X_{i+1})|$ nodes in each $I_{i-1}(X_i)$. connecting any one state in $I_{i-1}(X_i)$ to any state in $I_i(X_{i+1}$ gives a low level path from s to t. Hence the total number of distinct paths of this type is $\prod_{i=1}^n |I_{i-1}(X_i)|$. The sum of the costs of all these low level paths shall be denoted by S_{st}. Since $x^i_j \overset{0}{\rightsquigarrow} x^{i+1}_k$ is on $\prod_{k=1, k\neq i, k\neq i+1}^n |I_{k-1}(X_k)| = (\prod_{l=1}^n |I_{l-1}(X_l)|)/(|I_{i-1}(X_i)| |I_i(X_{i+1})|)$ distinct low level paths from s to t of the form described above. we have.

$$S_{st} = \sum_{y_1 \in I_0(X_1)} (\sum_{y_2 \in I_1(X_2)} d^0_{X_1 X_2}(y_1, y_2) + (\sum_{y_3 \in I_2(X_3)} d^0_{X_2 X_3}(y_2, y_3) + ...$$

$$+(\sum_{y_n \in I_{n-1}(X_n)} d^0_{X_{n-1} X_n}(y_{n-1}, y_n))...))$$

$$= \sum_{i=1}^{n-1} \sum_{j=1}^{|I_{i-1}(X_i)|} \sum_{k=1}^{|I_i(X_{i+1})|} (d^0_{X_i X_{i+1}}(x^i_j, x^{i+1}_k) \frac{\prod_{l=1}^n |I_{l-1}(X_l)|}{|I_{i-1}(X_i)||I_i(X_{i+1})|})$$

$$= (\prod_{l=1}^n |I_{l-1}(X_l)|) \sum_{i=1}^{n-1} (\frac{\sum_{j=1}^{|I_{i-1}(X_i)|} \sum_{k=1}^{|I_i(X_{i+1})|} d^0_{X_i X_{i+1}}(x^i_j, x^{i+1}_k)}{|I_{i-1}(X_i)||I_i(X_{i+1})|})$$

$$= (\prod_{k=1}^{n} |I_{k-1}(X_k)|)(\sum_{i=1}^{n-1} \overline{D}_{i-1}(X_i, X_{i+1})).$$

Since $d_P^0(s,t)$ is less than or equal to the cost of any of the above paths,

$$(\prod_{k=1}^{n} |I_{k-1}(X_k)|) d_P^0(s,t) \leq S_{st}.$$

Therefore.

$$d_P^0(s,t) \leq \sum_{i=0}^{n-1} \overline{D}_{i-1}(X_i, X_{i-1}).$$

□

We observe that $\overline{D}\,(\cdot,\cdot)$ is a valid candidate for a high level cost function C_h in the context of semi-dual HADP algorithms.

3.6. wST-IBC Partitions

Evidently. the number of states in M_h^{sd} is usually strictly greater than number of states in M_h . therefore. performing HADP with M_h^{sd} is more complex than with M_h. To counteract this affect we shall weaken the IBC condition with the objective of decreasing $|\pi|$ and thus decreasing $|U_h|$. For systems for which a direction of flow from start states to target states is defined. we formulated in [**15**] a generalisation of IBC partition called ST-IBC partition and within this framework we derived a hierarchical control structure based on the lattice of ST-IBC partition machines.

In [**15**]. the notion of the ST-IBC property of a block $X_i \in \pi$ was defined as follows: if (i) there is a path internal to X_i from every state in $I(X_i)$ to every state in $O(X_i)$: (ii) all states in $O(X_i)$ are mutually accessible by paths internal to X_i, then X_i is said to be ST-IBC.

Now we may define a slightly different ST-IBC property of blocks by dropping the mutual accessibility of states in $O(X_i)$.

Definition 3.15. (wST-IBC) A partition block $X_i \in \pi$ is *weak ST-IBC* (wST-IBC) if for any state $x \in I(X_i)$ and any state $y \in O(X_i)$ there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$ and, for all $u' < u$, $\delta(x, u') \in X_i$. □

Denote the partition machine based on an ST-IBC partition π as $M_{ST}^{\pi} = \{\pi, \Sigma_h, \delta_h\}$. The dynamics of the ST-DC (see Chapter 2) partition machine M_{ST}^{π} is defined by the ST-DC relations over $\pi \times \pi$, i.e., if (X_i, X_j) is ST-DC, then there exists $U \in \Sigma_h$ such that $\delta_h(X_i, U) = X_j$.

Definition 3.16. (ST-Control Consistency) A two level hierarchy $\{M, M_{ST}^{\pi}\}$ is said to be *ST-control consistent (ST-CC)* if and only if the following two accessibility conditions hold:
(1) For all $x \in I(X_i) \in \pi$ and $y \in I(X_j) \in \pi$ if there exists $u \in \Sigma^*$ such that $\delta(x, u) = y$ then there exists $U \in \Sigma_h^*$ such that $\delta_h(X_i, U) = X_j$ and $Trj(x, u)$ is contained in $Trj(X_i, U)$.
(2) For all $X_i, X_j \in \pi$, if there exists $U \in \Sigma_h^*$ such that $\delta_h(X_i, U) = X_j$, then for all $x \in I(X_i)$, there exists $y \in I(X_j)$ and $u \in \Sigma^*$ such that $\delta(x, u) = y$ and $Trj(x, u)$ is contained in $Trj(X_i, U)$. □

Theorem 3.13. *If partition π is wST-IBC, the hierarchy $\{M, M_{ST}^{\pi}\}$ is ST-CC.*

□

The proof is similar to that to Theorem 3.3.

An algorithm to find a wST-IBC block containing a given seed as an in-set state in a constraint can be found in [**53**]. Partitioning X into a wST-IBC partition π is a process of high complexity, hence we use an algorithm to improve an existing IBC partition by generating a related wST-IBC partition. It functions by growing a given IBC block into a wST-IBC block with respect to a given constraint (see [**53**]).

3.7. Applications to the Broken Manhattan Grid Problem

We use a class of examples called the Broken Manhattan Grid (BMG) problems to illustrate the generation of hierarchical control structures and the operation of HADP algorithms.

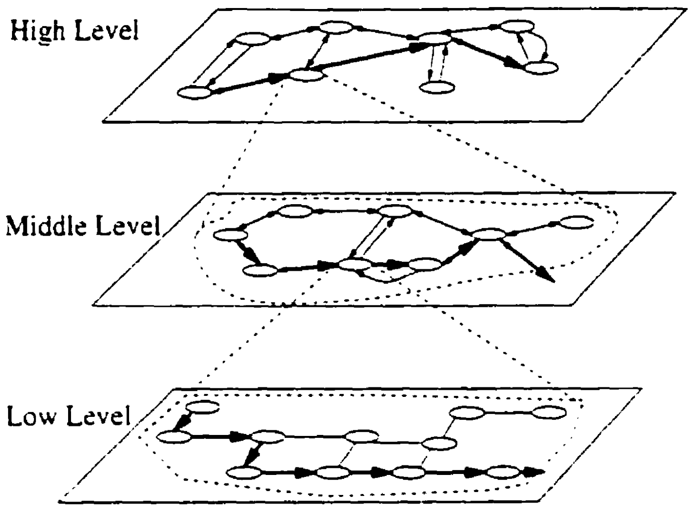

FIGURE 3.6. Three-level hierarchy for a Broken Manhattan Grid problem

Consider a graph with a large number of nodes and edges, which are formed by randomly removing some nodes and edges from a regular grid. A three-level hierarchy is constructed for such a system so that the middle level model is an IBC partition machine of the low level model (i.e. it forms an FSM abstracting the broken grid itself): and the high level model is an IBC partition machine of the middle level model.

Example 3.2. Consider a 500×500 regular grid (A grid is regular in the sense that every node has connections to its neighbouring nodes). Here we assume each link has unit cost. We randomly remove 10% of the nodes and their connecting edges. A middle level block of the hierarchy is shown in Figure 3.9. The simulation is done with C on a Silicon Graphics O_2 work station (CPU 180MHz, main memory 96M). In general, remarkable accelerations for the sub-optimal path calculations using HADP($D^{+/-}$) have been obtained. For example, when the start and target nodes are (1,1) and (499,499) respectively, the atomic full size DP takes over 30K seconds to find an optimal path with a cost of 1060: the HADP($D^{+/-}$) described above gives a sub-optimal cost of 1165 (less than 10% higher) in less than 9 seconds. □

Example 3.3. In this example, a 100×100 unidirectional regular grid (this can be viewed as a directed graph and moreover, and, in fact, as a finite state machine) is given to represent a transportation network. We assign the direction of links as follows: odd-numbered vertical streets are north bound, even-numbered ones are

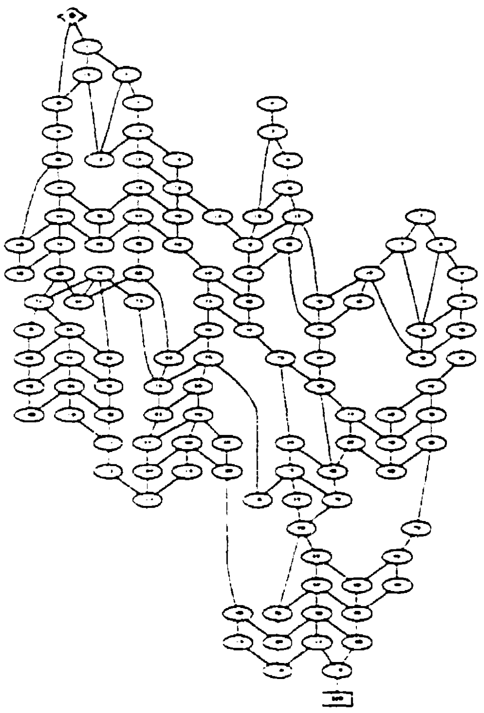

FIGURE 3.7. A Broken Manhattan Grid

south bound: odd-numbered horizontal streets are east bound, even-numbered ones are west bound. Here we assume each link has unit cost. Subsequently we randomly remove 10% of the nodes and corresponding edges concerned with them. This grid has some resemblance to the Manhattan area. When the start and target nodes are (10,99) and (89,88) respectively, (see Figure 3.8 for a high level block structure) the atomic full size DP takes over 55 seconds to find an optimal path with the cost of 112. We use the semi dual method to form a hierarchy, the D^- cost of the D^- optimal high level path is 88 and the $D^{+/-}$ cost of the $D^{+/-}$ optimal high level path is 116, so the error of HADP solution is less than (116-88)/88<32% of the global optimum. In fact, the HADP($D^{+/-}$) gives a sub-optimal cost 116 (<4% higher) in 3 seconds. □

In a graph generated from a 88 × 88 regular unidirectional grid with 10% nodes randomly removed, we choose the start states as 1, 1111, 2222, 3333, 4466, 5666 and 6666, and we also set the target to be in the same set as above. Then, we carry out HADP with semi dual graph for the start target pair chosen from the above set respectively. The ratio of HADP cost and global optimum as well as the speed up of HADP are visualised in Figure 3.10 and 3.11. In Figure 3.12, the distribution of

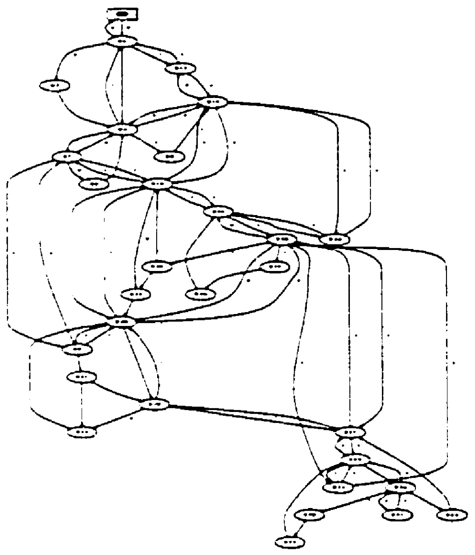

FIGURE 3.8. A block at the top level of Example 3.2

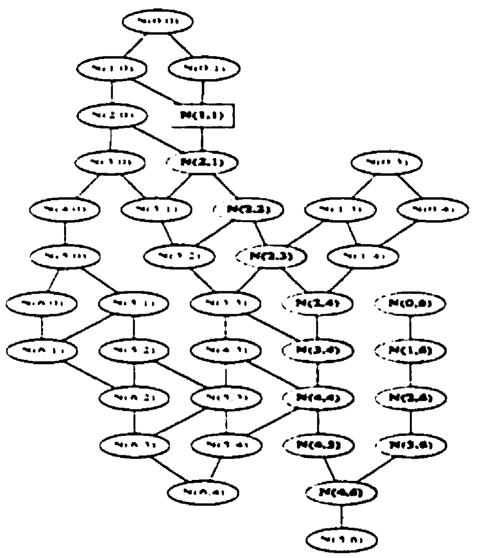

FIGURE 3.9. A block at the middle level of Example 3.2

$\alpha(X_i..X_j)$ is given when the start state is 1111 and the target state is 6666.

A rough comparison of the time complexity of HADP (with a two level hierarchy) and that of standard DP for finite state machines in general reveals the drastic speed-up of computation obtained by HADP. Specifically, suppose we take Dijkstra's

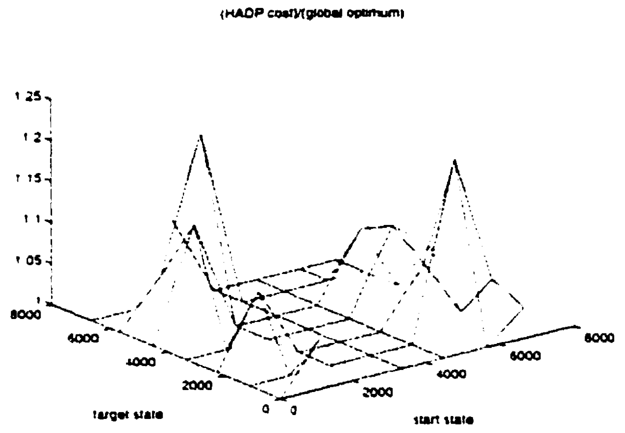

FIGURE 3.10. The ratio of the costs of HADP solution and global optimum

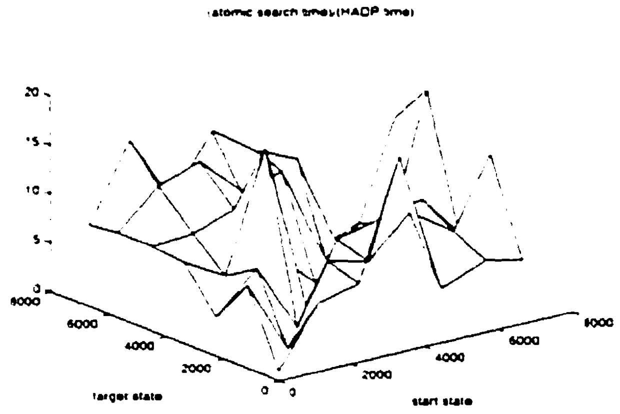

FIGURE 3.11. The ratio of the times of HADP solution atomic search

shortest path algorithm which is of $O(n^2)$ for a graph with n nodes. If a partition gives n_1 equal-sized blocks, each with n_2 low level nodes ($n_1 \times n_2 = n$), then the HADP algorithm has a time complexity $O(n_1^2)$ at the high level, and $O(l \times n_2^2)$ at

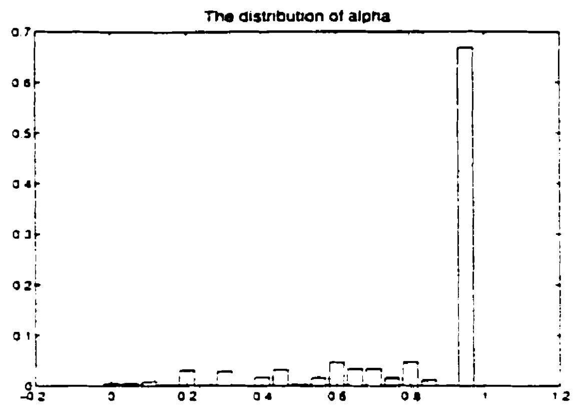

FIGURE 3.12. The distribution of α in Example 3.3

the low level. Here l is the number of blocks on the path obtained by the application of DP to the high level system. Therefore, the speed-up may be estimated by $O(n^2)/(O(n_1^2)+O(l*(n/n_1)^2))$. In many cases, l and n_1 are both approximately equal to $\sqrt{n}$, and hence an estimate of the increase of efficiency due to the use of HADP is $O(\sqrt{n})$.

Incidentally, for BMG problems, the effort needed to find an IBC block containing a given node v in a constraint with n_3 nodes is linear in n_3, because every node in this constraint only need labelling twice to see if it is reachable from and co-accessible to v. If we use n_3 equal-sized constraints in the pre-processing, i.e., $n_3 \times n_3 = n$, the time complexity of partition is $O(n^{\frac{3}{2}})$. To get D^- parameters for each block, we need $O(n_1 \times n_4 \times (n_2 + n_4)^2)$ extra computations if each block has n_4 in-set states. However, when s and t change, only parameters concerned with X_s and X_t need to be modified and so the remainder of the pre-processing can be done off-line.

CHAPTER 4

Relational Multi-agent Finite State Machines (MA($\mathcal{R}$) FSM)

4.1. Introduction

In the areas of transportation management, telecommunication networks and manufacturing systems, many problems involve multiple agents running interactively ([40]). Such multi-agent systems are distinguished from classical single agent systems in that both task specifications and cost functions may differ from agent to agent in the cooperative as well as in the competitive case. Due to the dynamical interactions between agents, and because of the inherent complexity of many physical systems, the analysis and control of multi-agent network systems often engenders problems of enormous complexity.

In this chapter, we formulate the notion of a relational multi-agent finite state machine. Consider a system of agents, each in the form of a forced event discrete event system, the dynamics of which are modelled by finite state machines, generally denoted $M = \{X, \Sigma, \delta\}$. We assume that the events always happen at discrete time instants. A *(state-event) configuration* of M is a pair of states and events (x, a), $x \in X, a \in \Sigma$, such that $\delta(x, a)!$, which is interpreted to mean that the system is in state x and is to take action (event) a. For n agents with models $M_1, M_2, \dots, M_n$, where $M_i = \{X_i, \Sigma_i, \delta_i\}$, $1 \leq i \leq n$, their joint configuration at time instant k is a vector $c(k) = [(x_1, a_1), (x_2, a_2), \dots, (x_n, a_n)]$, where $x_i \in X_i, a_i \in \Sigma_i$, and $\delta_i(x_i, a_i)!$ In the MA($\mathcal{R}$) FSM formulation, the relation $\mathcal{R}$ is a subset of $X_1 \times \Sigma_1 \times X_2 \times \Sigma_2 \times \dots \times X_n \times \Sigma_n$.

The interaction between n agents is represented by the collection of forbidden configurations $\mathcal{R}$. that is to say, the dynamics respecting the prohibitions $c(k) \notin \mathcal{R}$. for any configuration $c(k)$ at any time instant $k \in \mathcal{N}$.

To motivate this notion. consider a simple system with two interacting agents in the presence of synchronous events ([**41**], [**47**]). When two agents are at states $x_1 \in X_1$. $x_2 \in X_2$ respectively. where there exists a synchronous event $a \in \Sigma_1 \cap \Sigma_2$ such that $\delta_1(x_1, a)!$ and $\delta_2(x_2, a)!$. their interaction forces either both of the agents to take a or neither of them to take a. In other words. for this system. $\mathcal{R} = \{[(x_1.a_1).(x_2.a_2)] | (a_1 \neq a_2) \wedge [(\delta_1(x_1, a_1)! \wedge \delta_2(x_2.a_1)!) \vee (\delta_1(x_1.a_2)! \wedge \delta_2(x_2.a_2)!)]$ $a_1 \in \Sigma_1$. $a_2 \in \Sigma_2$. $x_1 \in X_1$, $x_2 \in X_2\}$. In this example. we note that if $a_1 \neq a_2$. $\delta(x_1.a_1)!$ but $\neg\delta(x_2.a_2)!$. then no transition is possible since the joint state cannot make a transition. and similarly in case $a_1 \neq a_2$. $\neq \delta(x_1, a_1)!$. $\delta(x_2.a_2)!$.

Other examples of forbidden interactions between two agents include mutually exclusive states: if $x_1.x_2$ cannot appear at the same time. then for all $a_1 \in \Sigma_1$. $a_2 \in \Sigma_2$. $[(x_1.a_1).(x_2.a_2)] \in \mathcal{R}_1$: and mutually exclusive events: if $a_1.a_2$ cannot happen at the same time. then for all $x_1 \in X_1$. $x_2 \in X_2$. $[(x_1.a_1).(x_2.a_2)] \in \mathcal{R}_2$.

In [**33**] and [**54**]. the formulation of a notion called multi-agent product for finite state machines and automata is presented. where the interaction between agents appears in the form of synchronous events.

In [**1**]. a theory of timed (finite) automaton is developed to model the behaviour of real-time systems. Timed words are infinite sequences of events each associated with a real-valued time. In hybrid systems. a set of guard conditions on the continuous states are imposed: only when these conditions are satisfied can a discrete transition take place ([**5**]. [**43**]). A finite set of clocks are attached to a finite automaton to keep track of the elapsed time. In [**69**], a control theory of timed discrete event systems is presented based on Ramadge-Wonham supervisory control theory: a special event, *tick*. represents a quantum time elapse between transitions. A hierarchical control of timed discrete event systems is developed in [**9**]. Usually, timed models possess much greater complexity than untimed discrete event system models.

A multi-agent system is said to be *synchronised* if all agents take actions simultaneously, that is to say, when a state transition happens in one agent, actions take place in all the other agents. One may view this to be a discrete clock shared by all the agents.

In this chapter, we first discuss multi-agent systems synchronised by a discrete time clock; then, second, a time counter is introduced to deal with multi-agent systems with non-simultaneous agents.

4.2. Synchronised Agents

In an event-driven finite state machine, $M = \{X, \Sigma, \delta\}$, events happen at discrete time instants and each state transition takes one unit time to complete, i.e., the *live time* of all events is 1. This will be called a (system) *quantum time unit* since it is the common indivisible minimum time for a system event to occur. The set of states X is partitioned into *transient states* X_T and *stable states* X_S. A *stable state* $x \in X_S$ is one that can be kept at the subsequent state transition; this of course is represented by a self-loop, i.e., there is $s_x \in \Sigma$ such that $\delta(x, s_x) = x$. *Transient states* are those without self-loops which cannot be held at the subsequent state transition. Once the system enters a transient state at any instant $k \in \mathcal{N}$, it takes the next action immediately, that is to say, the system will be in a distinct state at the instant $k+1$.

Definition 4.1. $((\|^{sim})_{i=1}^{n} M_i)$ The *simultaneous product machine* of n finite state machines $M_i = \{X_i, \Sigma_i, \delta_i\}, 1 \leq i \leq n$, is a finite state machine denoted by $(\|^{sim})_{i=1}^{n}(M_i) \equiv \{X, \Sigma, \delta\}$, where
$X = X_1 \times X_2 \times ... \times X_n$, $\Sigma = \Sigma_1 \times \Sigma_2 \times ... \times \Sigma_n$, and

$$\delta\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_n \end{bmatrix}\right) = \begin{cases} \begin{bmatrix} \delta_1(x_1, \sigma_1) \\ \delta_2(x_2, \sigma_2) \\ \vdots \\ \delta_n(x_n, \sigma_n) \end{bmatrix}, & \delta_i(x_i, \sigma_i)!,\ 1 \le i \le n. \\ \text{undefined}, & \text{otherwise}. \end{cases}$$

□

Evidently, $(||^{sim})_{i=1}^{n} M_i = M_1 ||^{sim} M_2 ||^{sim} \ldots ||^{sim} M_n$.

Example 4.1. In a game, at each discrete time instant, two players simultaneously make one of three gestures: scissors, hammer, cloth. This is a two-agent system with $\mathcal{R} = \emptyset$. □

Definition 4.2. ($(||_{-\mathcal{R}})_{i=1}^{n} M_i$) The *relational multi-agent product machine (with relation $\mathcal{R}$) ($MA(\mathcal{R})$)*, of n finite state machines $M_i = \{X_i, \Sigma_i, \delta_i\}, 1 \le i \le n$, is a finite state machine denoted by $(||_{-\mathcal{R}})_{i=1}^{n} M_i \equiv \{X, \Sigma, \delta\}$, where $X = X_1 \times X_2 \times \ldots \times X_n$, $\Sigma = \Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_n$, and

$$\delta\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_n \end{bmatrix}\right) = \begin{cases} \begin{bmatrix} \delta_1(x_1, \sigma_2) \\ \delta_2(x_2, \sigma_2) \\ \vdots \\ \delta_2(x_n, \sigma_n) \end{bmatrix}, & \begin{bmatrix} (x_1, \sigma_1) \\ (x_2, \sigma_2) \\ \vdots \\ (x_n, \sigma_n) \end{bmatrix} \notin \mathcal{R} \text{ and } \delta_i(x_i, \sigma_i)!,\ 1 \le i \le n. \\ \text{undefined} & \text{otherwise}. \end{cases}$$

□

Example 4.2. Suppose all streets in an area are one-way one-lanes. The actions an automobile may take at an intersection include: stop, go east-west bound and go south-north bound, i.e. $\Sigma = \{a, b, s\}$. This is shown in Figure 4.1. If two automobiles (modelled by $M_1 = M_2 = M$) meet at the same intersection, one of the automobiles has to stop for one (quantum) time unit. In this case, $\mathcal{R} = \{([x_1, \sigma_1], [x_2, \sigma_2]) \mid x_1 = x_2, \sigma_1, \sigma_2 \in \Sigma, \delta_1(x_1, \sigma_1) \neq x_1, \delta_2(x_2, \sigma_2) \neq x_2\}$. □

A multi-agent system consisting of $M_1, M_2, \ldots, M_n$, is said to be *controllable*, if $(||_{-\mathcal{R}})_{i=1}^{n} M_i$ is controllable, i.e., for any states $x, x' \in X$, $X = X_1 \times X_2 \times \ldots \times X_n$,

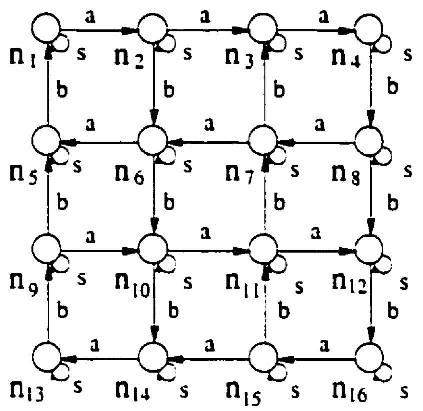

FIGURE 4.1. A Finite State Machine $M = \{X, \Sigma, \delta\}$

there exists $u \in (\Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_n)^*$, such that $\delta(x, u) = x'$.

It is straightforward to see that M_1 and M_2 are controllable if $M_1 \|^{sim} M_2$ is controllable. This is because, for any $x, x' \in X_1$, $y, y' \in X_2$, if there exists a sequence of input vectors $u([x, y], [x', y']) = [u_1(x, x'), u_2(y, y')] \in (\Sigma_1 \times \Sigma_2)^*$ which drives the two agent system from $[x, y]$ to $[y, y']$, then, for M_1 and M_2 respectively, $\delta_1(x, u_1(x, x')) = x'$ and $\delta_2(y, u_2(y, y')) = y'$. This is formalised by the following lemma.

Lemma 4.1. *Let M_i, $1 \le i \le n$ be n finite state machines. If $(\|^{sim})_{j=1}^{n} M_j$ is controllable, M_i is controllable, $1 \le i \le n$.* □

For a path $p(x, y) \in X_i^*$ between two states $x \in X_i$ and $y \in X_i$, define its *length* $|p(x, y)|$ as the number of state transitions on $p(x, y)$. A *circuit* is defined to be a path $p(x, y)$ for which $|p(x, y)| \ge 1$ and $x = y$. The following theorem gives a necessary and sufficient condition for a simultaneous product machine of two controllable finite state machines to be controllable. In the proof of this theorem, a "pumping" technique is used (see [**31**]).

Theorem 4.1. *For two controllable finite state machines M_1 and M_2, $|X_1| \ge 2$ or $|X_2| \ge 2$, $M_1 \|^{sim} M_2$ is controllable if and only if there are circuits C_1 and C_2 in X_1, X_2 respectively satisfying $|C_1| - |C_2| = \pm 1$.*

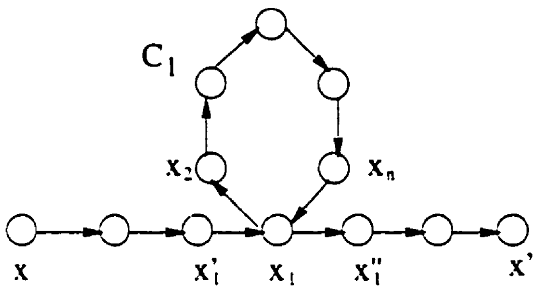

FIGURE 4.2. A circuit in M_1

Proof:

Consider arbitrary initial and final state pairs $[x, y], [x', y'] \in X_1 \times X_2$. Suppose C_1 and C_2 are circuits in M_1 and M_2 respectively, and $|C_1| - |C_2| = \pm 1$. Without loss of generality, let $|C_1| - |C_2| = 1$. Denote $C_1 = x_1 \to x_2 \to \dots \to x_{n-1}$, $x_1 = x_{n-1}, x_2, \dots, x_n \in X_1$, and $C_2 = y_1 \to y_2 \to \dots \to y_n$, $y_1 = y_n, y_2, \dots, y_{n-1} \in X_2$. Note that the states on these circuits are not necessarily distinct from one another. For any $x, x' \in X_1$ and $y, y' \in X_2$, because of the controllability of M_1 and M_2, there are paths $p_1(x, x') = x \to \dots x'_1 \to x_1 \to x''_1 \to \dots \to x'$ (see Figure 4.2) and $p_2^0(y, y') = y \to \dots \to y'_1 \to y_1 \to y''_1 \to \dots \to y'$. Clearly, by repeating circuit C_2 on $p_2^0(y, y')$, we may generate countably many distinct paths from y to y'. Choose a path $p_2(y, y')$ such that $|p_2(y, y')| - |p_1(x, x')| = k > 0$. Hence there exists a path

$$p'_1(x, x') = x \to \dots \to x'_1 \to \underbrace{C_1C_1 \dots C_1}_{k} \to x''_1 \to \dots \to x'.$$

from x to x' in M_1, and a path

$$p'_2(y, y') = y \to \dots \to y'_1 \to \underbrace{C_2C_2 \dots C_2}_{k} \to y''_1 \to \dots \to y'.$$

from y to y' in M_2. Since $|p'_1(x, x')| = |p_1(x, x')| + (|p_2(y, y')| - |p_1(x, x')|)|C_1|$, $|p'_2(y, y')| = |p_2(y, y')| + (|p_2(y, y')| - |p_1(x, x')|)|C_2|$, and $|C_1| - |C_2| = 1$, it follows that $|p'_1(x, x')| = |p'_2(y, y')|$. Therefore, jointly, $[p'_1(x, x'), p'_2(y, y')]$ is a vector path from $[x, y]$ to $[x', y']$ in $M_1\|^{sim}M_2$. Because x, x', y, y' are arbitrary, $M_1\|^{sim}M_2$ is controllable.

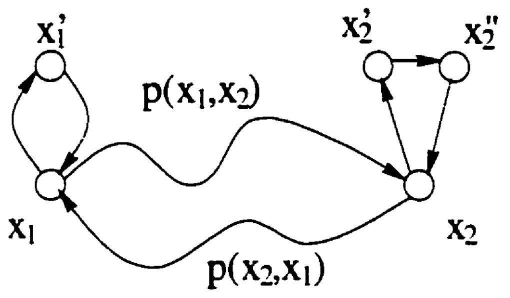

FIGURE 4.3. Circuits in M_1

Conversely, suppose $M_1||^{sim}M_2$ is controllable. Without loss of generality, let $|X_1| \geq 2$. Because M_1 is controllable, there exists a path $x_1 \rightarrow x_2$ in M_1, for which $x_1 \neq x_2$. By the controllability of $M_1||^{sim}M_2$, there is a path in $M_1||^{sim}M_2$ from $[x_2, y_1]$ to $[x_1, y_1]$ for any $y_1 \in X_2$. Because $x_1 \neq x_2$, this path is longer than 1. Let this path from $[x_3, y_2]$ to $[x_1, y_1]$ be $[p_1(x_2, x_1), p_2(y_1, y_1)]$ whose corresponding pair of paths in M_1, M_2 are $p_1(x_2, x_1)$, and $p_2(y_1, y_1)$, $|p_1(x_2, x_1)| = |p_2(y_1, y_1)| \leq 1$. Then, $C_1 = x_1 \rightarrow p_1(x_3, x_1)$ and $C_2 = p_2(y_2, y_1)$ are two circuits in M_1 and M_2 respectively. Obviously, $|C_1| - |C_2| = 1$. □

Corollary 4.1. *For two controllable finite state machines M_1 and M_2, if $|X_2| \geq 2$ and in M_1 there are circuits with 2 and 3 transitions respectively, then $M_1||^{sim}M_2$ is controllable.*

Proof:

Suppose in M_1, there is a circuit $C_1 = x_1 \rightarrow x_1' \rightarrow x_1$ from x_1 and back to x_1 with length of 2, and there is a circuit $C_2 = x_2 \rightarrow x_2' \rightarrow x_2'' \rightarrow x_2$ from x_2 to x_2 with $|C_1| = 3$ (Figure 4.3). Because M_1 is controllable, there is a path $p(x_1, x_2)$ from x_1 to x_2, and there is a path $p(x_2, x_1)$ from x_2 to x_1. Let $k = |p(x_1, x_2)| + |p(x_2, x_1)|$. Because M_2 is controllable and $|M_2| \geq 2$, we can obtain circuits longer than k by repeating cycles. Let C' be such a circuit in M_2 and $|C'| > k$.

If $|C'| - k = 2m - 1$ for some integer $m > 0$, construct a circuit from x_2 to x_2 in M_1 as follows:

$$C'' = p(x_2, x_1) \underbrace{\rightarrow x_1' \rightarrow x_1 \rightarrow x_1' \rightarrow \ldots \rightarrow x_1' \rightarrow}_{2m} p(x_1, x_2).$$

such that $|C''| - |C'| = |p(x_1, x_2)| + 2m + |p(x_2, x_1)| - |C'| = 1$.

If $|C'| - k = 2m$ for some integer $m > 0$, construct a circuit from x_1 to x_1 in M_1 as follows:

$$C'' = p(x_1, x_2) \underbrace{\rightarrow x_2' \rightarrow x_2'' \rightarrow}_{3} p(x_2, x_1) \underbrace{\rightarrow x_1' \rightarrow x_1 \ldots \rightarrow x_1' \rightarrow x_1}_{2(m-1)}$$

such that $|C''| - |C'| = |p(x_1, x_2)| + 2m + 1 + |p(x_2, x_1)| - |C'| = 1$.

Therefore, by Theorem 4.1, $M_1 ||^{sim} M_2$ is controllable. $\square$

Note that since a self-loop at a state generates circuits with arbitrary lengths, we obtain the following corollary as a direct application of Corollary 4.1.

Corollary 4.2. *For two controllable finite state machines M_1 and M_2, if there is a state in M_1 with a self-loop defined, then $M_1 ||^{sim} M_2$ is controllable.* $\square$

We may now make the important observation that since $M_1 ||^{sim} M_2$ may be taken as a finite state machine M', and since for any other finite state machine M'' we may repeat the argument above to inductively obtain the results concerning $(||^{sim})_{i=1}^n M_i$ for any $n > 2$. Set $M^1 = M_1$ and define finite state machines $M^i = M^{i-1} ||^{sim} M_i$, for $2 \leq i \leq n$. By the associativity of $||^{sim}$, a necessary and sufficient condition for the controllability of $(||^{sim})_{i=1}^n M_i$ is given by a test as to whether there are two circuits in M^{i-1} and M_i with the length difference of 1, $2 \leq i \leq n$.

Theorem 4.2. *Let M_i, $1 \leq i \leq n$ $(n \geq 2)$, be n controllable finite state machines. Then $(||^{sim})_{i=1}^n M_i$ is controllable if and only if for any M_i, there is a circuit C_i such that there are, respectively, circuits C_j^i in M_j, $1 \leq j \leq n$, $j \neq i$, and each $|C_j^i|$ is co-prime with $|C_i|$.*

Proof:

$\Longrightarrow$

Suppose for any M_i, $1 \leq i \leq n$, there is a circuit C_i such that there are circuits C_j^i in M_j, $1 \leq j \leq n$, $j \neq i$, such that each $|C_j^i|$ is co-prime with $|C_i|$. For arbitrary circuits C_i' and C_j', respectively, in M_i and M_j, $1 \leq i, j \leq n$, $i \neq j$, respectively, by repeating C_i' $|C_j'|$ times in M_i and repeating C_j' $|C_i'|$ times in M_j, we can obtain a circuit with length $|C_i'||C_j'|$ in $M_i||^{sim}M_j$. As a result, in $(||^{sim})_{i=1}^{k}(M_i)$, $1 \leq k \leq n-1$, there are circuits of length $l\prod_{j=1}^{k}|C_j^{k+1}|$, where l can be an arbitrary positive integer. Also, by repeating the circuit C_{k+1} m times, we can obtain a circuit with length $m|C_{k+1}|$ in M_{k+1}, where m can be an arbitrary positive integer.

By the running assumption, $|C_{k+1}|$ is co-prime with all $|C_j^{k+1}|$, $1 \leq j \leq k$, thus $|C_{k+1}|$ is co-prime with $\prod_{j=1}^{k}|C_j^{k+1}|$. Therefore, there are integers l_1, l_2 such that $l_1(\prod_{j=1}^{k}|C_j^{k+1}|) - l_2|C_{k+1}| = 1$ ([**24**]). Recursively applying Theorem 4.1, it follows that $(||^{sim})_{i=1}^{n}M_i$ is controllable.

$\Longleftarrow$

Suppose $(||^{sim})_{i=1}^{n}M_i$ is controllable. By the associativity of $||^{sim}$, for arbitrary i, $1 \leq i \leq n$, $(||^{sim})_{i=1}^{n}M_i = ((||^{sim})_{j=1,j\neq i}^{n}M_j)\ ||^{sim}M_i$. By Lemma 4.1, $(||^{sim})_{j=1,j\neq i}^{n}M_j$ is controllable. Thus, according to Theorem 4.1, there are circuits C' and C_i respectively in $(||^{sim})_{j=1,j\neq i}^{n}M_j$ and M_i such that $||C'| - |C_i|| = 1$. In other words, $|C'|$ and $|C_i|$ are co-prime. Clearly, since C' is a circuit in $(||^{sim})_{j=1,j\neq i}^{n}M_j$, there are circuits C_j in M_j correspondent to C' with length $|C_j| = |C'|$, $1 \leq j \leq n$, $j \neq i$. Hence, $|C_i|$ and $|C_j|$ are co-prime, $1 \leq j \leq n$, $j \neq i$. □

Theorem 4.2 is a necessary and sufficient condition for $(||^{sim})_{i=1}^{n}M_i$ to be controllable. For example, in case $n = 3$, assume there are circuits in M_1 with length 2 and 3, the length of the only circuits in M_2 are $3m$, the length of the only circuit in M_3 is $4n$, $m, n \in \mathcal{N}^+$. Because there is a circuit in $M_1||^{sim}M_2$ with length 3, which is co-prime with 4, the length of a circuit in M_3, $M_1||^{sim}M_2||^{sim}M_3$ is controllable. Suppose there were no circuit in M_1 with length co-prime with both 3 and 4. Since the lengths of the circuits in $M_2||^{sim}M_3$ are multiples of 12, the lengths of all circuits in M_1 thus would not be co-prime with the length of any circuit in $M_2||^{sim}M_3$. Hence, no circuits C_1 in M_1 and C' in $M_2||^{sim}M_3$ such that $||C_1| - |C'|| = 1$. This contradicts

Theorem 4.1. Therefore, there is a circuit in M_1 with length co-prime with both 3 and 4.

The following theorem states that the controllability of a multi-agent system with $\mathcal{R} \neq \emptyset$ is dependent on the state-to-state reachability of the state pairs given in $\mathcal{R}$.

Theorem 4.3. *For two controllable finite state machines M_1 and M_2, $M_1 \|_{\neg\mathcal{R}} M_2$ is controllable if and only if the following two conditions hold:*
(1) $M_1 \|^{sim} M_2$ is controllable.
(2) For any $[(x_1, \sigma_1), (x_2, \sigma_2)]) \in \mathcal{R}$, there are $s_1 = a_1 a_2 ... a_k \in \Sigma_1^*$ and $s_2 = b_1 b_2 ... b_k \in \Sigma_2^*$ such that $[(\delta_1(x_1, a_1 a_2 ... a_i), a_{i+1}), \delta_2(x_2, b_1 b_2 ... b_i), b_{i+1})] \notin \mathcal{R}$, for all $1 \leq i \leq k$, and such that $[\delta_1(x_1, s_1), \delta_2(x_2, s_2)] = [\delta_1(x_1, \sigma_1), \delta_2(x_2, \sigma_2)]$.

Proof:

The "only if" direction follows immediately from the definitions.

For the "if" direction, suppose conditions 1 and 2 hold. For any vector states $[x, y]$ and $[x', y']$, because $M_1 \|^{sim} M_2$ is controllable, there is a control $u([x, y], [x', y']) = [u_1(x, x'), u_2(y, y')] \in (\Sigma_1 \times \Sigma_2)^*$ driving $M_1 \|^{sim} M_2$ from $[x, y]$ to $[x', y']$. Denote

$$u([x, y], [x', y']) = \begin{bmatrix} c_1 c_2 ... c_n \\ d_1 d_2 ... d_n \end{bmatrix}.$$

Denote $x_i = \delta_1(x, c_1 c_2 ... c_i)$ and $y_i = \delta_2(y, d_1 d_2 ... d_i)$, $1 \leq i < n$, where $x_1 = x, y_1 = y$, $x_n = x', y_n = y'$.

By Condition (2), if there are any x_i and y_i, $1 \leq i < n$, such that $[(x_i, c_{i+1}), (y_i, d_{i+1})] \in \mathcal{R}$, there exists $[s_1^i, s_2^i]$, such that for all $v_1, w_1 \in \Sigma_1^*, a \in \Sigma$ and $v_2, w_2 \in \Sigma_2^*, b \in \Sigma_2$, $s_1^i = v_1 a w_1$, $s_2^i = v_2 b w_2$ and $|v_1| = |v_2|$,

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} (\delta_1(x_i, v_1), a) \\ (\delta_2(y_i, v_2), b) \end{bmatrix} \notin \mathcal{R}.$$

Replacing all such $[c_{i+1}, d_{i+1}]$ in $u([x, y], [x', y'])$ by $[s_1^i, s_2^i]$, we obtain a control sequence $u'([x, y], [x', y'])$ steering $M_1 \|^{sim} M_2$ from $[x, y]$ to $[x', y']$ without going through any configuration in $\mathcal{R}$. That is to say, $u'([x, y], [x', y'])$ is a valid control from $[x, y]$ to $[x', y']$ in $M_1 \|_{\neg\mathcal{R}} M_2$. Hence, $M_1 \|_{\neg\mathcal{R}} M_2$ is controllable. $\square$

For two states $x \in X_1, y \in X_2$, and two sequences of actions $s_1 = a_1 a_2 \dots a_n \in \Sigma_1^*$, $s_2 = b_1 b_2 \dots b_n \in \Sigma_2^*$, if

$$\begin{bmatrix} (\delta_1(x, a_1 a_2 \dots a_j), a_{j+1}) \\ (\delta_2(y, b_1 b_2 \dots b_j), b_{j+1}) \end{bmatrix} \notin \mathcal{R},$$

for all $1 \leq i < n$, write $[(x, s_1), (y, s_2)] \cap \mathcal{R} = \emptyset$.

We observe that for n finite state machines, M_i, $1 \leq i \leq n$, if (1) $(\|^{sim})_{i=1}^n (M_i)$ is controllable, and if (2) for any $[(x_1, a_1), (x_2, a_2), \dots, (x_n, a_n)] \in \mathcal{R}$ with $\delta_i(x_i, a_i)!$, $1 \leq i \leq n$, there is a vector string $[s_1, s_2, \dots, s_n] \in \Sigma_1 \times \Sigma_2 \times \dots \times \Sigma_n$, $|s_1| = |s_2| = \dots = |s_n|$, such that $\delta_i(x_i, s_i) = \delta_i(x_i, a_i)$ and $[(x_1, s_1), (x_2, s_2), \dots, (x_i, s_i)] \cap \mathcal{R} = \emptyset$ (that is to say, there is a path in $(\|_{\neg\mathcal{R}})_{i=1}^n M_i$ from $[x_1, x_2, \dots, x_n]$ to $[\delta_1(x_1, a_1), \delta_2(x_2, a_2), \dots, \delta_n(x_n, a_n)]$), $(\|_{\neg\mathcal{R}})_{i=1}^n M_i$ is controllable.

We shall apply the following theorem in Example 4.4.

Theorem 4.4. *For two controllable finite state machines M_1 and M_2, $M_1 \|_{\neg\mathcal{R}} M_2$ is controllable if and only if the following three conditions are satisfied.*
(1) For all $[(x, \sigma_1), (y, \sigma_2)] \in \mathcal{R}$, there exists $[s_1, s_2] \in (\Sigma_1 \times \Sigma_2)^*$ where $s_1 = a_1 a_2 \dots a_n$ and $s_2 = b_1 b_2 \dots b_n$, such that,

$$\begin{bmatrix} (x, s_1) \\ (y, s_2) \end{bmatrix} \cap \mathcal{R} = \emptyset.$$

and such that $\delta_1(x, s_1) = \delta_1(x, \sigma_1)$;
(2) For all $[(x, \sigma_1), (y, \sigma_2)] \in \mathcal{R}$, there exist $y' \in X_2$ and $[s_1, s_2] \in (\Sigma_1 \times \Sigma_2)^*$ where $s_1 = a_1 a_2 \dots a_n$ and $s_2 = b_1 b_2 \dots b_n$, such that,

$$\begin{bmatrix} (x, s_1) \\ (y', s_2) \end{bmatrix} \cap \mathcal{R} = \emptyset.$$

and such that $\delta([x, y'], [s_1, s_2]) = [\delta_1(x, \sigma_1), \delta_2(y, \sigma_2)]$;
(3) There exists $x_0 \in X_1$ such that for all $y_1 \in X_2$ and $y_2 \in X_2$, there exists $[s_1, s_2] \in$

$(\Sigma_1 \times \Sigma_2)^*$ *where* $s_1 = a_1a_2...a_n$ *and* $s_2 = b_1b_2...b_n$, *such that*

$$\begin{bmatrix} (x_0, s_1) \\ (y_1, s_2) \end{bmatrix} \cap \mathcal{R} = \emptyset.$$

and such that $\delta([x_0, y_1], [s_1, s_2]) = [x_0, y_2]$. □

These conditions can be paraphrased as follows: (1) means $M_1||_{\neg\mathcal{R}}M_2$ can escape from $\mathcal{R}$-configurations with the M_1 move reproduced with non-$\mathcal{R}$-configuration moves; (2) says all targets of $\mathcal{R}$-configurations, $[\delta_1(x, \sigma_1), \delta_2(y, \sigma_2)]$, can be reached from a pair of initial states $[x, y']$ with a sequence of non-$\mathcal{R}$-configuration moves; (3) means that M_1 can loop at x_0 while M_2 moves from y_1 to y_2 with non-$\mathcal{R}$-configuration moves.

Although all assumptions are made with M_1, the active agent, the conclusion holds, symmetrically, if all conditions for M_1 and M_2 are swapped.

Proof:

Suppose the above conditions (1)-(3) are true. We shall demonstrate that these imply the conditions of Theorem 4.3 and hence that $M_1||_{\neg\mathcal{R}}M_2$ is controllable.

Let $x_0 \in X_1$ be the state in X_1 such that Condition (3) holds. Consider any forbidden configuration $[(x, \sigma_1), (y, \sigma_2)] \in \mathcal{R}$; because M_1 is controllable, there exists a finite sequence of actions $s = a_1a_2...a_n \in \Sigma_1^*$ such that $\delta_1(x, s) = x_0$.

Set $i = 1$, $x_1 = x$, $y_1 = y$, $s_1 = \epsilon$ and $s_2 = \epsilon$ (we recall that ϵ is an empty string, not a self-loop action). By Condition (1), there is $[s_1^i, s_2^i] \in (\Sigma_1 \times \Sigma_2)^*$ such that $\delta_1(x, s_1^i) = \delta_1(x_i, a_i)$ and $[s_1^i, s_2^i]$ steers the system from $[x_i, y_i]$ to $[\delta_1(x_i, a_i), \delta_2(y_i, s_2^i)]$ with $[(x_i, s_1^i), (y_i, s_2^i)] \cap \mathcal{R} = \emptyset$.

Let $i = i + 1$, $s_1 = s_1s_1^i$ and $s_2 = s_2^i$. Denote $x_i = \delta_1(x, s_1)$, $y_i = \delta_2(y, s_2)$. Recursively continue this procedure until $\delta_1(x, s_1) = x_0$. We obtain $[s_1, s_2] \in (\Sigma_1 \times \Sigma_2)^*$

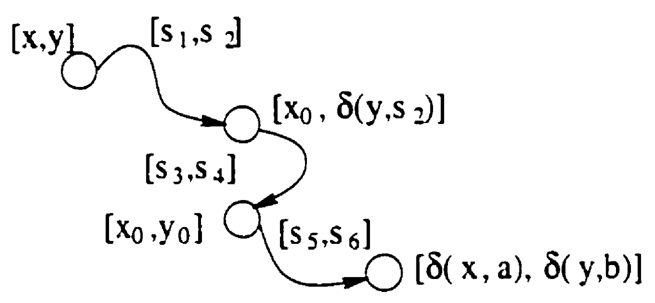

FIGURE 4.4. A path in $M_1\|_{\neg\mathcal{R}}M_2$ from $[x.y]$ to $[\delta_1(x.\sigma_1).\delta_2(y.\sigma_2)]$

such that $\delta([x.y].[s_1.s_2]) = [x_0.\delta_2(y.s_2)]$ and $[(x.s_1).\ (y.s_2)]\cap\mathcal{R} = \emptyset$.

Similarly. by Condition (2). working backwards from $[\delta_1(x.\sigma_1).\ \delta_2(y.\sigma_2)]$. we can find $[s_5.s_6] \in (\Sigma_1\times\Sigma_2)^*$ such that $\delta([x_0.y_0].[s_5.s_6]) = [\delta_1(x.\sigma_1).\ \delta_2(y.\sigma_2)]$ for some $y_0 \in X_2$. and the path driven by $[s_5.s_6]$ does not go through $\mathcal{R}$.

By conditions (3). there is $[s_3.s_4] \in (\Sigma_1\times\Sigma_2)^*$ such that $\delta([x_0.\delta_2(y.s_2)].[s_3.s_4]) = [x_0.y_0]$ and $[(x_0.s_3).\ (\delta_2(y.s_2).s_4)]\cap\mathcal{R} = \emptyset$.

Therefore. $\delta([x.y].[s_1s_3s_5.s_2s_4s_6]) = [\delta_1(x.\sigma_1).\delta_2(y.\sigma_2)]$ (see Figure 4.4) and it follows recursively that $[(x.s_1s_3s_5).\ (\delta_2(y.s_2s_4s_6)]\cap\mathcal{R} = \emptyset$. According to Theorem 4.3. $M_1\|_{\neg\mathcal{R}}M_2$ is controllable.

The "only if" direction follows immediately from the definitions. □

A finite state machine M is said to be *controllable with respect to* $\Sigma'\subset\Sigma$ if all states of M are mutually accessible via transition events restricted to Σ'. Clearly, if $M_1\|_{\neg\mathcal{R}}M_2$ is controllable. and $[(x.a),(y.b)]\in\mathcal{R}$ for some $x\in X_1$. $a\in\Sigma_1$. $y\in X_2$ and $b\in\Sigma_2$. then M_1 and M_2 must be controllable with respect to $\Sigma_1-\{a\}$ and $\Sigma_2-\{b\}$. If two finite state machines M_1 and M_2 are controllable, the following theorem gives a sufficient condition for $M_1\|_{\neg\mathcal{R}}M_2$ to be controllable.

Theorem 4.5. *For two controllable finite state machines M_1 and M_2. if there exist $E_1\subset\Sigma_1.E_2\subset\Sigma_2$ such that $\mathcal{R}\subseteq(X_1\times E_1\times X_2\times\Sigma_2)\cup(X_1\times\Sigma_1\times X_2\times E_2)$. then $M_1\|_{\neg\mathcal{R}}M_2$ is controllable if the following conditions are satisfied:*

(1) There exist circuits C_1, C_2 in M_1 or M_2 with events in $\Sigma_1 - E_1$ and $\Sigma_2 - E_2$ respectively, and $|C_1| - |C_2| = \pm 1$.
(2) M_1 is controllable with respect to $\Sigma_1 - E_1$.
(3) M_2 is controllable with respect to $\Sigma_2 - E_2$.

Proof:

It is sufficient to prove that $M_1 \|^{sim} M_2$ is controllable with respect to $(\Sigma_1 - E_1) \times (\Sigma_2 - E_2)$ to demonstrate that $M_1 \|_{\neg\mathcal{R}} M_2$ is controllable, and under hypotheses (1)-(3), this follows from Corollary 4.1. □

Example 4.3. Two finite state machines M_1 and M_2 are shown in Figure 4.5.

A list of the mutually exclusive events in M_1 and M_2 are:
(1) events in $\{a_1, b_1, d_1\}$ and events in $\{b_2, d_2, e_2\}$;
(2) c_1 and events in $\{a_2, b_2, c_2, d_2\}$;
(3) events in $\{e_1, f_1\}$ and events in $\{b_2, d_2\}$.

Thus, the interaction of M_1 and M_2 is given by a forbidden set

$$\begin{aligned}\mathcal{R} = \{ \; & [(x_1, a_1), (y_2, b_2)], \; [(x_1, a_1), (y_3, d_2)], \; [(x_1, a_1), (y_1, e_2)], \\ & [(x_2, b_1), (y_1, e_2)], \; [(x_2, b_1), (y_2, b_2)], \; [(x_2, b_1), (y_3, d_2)], \\ & [(x_2, c_1), (y_a, a_2)], \; [(x_2, c_1), (y_2, b_2)], \; [(x_2, c_1), (y_2, c_2)], \\ & [(x_2, c_1), (y_3, d_2)], \; [(x_3, d_1), (y_1, e_2)], \; [(x_3, d_1), (y_2, b_2)], \\ & [(x_3, d_1), (y_3, d_2)], \; [(x_1, e_1), (y_2, b_2)], \; [(x_1, e_1), (y_3, d_2)], \\ & [(x_3, f_1), (y_2, b_2)], \; [(x_3, f_1), (y_3, d_2)]\}.\end{aligned}$$

Set $E_1 = \{c_1, d_1\}$ and $E_2 = \{b_2, d_2, e_2\}$. It is easy to verify that without E_1 and E_2 respectively, M_1 and M_2 are both controllable. And without E_1, there are circuits $x_1 \to x_2 \to x_1$ with length of 2, $x_1 \to x_3 \to x_2 \to x_1$ with length of 3 in M_1. By the theorem above, $M_1 \|_{\neg\mathcal{R}} M_2$ is controllable.

□

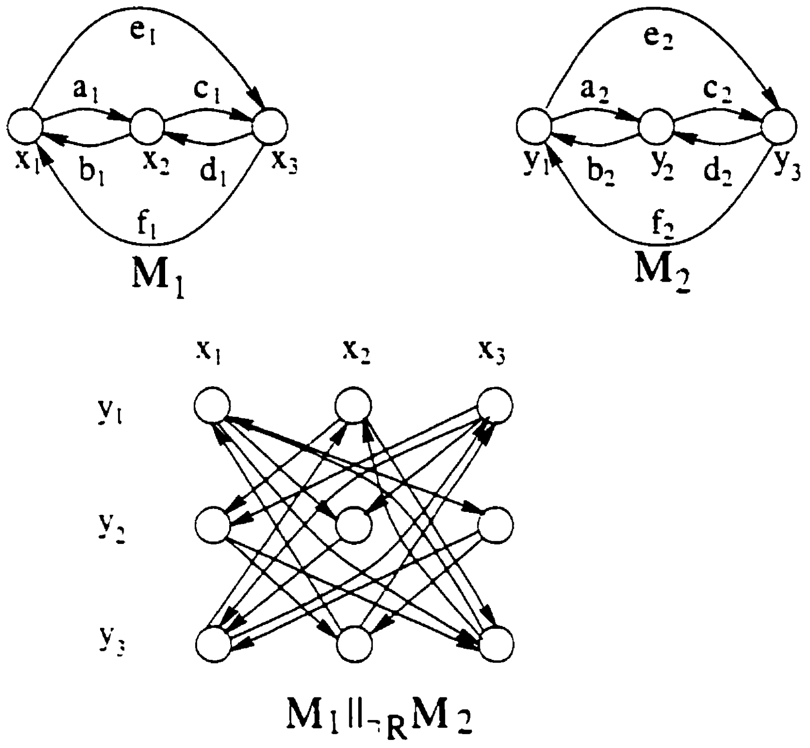

FIGURE 4.5. Two interacting agents

Let a finite state machine with cost be denoted by $M = \{X, \Sigma, \delta, d\}$, where $d : X \times \Sigma \rightarrow \mathcal{R}^+$ associates an event taken at a state with a non-negative cost. The joint cost of a multi-agent system configuration may be represented by a function of the component costs, i.e., $d([(x_1, a_1), (x_2, a_2), \ldots, (x_n, a_n)]) = f(d_1(x_1, a_1), d_2(x_2, a_2), \ldots, d_n(x_n, a_n))$. f may take different forms in various contexts. The state to state optimal control problem of multi-agent systems of the $MA(\mathcal{R})$ FSM type can then in principle be solved by applying dynamic programming in the $MA(\mathcal{R})$ model.

Example 4.4. In Example 4.2, two automobiles at s_1 and s_2 respectively are going to t_1 and t_2. Suppose in M, $d(x, a) = d(x, b) = 1$, and $d(x, s) = 5$ for all $x \in X$. Since a task is completed after an automobile arrives at its goal, the joint cost of $M\|_{\neg\mathcal{R}}M$ takes the form

$$d([(x_1, \sigma_1), (x_2, \sigma_2)]) = \begin{cases} d(x_1, \sigma_1) + d(x_2, \sigma_2) & \text{if } x_1 \neq t_1, x_2 \neq t_2 \\ d(x_1, \sigma_1) & \text{if } x_1 \neq t_1,\ x_2 = t_2,\ \sigma_2 = s \\ d(x_2, \sigma_2) & \text{if } x_1 = t_1,\ \sigma_1 = s,\ x_2 \neq t_2 \end{cases}.$$

If two automobiles meet at an intersection, one can continue while the other stops, that is to say, the first condition of Theorem 4.4 holds. For any $[(n_i, \sigma_1), (n_i, \sigma_2)] \in \mathcal{R}$,

$1 \leq i \leq 16$, $\sigma_1, \sigma_2 \neq s$, reset the second automobile to $\delta_2(n_i, \sigma_2)$ and let it take action s while the first one takes σ_1 at n_i, the second condition of Theorem 4.4 is satisfied. We know that the first automobile may stay at any n_i while the second one moves from n_k to n_j, $1 \leq i, j, k \leq 16$, hence the third condition of Theorem 4.4 is true. Therefore, $M\|_{\neg\mathcal{R}}M$ is controllable, that is to say, for any $[s_1, s_2], [t_1, t_2]$, the optimal control exists. For example, when $[s_1, s_2] = [n_6, n_9]$, $[t_1, t_2] = [n_{11}, n_{15}]$, one optimal control is $[ababba, aaabas]$ with the joint cost of 11. $\square$

Since a multi-agent system has a much larger state space than its component finite state machines, it is extremely useful if one can obtain an IBC partition machine of $MA(\mathcal{R})$ based on partition machines of individual M_i's . This could be followed by successive aggregation cycles. Assume this significant step can be carried out for a system $MA(\mathcal{R})$, we have the following application of the HADP to multi-agent systems.

HADP for MA($\mathcal{R}$) Systems

(1) Compute the $MAP(\mathcal{R})$ $(\|_{\neg\mathcal{R}})_{i=1}^{n} M_i$ of n interacting finite state machines M_i, $1 \leq i \leq n$.
(2) Successively create a hierarchy of IBC partition machines with base machine $(\|_{\neg\mathcal{R}})_{i=1}^{n} M_i$.
(3) Apply HADP to the hierarchy. $\square$

In the following corollary, Condition (2) implies the conditions (1) and (2) of Theorem 4.4.

Corollary 4.3. *Let $X_1 \in \pi_1$ and $X_2 \in \pi_2$ be IBC and the conditions (1) and (2) hold:*
(1) For any $x \in X_1$, $y \in X_2$, $a \in \Sigma_1, b \in \Sigma_2$ such that $\delta_1(x, a) \in X_1$, $\delta_2(y, b) \in X_2$ and $[(x, a), (y, b)] \in \mathcal{R}$, there exists $b' \in \Sigma_2$ such that $\delta_2(y, b') \in X_2$ and $[(x, a), (y, b')] \notin \mathcal{R}$.
(2) There exists $x_0 \in X_1$, $c \in \Sigma_1$ such that $\delta_1(x_0, c) = x_0$ and for all $y \in X_2$, $b \in \Sigma_2$ such that $\delta_2(x, b) \in X_2$, $[(x_0, c), (y, b)] \notin \mathcal{R}$. Then $X_1 \times X_2$ is IBC. $\square$

If $X_i^1 \in \pi_1$ and $X_j^2 \in \pi_2$, let $\mathcal{R}(X_i^1, X_j^2)$ be the set of forbidden configurations restricted to $X_i^1 \times X_j^2$, i.e., $\mathcal{R}(X_i^1, X_j^2) = \{[(x, a), (y, b)] \in \mathcal{R} | [x, y] \in X_i^1 \times X_j^2\}$.

Define $\Sigma_j^i = \{a \in \Sigma_i | \delta(x,a)! \text{ for some } x \in X_j^i\}$ for $i = 1,2$, $1 \leq j \leq |\pi_i|$. According to Theorem 4.5, we obtain the corollary below.

Corollary 4.4. *If $X_i^1 \in \pi_1$ and $X_j^2 \in \pi_2$ are IBC, and there exist $E_i^1 \subseteq \Sigma_i^1$ and $E_j^2 \in \Sigma_j^2$ such that for all $x \in X_i^1, y \in X_j^2$, $a \in \Sigma_i^1, b \in \Sigma_j^2$, the configuration $[(x,a),(y,b)] \notin \mathcal{R}(X_i^1, X_j^2)$, and the following conditions hold.*
(1) There exist circuits $C(X_i^1), C(X_j^2)$ in X_1 and X_2 with events in $\Sigma_i^1 - E_i^1$, $\Sigma_j^2 - E_j^2$ respectively, and $|C(X_i^1)| - |C(X_j^2)| = \pm 1$.
(2) X_i^1 is IBC with respect to $\Sigma_i^1 - E_i^1$.
(3) X_j^2 is IBC with respect to $\Sigma_j^2 - E_j^2$.
Then $X_i^1 \times X_j^2$ is an IBC block of $M_1 ||_{-\mathcal{R}} M_2$. □

If $X_i^1 \times X_j^2$ is IBC, for all $X_i^1 \times X_j^2 \in \pi_1 \times \pi_2$, the dynamics of an IBC partition machine of $M_1 ||_{-\mathcal{R}} M_2$ based on $\pi_1 \times \pi_2$ are given by the DC relations between elements of $\pi_1 \times \pi_2$. If there exists $x \in O(X_i^1), a \in \Sigma_i^1$, such that $\delta_1(x,a) \in I(X_k^1)$, and there exists $[y,b] \in X_j^2 \times \Sigma_j^2$, $\delta_2(y,b) \in X_l^2$, $[(x,a),(y,b)] \notin \mathcal{R}$, then $\langle [X_i^1, X_j^2], [X_k^1, X_l^2] \rangle$ is DC.

4.3. Events with Different Live Times

Up to this point, all events have been assumed to have the same live time, that is to say, all transitions are assumed to take the same quantum time unit (taken to be 1 for convenience) to complete. We now consider systems in which distinct events may have different live times. In a timed finite state machine $M = \{X, \Sigma, \delta, t\}$, an event $\sigma \in \Sigma$ is associated with a time measurement $t(\sigma) > 0$ that indicates the live time of an event. Although in practice, events may occur at any real-valued time, in this setting, a discrete-time clock is used. Hence, we assume that all events have durations $t(\sigma)$, which are integral multiples of the basic quantum time interval which is again taken to be 1.

In many cases, minimal time control problems for multi-agent systems are of interest. This is the problem of seeking a controlled path between two vector states that takes the minimal time to finish. In this section, a simplified clock structure, a *time counter*, is proposed to record the difference of live times of two events taken by two agents. With the help of such a time counter, a minimal time control problem for multi-agent systems may be transfered into a point to point shortest path problem and thus solved with HADP.

For a system consisting of n agents modelled by finite state machines M_i, $1 \leq i \leq n$, a time counter observes the progress of the running events in each of M_i. Since distinct events may have different live times, when the systems M_i take events in parallel, the state transitions in n agents may not complete at the same time instant. The function of a time counter is to record the residual live times of the events being executed at M_j, $1 \leq j \leq n$, $j \neq i$, at the moment an state transition in M_i is completed.

A *time counter* for two timed finite state machines $M_1 = \{X_1, \Sigma_1, \delta_1, t_1\}$ and $M_2 = \{X_2, \Sigma_2, \delta_2, t_2\}$ is a third finite state machine $C(M_1, M_2) = \{D_t, \Sigma_t, \delta_t, t\}$. The set of counter states, $D_t = \{[(\Sigma_1 \times \{0, 1, 2, ...T_2\}) \cup (\Sigma_2 \times \{0, 1, 2, ...T_1\})] \times \{1, 2\}\} \cup \{RESET\}$, where $T_1 = \max\{t_1(a) - t_2(b), 0\}$, $T_2 = \max\{t_2(b) - t_1(a), 0\}$ for all $a \in \Sigma_1, b \in \Sigma_2$, and $RESET$ is a distinguished state representing the reset of the counter to zero. A state other than $RESET$ is a triple consisting of an event and two integers, which give information about the residual live time for a running event that is being executed by an agent. The set of vector events is $\Sigma_t = [(\Sigma_1 \cup \{-\}) \times \Sigma_2] \cup [\Sigma_1 \times (\Sigma_2 \cup \{-\})]$, where the symbol $-$ means that an agent is in the process of executing an incomplete event. The counter state transition function is defined as follows:

$$\delta_t(RESET, \begin{bmatrix} a \\ b \end{bmatrix}) = \begin{cases} (a, t_1(a) - t_2(b), 1), & \text{if } t_1(a) - t_2(b) > 0 \\ RESET, & \text{if } t_1(a) - t_2(b) = 0 \\ (b, t_2(b) - t_1(a), 2), & \text{if } t_1(a) - t_2(b) < 0 \end{cases}$$

$$\text{if } [a, b] \in \Sigma_1 \times \Sigma_2.$$

$$\delta_t((a, \tau, 1), \begin{bmatrix} - \\ b \end{bmatrix}) = \begin{cases} (a, \tau - t_2(b), 1), & \text{if } \tau - t_2(b) > 0 \\ RESET, & \text{if } \tau - t_2(b) = 0 \\ (b, t_2(b) - \tau, 2), & \text{if } \tau - t_2(b) < 0 \end{cases}$$

and,

$$\delta_t((b, \tau, 2), \begin{bmatrix} a \\ - \end{bmatrix}) = \begin{cases} (a, t_1(a) - \tau, 1), & \text{if } t_1(a) - \tau > 0 \\ RESET, & \text{if } t_1(a) - \tau = 0 \\ (b, \tau - t_1(a), 2), & \text{if } t_1(a) - \tau < 0 \end{cases}.$$

Otherwise, δ_t is undefined. The time measurement $t : D_t \times \Sigma_t \rightarrow \mathcal{N}^+$ is defined by:

$$t(RESET, \begin{bmatrix} a \\ b \end{bmatrix}) = \min(t_1(a), t_2(b)),$$

$$t((a, \tau, 1), \begin{bmatrix} - \\ b \end{bmatrix}) = \min(\tau, t_2(b)),$$

and

$$t((b, \tau, 2), \begin{bmatrix} a \\ - \end{bmatrix}) = \min(t_1(a), \tau).$$

Consider two agents M_1 and M_2 interacting with one another with the forbidden relation denoted by $\mathcal{R} \subseteq \Sigma_1 \times \Sigma_2$. $[a, b] \in \mathcal{R}$ means two events $a \in \Sigma_1$ and $b \in \Sigma_2$ are forbidden to happen at the same time. This is a simplified version of the forbidden configurations introduced earlier.

Definition 4.3. ($M_1(||^t_{-\mathcal{R}})M_2$) For two finite state machines M_1 and M_2, their *timed multi-agent product machine* $M_1(||^t_{-\mathcal{R}})M_2 = \{X_1 \times X_2 \times D_t, \Sigma_t - \mathcal{R}, \delta, t\}$, has the state transition function defined as follows:

$$\delta(\begin{bmatrix} x \\ y \\ RESET \end{bmatrix}, \begin{bmatrix} a \\ b \end{bmatrix}) = \begin{bmatrix} \delta_1(x, a) \\ \delta_2(y, b) \\ \delta_t(RESET, \begin{bmatrix} a \\ b \end{bmatrix}) \end{bmatrix},$$

if $[a, b] \notin \mathcal{R}$, $a \neq -, b \neq -$, $\delta_1(x, a)!$ and $\delta_2(y, b)!$,

$$\delta(\begin{bmatrix} x \\ y \\ (c, \tau, 1) \end{bmatrix}, \begin{bmatrix} a \\ b \end{bmatrix}) = \begin{bmatrix} x \\ \delta_2(y, b) \\ \delta_t(s, \begin{bmatrix} a \\ b \end{bmatrix}) \end{bmatrix},$$

if $a = -$, $\delta_2(y,b)!$ and $[c,b] \notin \mathcal{R}$,

$$\delta(\begin{bmatrix} x \\ y \\ (d,\tau,2) \end{bmatrix}, \begin{bmatrix} a \\ b \end{bmatrix}) = \begin{bmatrix} \delta_1(x,a) \\ y \\ \delta_t(s, \begin{bmatrix} a \\ b \end{bmatrix}) \end{bmatrix},$$

if $b = -$, $\delta_1(x,a)!$ and $[a,d] \notin \mathcal{R}$.

□

The time counter records the residual live times of the events running in all of the agents. In $M_1(||^t_{\neg\mathcal{R}})M_2$, a state transition takes place whenever a state transition of one of the agents is completed.

The control problem for two interacting timed agents M_1, M_2, is to find a sequences of (vector) events that steers the multi-agent system from the start (vector) state $[x,y]$ to the target state $[x',y']$ such that the system enters $[x',y']$ at the same time point, i.e., find a path from $[x,y,RESET]$ to $[x',y',RESET]$ in $M_1(||^t_{\neg\mathcal{R}})M_2$. The minimal time control problems for interacting timed agents can be solved with DP in their timed multi-agent product.

Example 4.5. In M_1 and M_2 (Figure 4.6), $t(a_1) = t(b_1) = t(b_3) = 1$ and $t(a_2) = t(b_2) = 2$. $\mathcal{R} = \emptyset$. $M_1||^t_{\neg\mathcal{R}}M_2$ is shown in Figure 4.7. □

Definition 4.4. A timed multi-agent system $M_1||^t_{\neg\mathcal{R}}M_2$ is said to be *controllable* if for any states $s = [x,y,RESET]$ and $t = [x',y',RESET]$, there are a sequence of vector events that steer the system from s to t. □

Let $A \subseteq \mathcal{N}^+$ be a set of positive integers, denote the greatest common divisor of elements in A by $gcd(A)$.

Theorem 4.6. *If $\mathcal{R} = \emptyset$ and there are circuits C_1, C_2 in M_1 and M_2 such that $t(C_1) - t(C_2) = gcd(\{|t(a) - t(b)||a \in \Sigma_1, b \in \Sigma_2\})$, then $M_1(||^t_{\neg\mathcal{R}})M_2$ is controllable.*

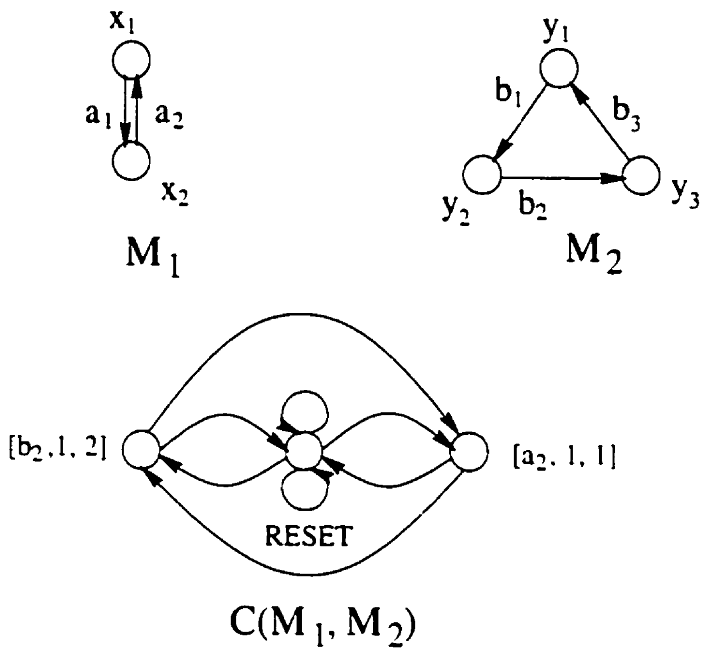

FIGURE 4.6. Two interacting timed agents

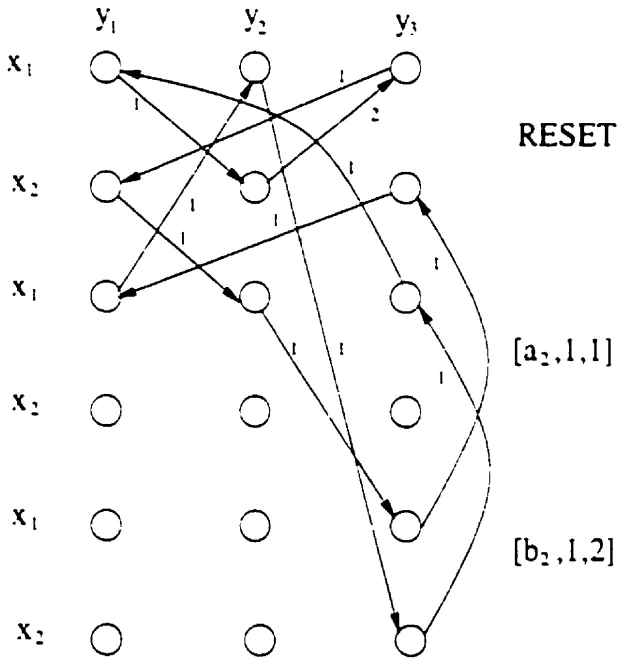

FIGURE 4.7. TMAP(M_1, M_2)

Proof:

Similar to that of Theorem 4.1. □

CHAPTER 5

Hierarchical Network Routing

5.1. Introduction

The solution of routing problems for cost-sensitive telecommunication and transportation networks has become critical for the provision of economic high quality service ([6], [63], [39]). Because (1) the network structure varies when traffic load changes, and (2) multiple users compete for the limited resources (transmission channels, buffers, etc.), the network routing problem has much higher complexity than the conventional single agent optimal trajectory problem addressed in Chapter 3. For a single task, the routing objective is to find a path from its origin s to its destination t with the minimal cost among all admissible paths. The multiple user nature of the networks under consideration suggests that traffic congestion may well occur when the number of flows carried on a link is close to its capacity; evidently this may lead to poor service quality and thus induce higher overall cost.

In networks with heavy traffic, the available size of buffers and the capacity of links constantly vary over the time ([35], [36]). In order to provide good service quality as well as complete tasks, the dynamical routing needs to be flexible, adaptive and functional in real-time ([67], [48]). To this end, hierarchical controllers based on the network status are proposed to solve dynamical routing problems. In this chapter, a generalisation of the HADP algorithm is proposed to cope with the problem of multi-agent time-varying networks. The key notions introduced here are (1) dynamical high level cost pattern, and (2) the state-dependent dynamical routing methods.

A *(buffered) network* is modelled by a directed graph, $G = \{\mathcal{N}, E, \delta, d, B, C\}$, where $\mathcal{N}$ is a set of nodes, E is a set of edges, $\delta : \mathcal{N} \times E \rightarrow \mathcal{N}$ represents the connectivity relations (if $e_i \in E$ is an edge from $n_j \in \mathcal{N}$ to $n_k \in \mathcal{N}$, $\delta(n_j, e_i) = n_k$), $d : E \rightarrow \mathcal{R}^+$ is a cost function mapping each edge to a positive real , $B : \mathcal{N} \rightarrow \mathcal{N}^+$ is a function mapping each node to a positive integer, $C : E \rightarrow \mathcal{N}^+$ is a function mapping an edge to a positive integer.

The cost function d may have different interpretations in various contexts. B can be used to represents the maximal size of the buffer at a given node, and C may represent the maximal capacity of transmission of a given edge, i.e., the maximal number of individual flows that can be handled by the edge at one time.

A *request* $r(s,t)$, $s, t \in \mathcal{N}$, $s \neq t$, is an ordered pair of nodes for which s represents an origin and t represents a destination. An (acyclic) *route assignment* for $r(s,t)$ is a mapping $R : (\gamma_k, r(s,t)) \rightarrow \{E^+ \cup \emptyset\}$, where γ_k is the network state at time instant k; and if $R(\gamma_k, r(s,t)) = e_1 e_2 ... e_n \in E^+$, then $e_i \neq e_j$ when $i \neq j$, and $\delta(s, e_1 e_2 ... e_n) = t$. A routing controller maps an element of $\mathcal{N} \times \mathcal{N}$ to $E^+ \cup \{\emptyset\}$, i.e., when a request $r(s,t)$ arises at time k, the controller either assigns a route $R(\gamma_k, (s,t)) \in E^+$ or rejects the request according to the current state of the network, which is modelled by $R(\gamma_k, r(s,t)) = \emptyset$.

In this chapter, networks are conceptually partitioned into two classes: the link network systems, denoted LNs, in which the capacity of nodes is assumed to be infinity; and the buffered network systems, denoted BNs, in which the capacity of links is assumed to be infinity. Actually, LN and BN can be viewed as duals, or two elements which can be combined later.

5.2. Incremental HADP

5.2.1. Dynamics of Network Topology as a Function of Traffic Loading.

Consider a network $G = \{\mathcal{N}, E, \delta, d, B, C\}$ in the class LN, in which the capacity of all nodes is assumed to be infinite, and hence the loading of traffic that can be handled by the network is constrained by the capacities of links.

Assume that the requests can enter the network at any time instant but only one request arises at a time. Also assume that the request is assigned a route or rejected instaneously. Because the network routing controller under consideration is event (request) driven, only the ordering of requests is relevant. By mapping the ordering of the occurrences of the requests to positive integers, the time instant at which a request arrives can be relabelled by an integer. Hence, we refer to an integer k as the time that a request arises, instead of the value registered by a clock.

For a request $r(s,t)$, if there are admissible paths (the current loading of any link on an admissible path is strictly less than its capacity) from s to the t, then one of these paths is assigned to this request; otherwise, the loading of this request on any path from the origin to the destination will lead to overflow, and thus the request is rejected.

For a route assignment $R(s,t)$, one unit of capacity of all links of $R(s,t)$ is reserved for the exclusive use of $r(s,t)$. In other words, if the request $r(s,t)$ is assigned a route $R(s,t)$ at time instant k ($k > 0$), then for every link e of $R(s,t)$, the available transmission capacity at time instant k of e is $C_k(e) = C_{k-1}(e) - 1$. After a task is completed on $R(s,t)$, one unit of the capacity of all links of $R(s,t)$ is released for the use of other requests. Hence, the capacity of a link is recursively given by,

$$C_k(e) = C_{k-1}(e) - A_k(e) + L_{k-1}(e),$$

$$e \in E, k \in \mathcal{N}^+$$

$$C_0(e) = C(e),$$

where $A_k(e)$ is the number of routes containing link e which are assigned at time k ($A_k(e)$ is 0 or 1, since only one request is processed at any given k), and $L_k(e)$ is the number of tasks which have been completed on link e at time k.

A local dynamical weighting corresponding to local traffic load may be used at a node to give higher preference to the link with relatively light traffic, and thus to avoid traffic congestion. One scheme we isolate is to set the load cost at a link $e \in E$ as

$$d_k(e) = \frac{C_0(e)}{C_k(e)} d(e).$$

If $C_k(e) = C_0(e)$, $d_k(e) = d(e)$. When $C_k(e) = 0$, i.e., $d_k(e) = \infty$, the link has no spare transmission capacity. In this case, an alternative of this link will be chosen if it exists. In the computation of an additive minimum cost trajectory, a node in an optimal path may of course be such that a link is chosen which does not have minimum cost at that node because it contributes to a globally optimal trajectory. A frequently used form of the traffic-dependent transmission costs in the probabilistic setting is

$$d_k(e) = \frac{E[C_0(e).T]}{E[C_0(e) - C_k(e).T]},$$

where $E[C_0(e) - C_k(e).T]$ is the Erlang-B formula for T Erlangs offered to $C_0(e) - C_k(e)$ channels ([**48**]).

In a dynamical network, the capacity information of the links is updated at each instant k immediately, when a route is assigned for a request or a task is step-wise (partially) completed. The dynamical weights are also updated. We shall denote the topology of a network $\mathcal{N}$ at time instant k by $G_k = \{\mathcal{N}, E_k, \delta, C_k, d_k\}$, where $E_k \triangleq \{e \in E | C_k(e) > 0\}$.

5.2.2. Throughput-Independent ST-IBC Partition Machines. For an ordered pair of distinct nodes $(x, y) \in \mathcal{N} \times \mathcal{N}$, let a subset $E_c(x, y) \subseteq E$ be such that there does not exist $u \in (E - E_c(x, y))^*$ such that $\delta(x, u) = y$. $E_c(x, y)$ is called a *cut* with respect to (x, y) ([**23**]). In other words, a cut with respect to (x, y) is a subset of edges that separate x from y. Denote the collection of all cuts for $(x, y) \in \mathcal{N} \times \mathcal{N}$, $x \neq y$, by $cut(x, y)$.

For two subsets of nodes, $X, Y \in 2^{\mathcal{N}}$, $E_c(X, Y)$ is a subset of edges such that for all $x \in X$ and for all $y \in Y$, $x \neq y$, there does not exist $u \in (E - E_c(x, y))^*$

$\delta(x, u) = y$. Denote the collection of all such $E_c(X, Y)$ by $cut(X, Y)$, $X, Y \in 2^N$.

Define the *minimal cut* of $x \in X$ and $y \in X$ with respect to a subset of nodes $X \subseteq N$ as follows, $mincut_{X_i}(x, y) \triangleq \min\{\sum_{e \in X \times X, e \in E_c(x,y)} C(e) | E_c(x, y) \in cut(x, y)\}$. For two subsets $X, Y \subseteq Z \subseteq N$, $mincut_Z(X, Y) \triangleq \min \{\sum_{e \in Z \times Z, e \in E_c(X,Y)} C(e) | E_c(X, Y) \in cut(X, Y)\}$. It is a basic result in graph theory that the maximal flow from x to y with respect to X is equal to $mincut_X(x, y)$ ([**23**]), and this also holds for any subsets $X, Y \subseteq Z$.

For a sequence of links $p = e_1 e_2 ... e_n \in E^\bullet$, define the *path-wise capacity* $PC(p) = \min\{C(e_i) | 1 \leq i \leq n\}$.

Suppose in a network G, there are two subsets of nodes $S \subseteq N$ and $T \subseteq N$, such that for all request $r(s, t)$, $s \in S$ and $t \in T$. Let π be a partition of N, the set of nodes of a network G. Recall the definitions of in-sets and out-sets ([**15**]). The property of throughput-independent ST-IBC for partition blocks as follows:

Definition 5.1. (**(TI-ST-IBC)** A block $X_i \in \pi$ is *throughput-independent ST-IBC* (TI-ST-IBC) if
(1) $\forall x \in I(X_i)$, $\exists y \in O(X_i)$ such that there is $u \in E^\bullet$, $\delta(x, u) = y$ and $\forall v < u$, $\delta(x, v) \in X_i$;
(2) $\forall x, y \in O(X_i)$, $x \neq y$, $u_{X_i}(x, y) = \{u \in E^\bullet$ such that $\delta(x, u) = y$ and $\forall v < u$, $\delta(x, v) \in X_i\} \neq \emptyset$;
(3) $\forall x, y \in O(X_i)$, there is $v \in u_{X_i}(x, y)$ such that $PC(v) \geq mincut_{X_i}(I(X_i), O(X_i))$.

□

The first two conditions in Definition 5.1 are those for a block to be ST-IBC ([**15**]); the third condition ensures the ST-IBC property of a block to be preserved, regardless of the varying traffic load in the sense that there exists a path from $x \in O(X_i)$ to $y \in O(X_i)$ whose capacity exceeds the maximal flow from $I(X_i)$ to $O(X_i)$. This property also implies that $I(X_i) \cap O(X_i) = \emptyset$.

Definition 5.2. ($I_k(X_i)$) For $X_i \in \pi$, the *in-set of X_i at time instant k, $I_k(X_i)$* is a subset of $I(X_i)$ such that for all $x \in I_k(X_i)$,
(1) $\exists\ y \notin X_i,\ e \in E_k$ such that $\delta(y, e) = x$ and,
(2) $\exists\ z \in O(X_i)$ such that $\exists\ u \in E_k^*,\ \delta(x, u) = z$ and $\forall\ v < u,\ \delta(x, v) \in X_i$. □

With the loading of traffic to X_i, some nodes in $I(X_i)$ may not be accessible to any outset node with respect to X_i. These nodes do not appear in $I_k(X_i)$. Define $I_{i,k}(X_j) = \{x \in I_k(X_j) \mid \exists y \in X_i\ \exists e \in E_k . \delta(y, e) = x\}$. As stated earlier, any route assignment is acyclic, i.e., for all $e \in E$, e appears no more than once in $R(s, t)$.

Lemma 5.1. *If X_i is TI-ST-IBC and $I_k(X_i) \neq \emptyset$, then for all $x \in I_k(X_i)$ and for all $z \in O(X_i)$, there is $p \in E_k^*$, such that $\delta(x, p) = z$ and for all $q < p$, $\delta(x, q) \in X_i$.*

Proof:

Suppose X_i is TI-ST-IBC. By the definition of $I_k(X_i)$, for an arbitrary $x \in I_k(X_i)$, there is $y \in O(X_i)$, and $u \in E_k^*$, such that $\delta(x, u) = z$ and for all $v < u$, $\delta(x, v) \in X_i$.

Let $z \neq y$ be an arbitrary node in $O(X_i)$. According to Definition 5.1, there is $w \in E^*$, such that $\delta(y, w) = z$ and for all $r < w$, $\delta(y, w) \in X_i$ and $PC(w) > mincut_{X_i}(I(X_i), O(X_i))$.

A path from any $s \in S$ to any $t \in T$ including a link more than twice must involve a circuit, and thus can clearly be replaced by another acyclic route assignment. That is to say, for any $s, t \in X$, if there is a path from s to t, then there exists an acyclic $R(s, t) \in E^*$. So, for any link $e \in E$, e is on $R(s, t)$ at most once.

By assumption, for all requests $r(s, t)$, $s \in S$, $t \in T$, and it is always the case that $R(s, t)$ goes through X_i from $I(X_i)$ to $O(X_i)$. Because $u \in E_k^*$, such that $\delta(x, u) = z$, at time instant k, there are strictly less than $mincut_{X_i}(I(X_i), O(X_i)) - 1$ route assignments going through X_i from $I(X_i)$ to $O(X_i)$. Hence, for any edge e in $X_i \times X_i$, at time k, e carries at most $mincut_{X_i}(I(X_i), O(X_i)) - 1$ individual flows.

Therefore, for any edge a in w, $a \in E_k$. That is to say $p = uw \in E_k^*$, $\delta(x, p) = z$ and for all $q < p$, $\delta(x, q) \in X_i$. □

A partition of X, π, is said to be an *throughput-independent ST-IBC partition* if all its blocks are throughput-independent ST-IBC. Let a throughput-independent ST-IBC partition machine of G be denoted by $G^h = \{\pi, E^h, \delta^h\}$, where $E^h = \{U_i^j | \langle X_i, X_j \rangle$ is ST-DC (see Chapter 2), $1 \leq i, j \leq |\pi|\}$, and $\delta^h(X_i, U_i^j) = X_j$.

5.2.3. Incremental HADP. Denote a two level hierarchy consisting of G^h and G by $\{G, G^h\}$. In this section we present an extension of HADP to $\{G, G^h\}$ which treats the multi-agent case under consideration.

Define $d_{X_i, X_j, k}(x, y) = \min\{d(u) | u \in E_k^*$ s.t. $\delta(x, u) = y$ and $\forall\, v < u, \delta(x, v) \in X_i\}$. The high level cost function $D_k^{+/-}$ is defined below.

Definition 5.3. $(D_k^{+/-}(X_i, X_j))$
$D_k^{+/-}(X_i, X_j) = \max_x \min_y \{\, d_{X_i, X_j, k}(x, y) | \, x \in I_k(X_i), y \in I_{i,k}(X_j)\}$.

□

Define $E_k^h = \{U_i^j | D_t^{+/-}(X_i, X_j) < \infty\}$, $G_k^h = \{\pi, E_k^h, \delta^h, D_k^{+/-}\}$, $k \in N^+$.

To apply HADP, we first set up a hierarchy consisting of a low level network G and its TI-ST-IBC partition machine G^h. Then an initial route assignment is made by HADP$(D^{+/-})$ based on $\{G, G^h\}$ (see Chapter 3).

Subsequently, at every time instant a request is received by the network, an observation is made on the traffic loading (links reserved and released) and the link capacities of the entire system are updated. For a request $r(s, t)$, arriving at time k, the high level costs $\{D_k^{+/-}\}$ are recalculated (similar to the treatment in Chapter 3). This latter step is formalised in the algorithm below which hence generalises G_k^h.

Algorithm 1 of Incremental HADP (IHADP) for $r(s,t)$ ***at*** $k > 0$

(1) For each $e \in E$, set $C_k(e) = C_{k-1}(e) + L_{k-1}(e)$, where $L_k(e)$ is the number of tasks which have been completed on link e at time k.

(2) Calculate $d_k(e) = (C_0(e)/C_k(e))d(e)$ for each $e \in E$.

(3) Calculate $I_k(X_i)$ for all $X_i \in \pi$.

(4) Calculate $D_k^{-/-}(X_i, X_j)$ for all DC pairs $\langle X_i, X_j \rangle$, $1 \leq i, j \leq |\pi|$. □

For request $r(s,t)$, let X^s and X^t be the blocks containing s and t respectively. If $X^s = X^t$, seek an optimal path from s to t with respect to X^s. Otherwise, we first seek a high level optimal solution with respect to $D_k^{-/-}$ from X^s to X^t in G_k^h. Then, a low level solution is sought by the process described in the algorithm formulated below.

Algorithm 2 of Incremental HADP (IHADP) for $r(s,t)$ ***at*** $k > 0$

(1) If $X^s = X^t$, seek an optimal path from s to t with respect to X^s, stop; else, set $I_k(X^s) = \{s\}$, $I_k(X^t) = \{t\}$.

(2) Calculate $D_k^{-/-}(X_i, X_j)$ for all DC pairs $\langle X^s, X_j \rangle$ and $\langle X_i, X^t \rangle$, $1 \leq i, j \leq |\pi|$.

(3) Set $E_k^h = \{U_i^j | D_k^{-/-}(X_i, X_j) < \infty\}$.

(4) Seek an optimal path from X^s to X^t in G_k^h. Denote this path by $Y_1 \rightarrow Y_2 \rightarrow \ldots \rightarrow Y_n$. If no path from X^s to X^t exists, stop.

(5) Let $x_1 = s$ and $x_n = t$.

(6) Start from $i = 1$, if $i < n - 1$, seek $u_i \in E_k^*$ such that $\delta(x_i, u_i) \in I_i(Y_{i-1})$ and $d(u) = \min_{y \in I_i(Y_{i-1})} d_{Y_i Y_{i+1}, k}\,(x_i, y)$. Set $x_{i-1} = \delta(x_i, u_i)$. Set $i = i + 1$.

(7) If $i = n - 1$, seek $u_i \in E_k^*$ such that $\delta(x_i, u_i) = x_n$. $u_1 u_2 \ldots u_{n-1}$ is a low level solution for $r(s,t)$.

(8) Set $C_k(e) = C_{k-1}(e) - 1$ for all $e \in E_k$ on the resulting low level path. □

Algorithm IHADP consists of algorithms 1 and 2 at every time instant $k \geq 1$.

Compared with the single task HADP, the extra computational time of the time-varying HADP lies in steps 1 and 2 in the above algorithm. Clearly, these two steps may be carried out locally for each DC block pairs and at each time instant $k \geq 1$..

If these steps can be performed in parallel, a greater speedup of HADP is expected.

The following basic theorem guarantees that if no high level solution exists in step 4, no low level solution exists in G_k for $r(s,t)$.

Theorem 5.1. *Let π be a TI-ST-IBC partition of X and $X^s \neq X^t$, then for any $s \in I(X^s)$ and $t \in I(X^t)$, there is $u \in E_k^*$ such that $\delta(s,u) = t$ if and only if there is $U \in (E_t^h)^*$ such that $\delta^h(X^s, U) = X^t$.*

Proof:

Suppose there is a high level path from X^s to X^t with finite cost at time instant k. Denote this path by $Y_1 \rightarrow Y_2 \rightarrow \ldots \rightarrow Y_n$, $Y_1 = X^s$ and $Y_n = X^t$. $\langle Y_i, Y_{i-1}\rangle$ is ST-DC and $D_k^{+/-}(Y_i, Y_{i-1}) < \infty$, $1 \leq i < n$. That is to say, in any Y_i, for all $x \in I_k(Y_i)$, there is $x' \in I_{i,k}(Y_{i-1})$, such that $d_{Y_i Y_{i-1},t}(x, x') \leq D_k^{+/-}(Y_i, Y_{i-1}) < \infty$ (by Definition 5.3), $I_{i,k}(Y_{i-1}) \neq \emptyset$.

For any node $z \in I_{i,k}(Y_{i-1})$, by Definition 5.2, there is $y \in O(Y_i)$ and $e \in E_k$ ($d_k(e) < \infty$) such that $\delta(y, e) = z$. By Lemma 5.1, since Y_i is TI-ST-IBC, for all $x \in I(Y_i)$, $y \in O(X_i)$, there is $u \in E_k^*$ such that $\delta(x, u) = y$, and moreover $\delta(x, ue) = z$ (see Figure 5.1). Because $d_{Y_i,k}(x, y) \leq d(u) < \infty$, $d_{Y_i Y_{i-1},k}(x, z) \leq d(ue) < \infty$ for arbitrary $x \in I(Y_i)$ and $z \in I_{i,k}(Y_{i-1})$.

Let $x_1 = s$ and $x_n = t$. Starting from Y_1, when $i < n-1$, seek $u_i \in E_k^*$ such that $\delta(x_i, u_i) \in I_i(Y_{i-1})$ and $d(u) = \min_{y \in I_i(Y_{i-1})} d_{Y_i Y_{i+1},k}(x_i, y)$, set $x_{i-1} = \delta(x_i, u_i)$. When $i = n-1$, seek $u_i \in E_k^*$ such that $\delta(x_i, u_i) = x_n$. It is clear that $u_1 u_2 \ldots u_{n-1}$ is a low level path from s to t and that $d(u_1 u_2 \ldots u_{n-1}) = \sum_{i=1}^{n-1} d_t(u_i) < \infty$.

Conversely, suppose at time k, there is a low level path $p(s,t)$ from s to t with finite cost. Denote the blocks containing this low level path by $Y_1, Y_2, \ldots, Y_n$ in order, $Y_1 = X^s$ and $Y_n = X^t$. Clearly, $I_k(X_i) \neq \emptyset$ and there are $x_i \in O(Y_i)$, $y_{i+1} \in I_i(Y_{i+1})$ and $e_i \in E^k$, $1 \leq i < n$, such that $\delta(x_i, e_i) = y_{i+1}$. Because π is TI-ST-IBC, it is

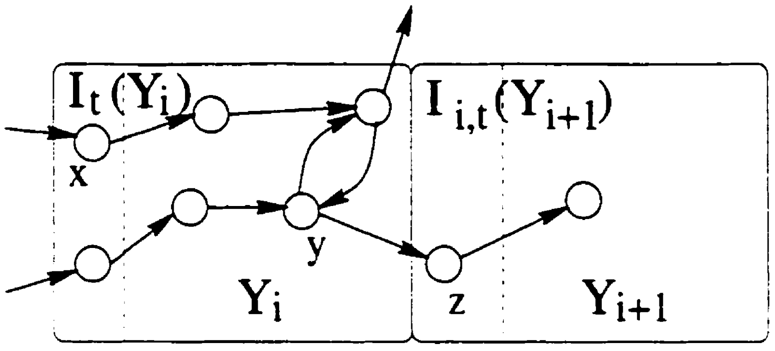

FIGURE 5.1. $d_{Y_iY_j,t}(x,z)$ is finite

straightforward to see that any $\langle X_i, X_j\rangle$ is ST-DC if there is a low level path directly from X_i to X_j. Thus, $\langle Y_i, Y_{i+1}\rangle$ is ST-DC, $1 \leq i \leq n-1$.

For any $x \in I_k(Y_i)$, by Lemma 5.1, there is $u \in E_k^*$ such that $\delta(x, u) = x_i \in O(Y_i)$, and moreover $\delta(x, ue_i) = y_{i+1} \in I_i(Y_{i+1})$. Thus, by definition, $D^{-,-}(Y_i, Y_{i+1}) < \infty$. Therefore, the cost of $L^* = L_1^{*2}L_2^{*3}...L_{n-1}^{*n}$ is finite and $\delta^h(X^s, L^*) = X^t$. □

It is worth remarking that if there is high level solution for X^s and X^t in G_k^h when $X^s \neq X^t$ or there is no solution for s and t with respect to X^s when $X^s = X^t$, $r(s,t)$ is rejected at time instant k. This corresponds to the case that an overload is taking place in the network.

5.3. Multiple Objective Network Routing

5.3.1. Network States of the Link Network Systems. Consider a network $G = \{X, E, \delta, d, C\}$ in the class of LN. Suppose n requests with origin-destination pairs, $r(s_i, t_i)$, $1 \leq i \leq n$, arrive at G simultaneously. Assume that for each i, $1 \leq i \leq n$, there is $R(s_i, t_i) \in E^*$ satisfying

$$\delta(s_i, R(s_i, t_i)) = t_i,$$

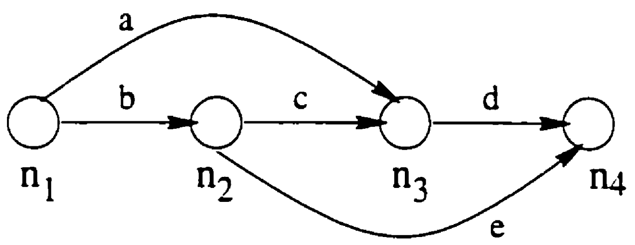

FIGURE 5.2. A simple network

$$1 \le i \le n.$$

and satisfying,

$$\sum_{i=1}^{n} \beta(e_j, R(s_i, t_i)) \le C(e_j), \tag{5.5.1}$$

$$1 \le j \le |E|.$$

where $\beta(e_j, r(s_i, t_i))$ is a unit cost set-membership (or characteristic) function for an edge e_j and a path $R(s_i, t_i))$ defined by,

$$\beta(e_j, R(s_i, t_i)) = \begin{cases} 1 & \text{if } e_j \text{ is on } R(s_i, t_i) \\ 0 & \text{otherwise} \end{cases}.$$

Then we say that *a valid routing* $\{R(s_i, t_i), 1 \le i \le n\}$ *for these n requests* $\{r(s_i, t_i), 1 \le i \le n\}$ *exists.*

When a valid routing exists for a set of requests $\{r(s_i, t_i), 1 \le i \le n\}$, the optimal control objective is to minimise the overall cost

$$\sum_{i=1}^{n} d(R(s_i, t_i)).$$

For convenience, from now on, only the case $n = 2$ will be discussed, but all conclusions hereafter can be extended to $n > 2$ with a little modification of the argument in each case.

Example 5.1. A network is shown in Figure 5.2. All links have a unit capacity. The costs $d(a) = d(e) = 3$, $d(b) = d(c) = d(d) = 1$. Two requests are $r(n_1, n_4)$ and $r(n_1, n_4)$. The optimal route for both requests is $R^0(n_1, n_4) = bcd$. But if this route is assigned to any of the requests, the other is blocked. □

The notion of *(vector) network state* can be used to represent the state of a network. A vector network state $\gamma = [c(e_1), c(e_2), ..., c(e_{|E|})]$. $0 \le c(e_i) \le C(e_i)$, $1 \le i \le |E|$, is a vector with each element denoting the available capacity of an edge.

The dynamics of a routing process may be expressed as a finite state machine $RP = \{\Gamma, E \cup \{\epsilon\}, \xi\}$, where $\Gamma = \{0, 1, ..., C(e_1)\} \times \{0, 1, ..., C(e_2)\} \times ... \times \{0, 1, ..., C(e_{|E|})\}$ is the set of all network states. ϵ is a distinguished event corresponding to no transition taking place. With this notation, the state transition function

$$\xi(\begin{bmatrix} c(e_1) \\ c(e_2) \\ \vdots \\ c(e_i) \\ \vdots \\ c(e_{|E|}) \end{bmatrix}, e_i) = \begin{cases} \begin{bmatrix} c(e_1) \\ c(e_2) \\ \vdots \\ c(e_i) - 1 \\ \vdots \\ c(e_{|E|}) \end{bmatrix}, & \text{if } c(e_i) \ge 1 \\ \text{undefined.} & \text{if } c(e_i) = 0 \end{cases}$$

if $e_i \neq \epsilon$, otherwise, $\xi(ns, \epsilon) = ns$.

A *routing controller* for a network G is a finite state machine $RC(G) = \{N \times N \times \Gamma, (E \cup \{\epsilon\}) \times (E \cup \{\epsilon\}), \rho, d'\}$, where

$$\rho(\begin{bmatrix} x \\ y \\ \gamma \end{bmatrix}, \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}) = \begin{cases} \begin{bmatrix} \delta(x, e_1) \\ \delta(y, e_2) \\ \xi(ns, e_1 e_2) \end{bmatrix}, & \delta(x, e_1)! \& \delta(y, e_2)! \& \xi(\gamma, e_1 e_2)! \\ \text{undefined.} & \text{otherwise} \end{cases}.$$

and the cost function $d'([x, y, \gamma], [e_1, e_2]) = d(x, e_1) + d(y, e_2)$ whenever $\rho([x, y, \gamma], [e_1, e_2])!$ $(d(x, \epsilon) \triangleq 0$ for all $x \in N)$.

If $[Trj(s_1, u_1), Trj(s_2, u_2)]$, $u_1, u_2 \in (E \cup \{\epsilon\})^*$, is a (vector) path from $[s_1, s_2, [C(e_1), C(e_2), ..., C(e_{|E|})]]$ to $[t_1, t_2, \gamma]$ in $RC(G)$, where $\gamma \in \Gamma$ may be any network state, then $\delta(s_1, u_1) = t_1$ and $\delta(s_2, u_2) = t_2$. Because ρ gives all admissible routing processes, u_1 and u_2 satisfy the capacity constraints (5.5.1). The optimal routing problem is thus converted to a problem of seeking a shortest path from

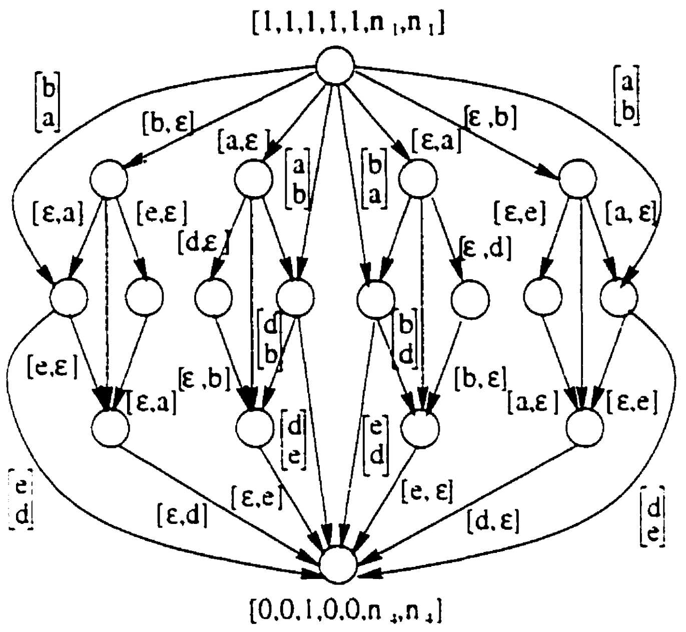

FIGURE 5.3. A routing controller for Example 5.1

$[s_1, s_2, [C(e_1), C(e_2), \ldots, C(e_{|E|})]]$ to a set of states $\{[t_1, t_2, \gamma] | ns \in \Gamma\}$ in $RC(G)$.

A part of $RC(G)$ for the network G in Example 5.1 is shown in Figure 5.3. We note that all legal non-blocking routings of the network are represented by $RC(G)$. An optimal control for $[n_1, n_1]$ and $[n_4, n_4]$ is $[ad, be]$ with the overall cost of 8.

5.3.2. Throughput IBC Partition Machines. Since $RC(G)$ has a huge state space if $|N|$ and $|E|$ of G are large, we are interested in studying the application of hierarchical control (specifically, HADP) for finding the optimal routing for large G, in order to reduce the computational complexity of DP procedure.

Assume that the origins of all requests are in $S \subseteq N$, a subset of nodes in G, and the destinations of all requests are in $T \subseteq N$. We may directly partition $N \times N \times \Gamma$ into ST-IBC partitions to obtain a high level model for $RC(G)$ for which HADP is applicable. Alternatively, hierarchical routing can be achieved by partitioning N, which is much smaller than $N \times N \times \Gamma$ (into some structures to be discussed) to form a high level network model G^h. Then we may construct a routing machine $RC^h(G^h)$ for this high level network. $RC^h(G^h)$ and $RC(G)$ form a hierarchy $\{RC(G), RC^h(G^h)\}$ to apply HADP. Let π be an arbitrary partition of N. We now present a formulation of $1-ST-DC$ and $2^+-ST-DC$ relations over partition blocks.

Definition 5.4. (1-ST-DC) An ordered pair of blocks $X_i \in \pi$ and $X_j \in \pi$ are *1-ST-DC* if:
(1) $\forall x \in I(X_i)$, there is $y \in I_i(X_j)$ such that there is $u \in E^*$, $\delta(x,u) = y$ and for all $v < u$, $\delta(x,v) \in X_i$;
(2) $mincut_{X_i X_j}(I(X_i), I_i(X_j)) = 1$. □

Clearly, if $\langle X_i, X_j\rangle$ is 1-ST-DC, $\langle X_i, X_j\rangle$ is ST-DC. Furthermore, $\langle X_i, X_j\rangle$ is 1-ST-DC implies that then only one flow entering X_i at any node in $I(X_i)$ can go through X_i to X_j without causing an overflow.

If n requests $r(s_i,t_i)$, $i \leq i \leq n$, are under consideration, for any link $e \in E$ such that $C(e) \geq n$, an acyclic route assignment $\{R(s_i,t_i), 1 \leq i \leq n\}$ will not lead to an overflow on e. Only the links with capacity strictly less than the number of requests are critical to avoiding overflow. Therefore, in case only 2 requests are under consideration at the same time, the links with capacity greater than 2 can be viewed as being in the same class, and we shall use 2^+ to represent capacities which are greater than or equal to 2.

Definition 5.5. (2^+-ST-DC) An ordered pair of blocks $X_i \in \pi$ and $X_j \in \pi$ are *2^+-ST-DC* if:
(1) $\forall x \in I(X_i)$, there is $y \in I_i(X_j)$ such that there is $u \in E^*$, $\delta(x,u) = y$ and for all $v < u$, $\delta(x,v) \in X_i$;
(2) $\forall [x,y] \in I(X_i) \times I(X_i)$, there is $[x',y'] \in I_i(X_j) \times I_i(X_j)$, such that there is $[u_1,u_2] \in E^* \times E^*$, $\delta(x,u_1) = x', \delta(y,u_2) = y'$, for all $v_1 < u_1, v_2 < u_2$, $\delta(x,v_1) \in X_i, \delta(y,v_2) \in X_i$, and for all $e \in E$, $\beta(e,u_1) + \beta(e,u_2) \leq C(e)$. □

The first clause in the above definition states the standard condition for a pair of blocks to be ST-DC. The second clause indicates that if $\langle X_i, X_j\rangle$ is 2^+-ST-DC, then two individual flows entering X_i at any two nodes in $I(X_i)$ can go through X_i to X_j

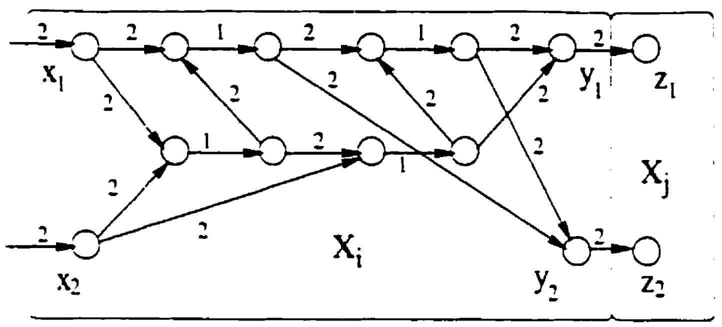

FIGURE 5.4. X_i is not 2^+-throughput IBC

without causing an overflow.

Definition 5.6. (1-throughput IBC) A block $X_i \in \pi$ is *1-throughput IBC* if:
(1) $\forall x \in I(X_i)$ and $\forall y \in O(X_i)$, there is $u \in E^*$ such that $\delta(x, u) = y$ and for all $v < u$, $\delta(x, v) \in X_i$;
(2) $mincut_{X_i}(I(X_i), O(X_i)) = 1$. □

In a 1-throughput IBC block $X_i \in \pi$, the elements in $O(X_i)$ are not required to be mutually accessible because only one flow from $I(X_i)$ to $O(X_i)$ is possible.

Definition 5.7. (2^+-throughput IBC) A block $X_i \in \pi$ is *2^+-throughput IBC* if:
(1) $\forall x \in I(X_i)$ and $\forall y \in O(X_i)$, there is $u \in E^*$ such that $\delta(x, u) = y$ and for all $v < u$, $\delta(x, v) \in X_i$;
(2) $\forall [x, y]^T \in I(X_i) \times I(X_i)$, and $\forall [x', y']^T \in O(X_i) \times O(X_i)$, there is $[u_1, u_2]^T \in E^* \times E^*$ such that $\delta(x, u_1) = x', \delta(y, u_2) = y'$, $\forall u_1' < u_1, u_2' < u_2$, $\delta(x, u_1'), \delta(y, u_2') \in X_i$, and for all $e \in E$, $\beta(e, u_1) + \beta(e, u_2) \leq C(e)$. □

Example 5.2. If $|I(X_i)| = 1$, i.e., $I(X_i)$ is a singleton, the condition $mincut(x, y) \geq 2$, for all $x \in I(X_i), y \in O(X_i)$, implies that X_i is 2^+-throughput IBC. But when X_i has more than one in-set node, this is not a sufficient condition for X_i to be 2^+-throughput IBC. In Figure 5.4, the capacities of links are as labelled. Although $mincut(x_i, y_j) = 2$ for all $i, j = 1, 2$, no admissible control with respect to X_i can drive the system from $[x_1, x_2]$ to $[y_1, y_2]$. □

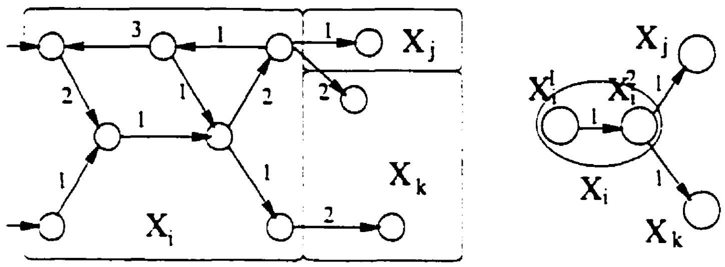

FIGURE 5.5. An 1-throughput IBC block and its abstraction

A partition π is *throughput IBC* if all its blocks are either 1-throughput IBC or 2^--throughput IBC. In order to avoid ambiguity, an internal "valve" is placed in a 1-throughput IBC to switch a flow to one of its succeeding blocks (see Figure 5.5). Hence, two nodes, X_i^1, X_i^2, are used to abstract a 1-throughput IBC block X_i. In contrast, for a 2^--throughput IBC block, it can be abstracted into a node when the number of requests is 2.

Analogously, if $n > 2$, blocks can be classified into 1-throughput IBC, 2-throughput IBC, ..., n^--throughput IBC, in accordance with their internal capacity to transmit flows. And, similarly, an internal valve $X_i^1 \to X_i^2$ with capacity k is used to represent the transmission capacity of block X_i, for a k-throughput IBC block X_i, $k < n$.

Lemma 5.2. *Let $X_i \in \pi$ be a 1-throughput IBC block, for any $X_j \in \pi$, if there are $x \in X_i$, $y \in X_j$, $e \in E$ such that $\delta(x, e) = y$, then $\langle X_i, X_j \rangle$ is 1-ST-DC.*

Proof:

Suppose $X_i \in \pi$ is 1-throughput IBC. If $x \in O(X_i)$, $y \in I_i(X_j)$, and $e \in E$ such that $\delta(x, e) = y$, by Definition 5.6(1), for all $z \in I(X_i)$, there is $u \in E^*$ such that $\delta(z, u) = x \in O(X_i)$ and for all $v < u$, $\delta(z, v) \in X_i$. Thus $\delta(z, ue) = y \in I_i(X_j)$ and for all $w < ue, \delta(z, w) \in X_i$. This also implies $mincut_{X_i X_j}(I(X_i), I_i(X_j)) \geq 1$.

By Definition 5.6(2), $mincut_{X_i}(I(X_i), O(X_i)) = 1$. Thus, $mincut_{X_iX_j}(I(X_i), I_i(X_j))$ $= \min\{ mincut_{X_i}(I(X_i), O(X_i)), mincut_{X_iX_j} (O(X_i), I_i(X_j))\} \leq mincut_{X_i} (I(X_i), O(X_i)) = 1$. So, $mincut_{X_iX_j} (I(X_i), I_i(X_j)) = 1$.

Therefore, $\langle X_i, X_j\rangle$ is 1-ST-DC. □

Lemma 5.3. *Let $X_i \in \pi$ be a 2^+-throughput IBC block. For any $X_j \in \pi$, if there are $x \in X_i$, $y \in X_j$, $e \in E$ such that $\delta(x, e) = y$, then $\langle X_i, X_j\rangle$ is either 1-ST-DC or 2^+-ST-DC.*

Proof:

Assume X_i is 2^+-throughput IBC. Let $x \in O(X_i)$, $y \in I_i(X_j)$, and $e \in E$ be such that $\delta(x, e) = y$. By a similar argument to that of the proof of Lemma 5.2, we know that the first condition for Definition 5.7 is satisfied and $mincut_{X_iX_j}(O(X_i), I_i(X_j)) \geq C(e) \geq 1$.

Evidently, it is the case that either $mincut_{X_iX_j}(O(X_i), I_i(X_j)) \geq 2$ or $mincut_{X_iX_j} (O(X_i), I_i(X_j)) = 1$. If $mincut_{X_iX_j} (O(X_i), I_i(X_j)) \geq 2$, then there are two cases:

(1) If there is $x \in O(X_i), e_1 \in E$ such that $\delta(x, e_1) = y \in I_i(X_j)$ and $C(e_1) \geq 2$. For any two nodes $x_1, x_2 \in I(X_i)$ (not necessarily distinct), by Definition 5.7, there are $u_1, u_2 \in E^*$ such that $\delta(x_1, u_1) = \delta(x_2, u_2) = x$, and for all $v_1 < u_1, v_2 < u_2$, $\delta(x_1, v_1), \delta(x_2, v_2) \in X_1$, for all $a \in E$, $\beta(a, u_1) + \beta(a, u_2) \leq C(a)$. Therefore $[u_1e_1, u_2e_1]$ drives the system from $[x_1, x_2]$ to $[y, y]$ via $[x, x]$ through X_i. It is clear $\beta(e_1, u_1e_1) + \beta(e_1, u_2e_1) = 2 \leq C(e_1)$. Hence, $\langle X_i, X_j\rangle$ is 2^+-ST-DC.

(2) If there are $x, x' \in O(X_i)$ (not necessarily distinct), $e_1, e_2 \in E$, $e_1 \neq e_2$, such that $\delta(x, e_1) = y \in I_i(X_j)$ and $\delta(x', e_2) = y' \in I_i(X_j)$. It follows from Definition 5.7 that for all $x_1, x_2 \in I(X_i)$, there are $u_1, u_2 \in E^*$ such that $\delta(x_1, u_1) = x$ and $\delta(x_2, u_2) = x'$, and for all $v_1 < u_1, v_2 < u_2$, $\delta(x_1, v_1), \delta(x_2, v_2) \in X_1$, and for all $a \in E$, $\beta(a, u_1) + \beta(a, u_2) \leq C(a)$. Furthermore, $[u_1e_1, u_2e_2]$ drives the system from $[x_1, x_2]$ to $[y, y']$ via $[x, x']$ through $X_i \times X_1$. It is clear $\beta(e_1, u_1e_1) = 1 \leq C(e_1)$ and

$\beta(e_2, u_2e_2) = 1 \le C(e_2)$. Hence, $\langle X_i, X_j\rangle$ is 2^--ST-DC.

When $mincut_{X_i X_j}(O(X_i), I_i(X_j)) = 1$, it implies that $O_j(X_i) = \{x\}$, $I_i(X_j) = \{y\}$ and $C(e) = 1$, it is clear that $\langle X_i, X_j\rangle$ is 1-ST-DC. □

Let π be a throughput IBC partition of X. The throughput IBC partition machine of G based on π is denoted by $G^h = \{\pi', E^h, \delta^h, C^h, D^h\}$, where
$\pi' = \{X_i \in \pi \mid X_i$ is 2^--throughput IBC$\} \cup \{X_i^1, X_i^2 \mid X_i$ is 1-throughput IBC$\}$;
$E^h = \{L_i^j \mid \langle X_i, X_j\rangle$ is 2^--ST-DC$\} \cup \{L_i^j \mid \langle X_i, X_j\rangle$ is 1-ST-DC $\} \cup \{L_i \mid X_i \in \pi$ is 1-throughput IBC$\}$.

High level transitions

The high level connectivity (transition) function of G^h is defined as follows: for any $X_i, X_j \in \pi$,
$\delta^h(X_i^1, L_i) = X_i^2$ if X_i is 1-throughput IBC;
$\delta^h(X_i, L_i^j) = X_j$, if $\langle X_i, X_j\rangle$ is 1-ST-DC or 2^--ST-DC, and X_i, X_j are 2^--throughput IBC;
$\delta^h(X_i^2, L_i^j) = X_j^1$, if $\langle X_i, X_j\rangle$ is 1-ST-DC and X_i, X_j are 1-throughput IBC;
$\delta^h(X_i^2, L_i^j) = X_j$, if $\langle X_i, X_j\rangle$ is 1-ST-DC and X_i is 1-throughput IBC, X_j is 2^--throughput IBC;
$\delta^h(X_i, L_i^j) = X_j^1$, if $\langle X_i, X_j\rangle$ is 1-ST-DC or 2^--ST-DC, X_i is 2^--throughput IBC, and X_i is 1-throughput IBC.

High level capacities

The high level capacity of a 1-ST-DC connection is 1, the capacity of a 2^--ST-DC connection is 2, i.e., for a 1-ST-IBC pair $\langle X_i, X_j\rangle$, $C^h(L_i^j) = 1$, for a 2^--ST-IBC pair $\langle X_i, X_j\rangle$, $C^h(L_i^j) = 2$. As stated earlier, there are two nodes representing a 1-throughput IBC block X_i: X_i^1 and X_i^2, which are connected with a link L_i with capacity of 1, i.e., $C^h(L_i) = 1$ for all 1-throughput IBC X_i.

High level costs

To calculate the high level costs, some modification of the in-sets is needed (see Section 3.4). If $s \in X_i$ and X_i is a 1-throughput IBC block, $X^s = X_i^1$. If $t \in X_i$ and

X_t is a 1 throughput IBC block, $X^t = X_t^2$. Let $[s_1, s_2] \in I(X^{s_1}) \times I(X^{s_2})$. In order to calculate high level costs, set $I(X^{s_1}) = \{s_1\}, I(X^{s_2}) = \{s_2\}$ whenever $X^{s_1} \neq X^{s_2}$; otherwise, $I(X^{s_1}) = \{s_1, s_2\}$; set $I(X^{t_1}) = \{t_1\}, I(X^{t_2}) = \{t_2\}$ whenever $X^{t_1} \neq X^{t_2}$; otherwise, $I(X^{t_1}) = \{t_1, t_2\}$; if $X^{s_1} = X^{t_2}$, $I(X^{s_1}) = I(X^{s_1}) \cup I(X^{t_2})$; if $X^{t_1} = X^{s_2}$, $I(X^{t_1}) = I(X^{t_1}) \cup I(X^{s_2})$.

For any $X_i, X_i \in \pi$, if $\langle X_i, X_j\rangle$ is 1-ST-DC, $D^h(U_i^j) \triangleq D^-(X_i, X_j)$ $\max_x \max_y$ $\{d_{X_i X_j}(x, y) \mid x \in I(X_i),\ y \in I_t(X_j)\}$. If $\langle X_i, X_j\rangle$ is 2^--ST-DC, let $D^h(U_i^j) = \frac{1}{2}$ $\max\{d(u_1) + d(u_2) \mid \forall [x_1, x_2] \in I(X_i) \times I(X_i), [y_1, y_2] \in I_t(X_j) \times I_t(X_k), \langle X_i, X_k\rangle$ 1- or 2^--ST-DC, and $\beta(e, u_1) + \beta(e, u_2) \leq C(e)$ for all $e \in E\}$. Let $D^h(U_i) = 0$, for all 1-throughput IBC $X_i \in \pi$.

The following theorem is an analogy to Theorem 2.1.

Theorem 5.2. *Let $G^h = \{\pi', E^h, \delta^h, C^h, D^h\}$ be a throughput IBC partition machine of $G = \{X, E, \delta, C, d\}$. For any $s_1 \in X^{s_1} \in \pi', s_2 \in X^{s_2} \in \pi'$, $t_1 \in X^{t_1} \in \pi', t_2 \in X^{t_2} \in \pi'$, there is a valid routing from $[s_1, s_2]$ to $[t_1, t_2]$ in G if and only if there is a valid routing from $[X^{s_1}, X^{s_2}]$ to $[X^{t_1}, X^{t_2}]$ in G^h.*

Proof:

$\Longrightarrow$

We shall prove that any given high level valid routing contains a low level valid routing. Suppose there is a valid routing from $[X^{s_1}, X^{s_2}]$ to $[X^{t_1}, X^{t_2}]$ in G^h, and denote the blocks on an optimal path (with respect to D^h) in order by $Y_1, Y_2, \ldots, Y_m \in \pi$ and $Z_1, Z_2, \ldots Z_n \in \pi$ respectively, where $s_1 \in Y_1$, $s_2 \in Z_1$, $t_1 \in Y_m$ and $t_2 \in Z_n$. Because an optimal path with respect to D^h is acyclic, if $i \neq j$, $Y_i \neq Y_j$, $Z_i \neq Z_j$; and if $Y_i = Z_j$, there is no $j' \neq j$ such that $Y_i = Z_{j'}$. Hence, for each Y_i, $1 \leq i \leq m$, if there is $Z_j = Y_i$ for some $1 \leq j \leq n$, then there are two individual flows in this routing going through Y_i in G^h, and by Lemmas 5.2 and 5.3, Y_i is a 2^--throughput IBC block.

Set $s_1^1 = s_1$, $s_1^2 = s_2$, $s_m^1 = t_1$, $s_n^2 = t_2$. Search from $i = 1$ to $i = m - 1$ by increasing order, if Y_i is such that there is $Z_j = Y_i$, then carry out the following procedure: (1) if s_i^1, s_j^2, s_{i-1}^1, s_{j-1}^2 are not set, arbitrarily choose $[s_i^1, s_j^2] \in I(Y_i) \times I(Z_j)$ and

$[s^1_{i-1}, s^2_{j-1}] \in I(Y_{i-1}) \times I(Z_{j-1})$;
(2) denote by $U(Y_i, Z_j)$ the set of $[u^1_i, u^2_j] \in E^* \times E^*$ such that $\delta(s^1_i, u^1_i) = s^1_{i-1}, \delta(s^2_j, u^2_j) = s^2_{j-1}$, $\forall u'_1 < u^1_i, u'_2 < u^2_j$, $\delta(s^1_i, u'_1) \in Y_i$, $\delta(s^2_j, u'_2) \in Z_j$, and for all $e \in E$, $\beta(e, u^1_i) + \beta(e, u^2_j) \le C(e)$. Since Y_i is 2^+-throughput IBC, $U(Y_i, Z_j) \neq \emptyset$. Seek $[u^{1^0}_i, u^{2^0}_j]$ such that $d(u^{1^0}_i) + d(u^{2^0}_j) \le d(u^1_i) + d(u^2_j)$ for all $[u^1_i, u^2_j] \in U(Y_i, Z_j)$.

We note that $d(u^{1^0}_i) + d(u^{2^0}_j) \le D^h(U(Y_i \to Y_{i-1})) + D^h(U(Z_j \to Z_{j-1}))$.

Starting from $i = 1$ to $m-1$ by increasing order, if each Y_i such that no $Z_j = Y_i$ exists, repeat the following procedure:
(1) if s^1_i, s^1_{i-1} are not set, arbitrarily choose $s^1_i \in I(Y_i)$ and $s^1_{i-1} \in I(Y_{i-1})$;
(2) no matter whether X_i is 1-throughput IBC or 2^+-throughput IBC, by Definitions 5.6 and 5.7, there is $u^1_i \in E^*$ such that $\delta(s^1_i, u^1_i) = s^1_{i-1}$ and for all $u'_i < u^1_i$, $\delta(s^1_i, u'_i) \in Y_i$. Denote the set of all such u^1_i by $U(Y_i)$. Seek $u^{1^0}_i \in U(Y_i)$ such that $d(u^{1^0}_i) \le d(u^1_i)$ for all $u^1_i \in U(Y_i)$.

It is clear that $d(u^{1^0}_i) \le D^h(U(Y_i \to Y_{i-1}))$.

Starting from $j = 1$ to $n-1$ by increasing order, for each Z_j such that no $Y_i = Z_j$ exists, repeat the process similar to above, seek $u^{2^0}_j \in U(Z_j)$. Also, $d(u^{2^0}_j) \le D^h(Z_j, Z_{j-1})$.

Clearly, $\delta(s_1, u^{1^0}_1 u^{1^0}_2 \ldots u^{1^0}_{m-1}) = t_1$, $\delta(s_2, u^{2^0}_1 u^{2^0}_2 \ldots u^{2^0}_{n-1}) = t_2$, and for all $e \in E$, $\beta(e, u^{1^0}_1 u^{1^0}_2 \ldots u^{1^0}_{m-1}) + \beta(e, u^{2^0}_1 u^{2^0}_2 \ldots u^{2^0}_{n-1}) \le C(e)$. Therefore, $[u^{1^0}_1 u^{1^0}_2 \ldots u^{1^0}_{m-1}, u^{2^0}_1 u^{2^0}_2 \ldots u^{2^0}_{n-1}]^T$ is a valid routing for $[s_1, s_2]^T$ and $[t_1, t_2]^T$ in G. And $d(u^{1^0}_1 u^{1^0}_2 \ldots u^{1^0}_{m-1}) + d(u^{2^0}_1 u^{2^0}_2 \ldots u^{2^0}_{n-1}) \le D^{h^{00}}([X^{s_1}, X^{s_2}]^T, [X^{t_1}, X^{t_2}]^T)$.

$\Longleftarrow$

We shall construct a high level valid routing based on any given low level valid routing. Suppose there is a valid routing $[u_1, u_2] \in E^* \times E^*$ from $[s_1, s_2] \in X \times X$ to $[t_1, t_2] \in X \times X$. Denote the high level blocks containing the low level paths $[p(s_1, u_2), p(s_2, u_2)]$

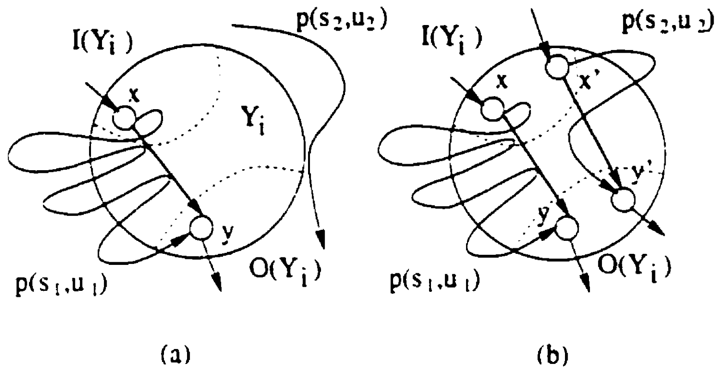

FIGURE 5.6. $p(s_1, u_1')$ and $p(s_2, u_2')$ go through Y_i no more than twice

driven by u_1 and u_2, respectively, from s_1 and s_2, respectively by $Y_1, Y_2, \ldots, Y_m \in \pi$ and $Z_1, Z_2, \ldots Z_n, \in \pi'$ by order.

Now we prove there are valid routing $[p(s_1, u_1'), p(s_2, u_2')]$ such that each of $p(s_1, u_1')$ and $p(s_2, u_2')$ goes through every Y_i, Z_j, $1 \le i \le m-1$, $1 \le j \le n-1$, no more than once.

For an arbitrary Y_i, if $p(s_1, u_1)$ goes through Y_i strictly more than once, because π is throughput IBC, Y_i is 2^--throughput IBC. Denote the first entry of $p(s_1, u_1)$ in $I(Y_i)$ by x, the last exit of $p(s_1, u_1)$ in $O(Y_i)$ by y. Let the segment from x to y of $p(s_1, u_1)$ be denoted by $x \rightsquigarrow y$.

There are two possible cases:
(1) $p(s_2, u_2)$ does not go through Y_i (see Figure 5.6(a)). By definition, there is $u \in E^*$ such that $\delta(x, u) = y$ and for all $u' < u$, $\delta(x, u') \in Y_i$. Replace $x \rightsquigarrow y$ by $p(x, u)$ for $p(s_1, u_1)$, denote the resulting path by $p(s_1, u_1 | p(x, u))$. $[p(s_1, u_1 | p(x, u)), p(s_2, u_2)]$ is a valid routing for $[s_1, s_2]$ and $[t_1, t_2]$.
(2) $p(s_2, u_2)$ goes through Y_i (see Figure 5.6(b)), denote the first entry of $p(s_1, u_1)$ in $I(Y_i)$ by x', the last exit of $p(s_1, u_1)$ in $O(Y_i)$ by y'. Let the segment from x' to y' of $p(s_2, u_2)$ be denoted by $x' \rightsquigarrow y'$. By definition, there is $[u, u'] \in E^* \times E^*$ such that $\delta(x, u) = y$ and for all $u_1 < u$, $\delta(x, u_1) \in Y_i$, $\delta(x', u') = y'$ and for all $u_2 < u'$, $\delta(x', u_2) \in Y_i$. Moreover, for all $e \in E$, $\beta(e, u) + \beta(e, u') \le C(e)$. Replace $x \rightsquigarrow y$ and $x' \rightsquigarrow y'$by $p(x, u)$ and $p(x', u')$ for $p(s_1, u_1)$ and $p(s_2, u_2)$ respectively, we obtain $[p(s_1, u_1 | p(x, u)), p(s_2, u_2 | p(x', u'))]$, a valid routing for $[s_1, s_2]$ and $[t_1, t_2]$.

In this way, we can obtain a valid routing $[p(s_1, u_1'), p(s_2, u_2')]$ for which $p(s_1, u_1')$ and $p(s_2, u_2')$ respectively go through each $Y_i, 1 \leq i \leq m$ and each $Z_j, 1 \leq j \leq n$ at most once. Rename the blocks containing $p(s_1, u_1')$ and $p(s_2, u_2')$ respectively by $V_1', V_2', \ldots, V_k'$ and $W_1', W_2', \ldots, W_l'$, for which $s_1 \in V_1', t_1 \in V_k', s_2 \in W_1', t_2 \in W_l'$. By Lemmas 5.2 and 5.3, if there is a low level transition from X_i to X_j, $\langle X_i, X_j \rangle$ is 1- or 2^--ST-DC. Hence, $\langle V_i', V_{i+1}' \rangle$ and $\langle W_j', W_{j+1}' \rangle$ all 1- or 2^--ST-DC, $1 \leq i \leq k-1$, $1 \leq j \leq l-1$. If $p(s_1, u_1')$ and $p(s_2, u_2')$ both go through $V_i' \to V_{i+1}'$, by definition, $\langle V_i', V_{i+1}' \rangle$ is 2^--ST-IBC. Therefore, $V_1' \to V_2' \to \ldots \to V_k'$ and $W_1' \to W_2' \to \ldots \to W_l'$ form a valid routing for $[X^{s_1}, X^{s_2}]$ to $[X^{t_1}, X^{t_2}]$ in G^h. □

Let the optimal high level cost for $[X^{s_1}, X^{s_2}]$ and $[X^{t_1}, X^{t_2}]$ be denote by $D([X^{s_1}, X^{s_2}], [X^{t_1}, X^{t_2}], D^h)$. from the proof of the above theorem, it is straightforward to see, there is a low level routing with a cost less than $D([X^{s_1}, X^{s_2}], [X^{t_1}, X^{t_2}], D^h)$.

5.4. Hierarchical Dynamical Routing for Networks with Buffers

5.4.1. Network States of Buffered Networks. Consider a network in the class BN represented by $G = \{N, E, \delta, d, B\}$ in which the capacity of links may be thought to be infinite. The capacity of a node, which may be thought to be the size of a buffer, is denoted $B(n) \in N^-$, for any $n \in N$.

Suppose the system is event-driven. We shall map the ordering of the occurrences of events to the positive integers, which is equivalent to viewing that the system clock as being discrete-valued. All exogenous events (the arrivals of new messages) take place non-simultaneously, i.e., at any event-indexed (or marked) time instant at most one new message arrives at G. Also assume that only one event (including control actions) happens at the a time instant.

Let $F_k(n)$ denote the total number of messages stored in the buffer of node $n \in N$ at a time instant $k \geq 1$. At time k, if $B(n_i) > F_k(n_i)$ for $n_i \in N$, a new message $msg_k(n_j)$, associated with a destination $n_j \in N$, $n_j \neq n_i$, may be accepted by the network and be stored in the buffer at the node n_i. This message $msg(n_j)$ can be either kept in the buffer at n_i or sent to $n_l \in N$ if the following two conditions are true: (1) there is a link $e \in E$ such that $\delta(n_i, e) = n_l$, and (2) $F_k(n_l) < B(n_l)$ when $n_l \neq n_j$. If $n_l = n_j$, $msg(n_j)$ reaches its destination and leaves the network without entering the buffer n_l; otherwise, it is to be stored in the buffer n_l.

A *(matrix) network state*, which bears the information on the number of messages $m_k(i, j)$, $1 \leq i, j \leq |N|$, at node n_i with destination as node n_j at time k is given by,

$$\gamma_k = \begin{bmatrix} 0 & m_k(1,2) & \ldots & m_k(1,|N|) \\ m_k(2,1) & 0 & \ldots\ldots & m_k(2,|N|) \\ & \ldots & \ldots & \\ m_k(|N|,1) & m_k(|N|,2) & \ldots & 0 \end{bmatrix}.$$

$$F_k(n_i) = \sum_{j=1}^{|N|} m_k(i,j) \leq B(n_i), \text{ for all } 1 \leq i \leq |N|,$$

where m_i^i is always 0. A distinguished state $\gamma_0 = [0]_{|N| \times |N|}$, the zero matrix, evidently represents the network state where no messages are being held by the network.

It may be proved by induction that the total number of network states is $\prod_{i=1}^{|N|} C_{B(n_i)}^{B(n_i)+|N|}$.

There are two classes of events in a BN network G, i.e., $E = A \cup T$, where
(1) $A = \{a_i^j | n_i \neq n_j\}$. Here a_i^j means a new message $msg(n_j)$ with destination n_j enters the network at node n_i.
(2) $T = \{t_{i,k}^j |$ there is a link in G from n_i to n_k, $n_i \neq n_j$, $n_k \neq n_i\}$. Here $t_{i,k}^j$ means that a message with destination n_j is transmitted from a node n_i via a link e_i^k to n_k.

It is clear that $|A| = |N| \times (|N| - 1)$, $|T| = (|N| - 1) \times |E|$.

Obviously, $t_{i,k}^j$ is a control action. When an action $t_{i,k}^j$ is taken at time instant l, if $n_k \neq n_j$, $m(i,j)$ decreases by 1 while $m(k,j)$ increases by 1, i.e., $m_{l+1}(i,j) = m_l(i,j) - 1$ and $m_{l+1}(k,j) = m_l(k,j) + 1$; if a message arrives at its destination, i.e., in case $n_k = n_j$, it leaves the network without entering the buffer at n_k, $m_{l+1}(i,j) =$

$m_l(i,j) - 1$, $F_{l+1}(k) = F_l(k)$. At time l, if the buffer of n_i is not full, a new message to n_j may arrive at n_i ($n_i \neq n_j$). The arrivals of new messages are spontaneous. Not every a_i^j is accepted by the network: if necessary, a_i^j may be rejected. After a_i^j is accepted by n_i, m_i^j increases by 1, i.e., $m_{l+1}(i,j) = m_l(i,j) + 1$.

The dynamics of the *buffered network state systems (BNS)* may be represented as a finite state machine, BNS(G)= $\{\Gamma, A \cup T, \nu\}$, where Γ is the set of all network states. We note that a network state (in the product system) is not a location in a network. Here $\nu(\gamma_k, u_k) = \gamma_{k+1}$, $u_k \in A \cup T$, $k \leq 0$. Explicitly,

$$\nu(\begin{bmatrix} 0 & m_k(1,2) & \dots & \dots & m_k(1,|N|) \\ & \dots & \dots & \dots & \\ m_k(i,1) & \dots & m_k(i,j) & \dots & m_k(i,|N|) \\ & \dots & \dots & \dots & \\ m_k(|N|,1) & \dots & \dots & \dots & 0 \end{bmatrix}, a_i^j) =$$

$$\begin{bmatrix} 0 & m_k(1,2) & \dots & \dots & m_k(1,|N|) \\ & \dots & \dots & \dots & \\ m_k(i,1) & \dots & m_k(i,j+1) & \dots & m_k(i,|N|) \\ & \dots & \dots & \dots & \\ m_k(|N|,1) & \dots & \dots & \dots & 0 \end{bmatrix},$$

$$\text{if } \sum_{l=0}^{N} m_k(i,l) < B(n_i).$$

$$\nu(\begin{bmatrix} 0 & m_k(1,2) & \dots & \dots & m_k(1,|N|) \\ & \dots & \dots & \dots & \\ m_k(l,1) & \dots & m_k(l,j) & \dots & m_k(l,|N|) \\ & \dots & \dots & \dots & \\ m_k(i,1) & \dots & m_k(i,j) & \dots & m_k(i,|N|) \\ & \dots & \dots & \dots & \\ m_k(|N|,1) & \dots & \dots & \dots & 0 \end{bmatrix}, t_{i,l}^j) =$$

$$\begin{bmatrix} 0 & m_k(1,2) & \dots & \dots & m_k(1,|N|) \\ & \dots & \dots & \dots & \\ m_k(l,1) & \dots & m_k(l,j)+1 & \dots & m_k(l,|N|) \\ & \dots & \dots & \dots & \\ m_k(i,1) & \dots & m_k(i,j)-1 & \dots & m_k(i,|N|) \\ & \dots & \dots & \dots & \\ m_k(|N|,1) & \dots & \dots & \dots & 0 \end{bmatrix}.$$

if $n_l \neq n_j$, i.e. $l \neq j$, and $\sum_{n=0}^{|N|} m_k(l,n) < B(n_l)$.

$$\nu(\begin{bmatrix} 0 & m_k(1,2) & \dots & \dots & m_k(1,|N|) \\ & \dots & \dots & \dots & \\ m_k(l,1) & \dots & m_k(l,j) & \dots & m_k(l,|N|) \\ & \dots & \dots & \dots & \\ m_k(i,1) & \dots & m_k(i,j) & \dots & m_k(i,|N|) \\ & \dots & \dots & \dots & \\ m_k(|N|,1) & \dots & \dots & \dots & 0 \end{bmatrix}, t^j_{i,l}) =$$

$$\begin{bmatrix} 0 & m_k(1,2) & \dots & \dots & m_k(1,|N|) \\ & \dots & \dots & \dots & \\ m_k(l,1) & \dots & 0 & \dots & m_k(l,|N|) \\ & \dots & \dots & \dots & \\ m_k(i,1) & \dots & m_k(i,j)-1 & \dots & m_k(i,|N|) \\ & \dots & \dots & \dots & \\ m_k(|N|,1) & \dots & \dots & \dots & 0 \end{bmatrix}.$$

if $n_l = n_j$, i.e. $l = j$

Otherwise, ν is undefined.

5.4.2. The Controllability of the Buffer Network State Systems. A network state is said to be *BNS controllable (to γ_0)* if there is a finite sequence of actions in T that drive the network to γ_0.

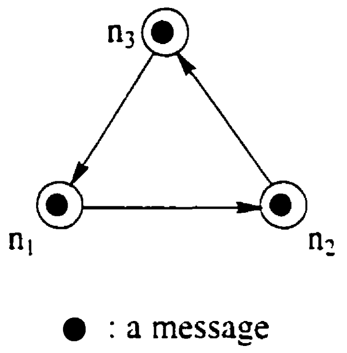

FIGURE 5.7. Network in deadlock if $m_k(1,3) = m_k(2,1) = m_k(3,2) = 1$

Uncontrollable network states include deadlocks and live-locks. A *deadlock* is a state of a network in which the network cannot accept any control actions. In contrast, a *live-lock* is a state such that the network can make some transitions in T but is not BNS controllable to γ_0.

Example 5.3. *In Figure 5.7, the network state is*

$$\gamma_k = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

The capacity of the buffers of all nodes is 1. This network is in a deadlock because no actions can be taken at this state.

In Figure 5.8, the network state is

$$\gamma_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The capacity of buffer of all nodes is 1. This network is in a live-lock because no message can be sent to its destination. □

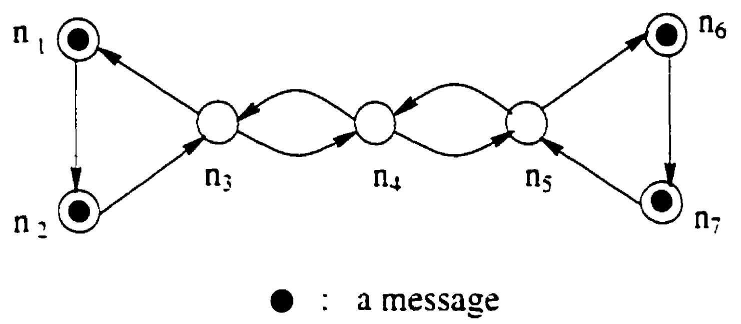

FIGURE 5.8. Network in a live-lock when $m_k(1,7) = m_k(2,7) = m_k(6,2) = m_k(7,2) = 1$

To prevent the network from entering deadlocks or live-locks, it is necessary that some arrivals of new messages should not be accepted, i.e., if $\nu(\gamma_k, a_i^j) = \gamma_{k-1}$, $k \geq 0$, such that γ_{k-1} is not BNS controllable, the message $msg(n_j)$ should be rejected and not put into the buffer at n_i. Given an initial network state γ_{k_0}, the control objective is to find a sequence of control actions in T that drive the network from γ_{k_0} to γ_0, which embodies the completion of all tasks, while avoiding getting into deadlocks or live-locks.

A network G is said to be *path-wise controllable* if for all $n_i, n_j \in N$, there exists a path from n_i to n_j.

Lemma 5.4. *If a network $G = \{N, E, \delta, d, B\}$ is completely path-wise connected, then $BNS(G)$ is deadlock free if and only if there is a node, $n_i \in N$, such that for all $n_j \in N$, $i \neq j$, there is $e_i^j \in E$, s.t. $\delta(n_i, e_i^j) = n_j$.*

Proof:

$\Longrightarrow$

For any network state $\gamma_{k_0} \neq \gamma_0$, all messages at n_i can be sent directly to their destinations n_j, $1 \leq j \leq |N|$, $j \neq i$, by control actions $t_i^{i,j}$. In other words, after taking

$$\underbrace{t^1_{i,1}\cdots t^1_{i,1}}_{m_{k_0}(i,1)}\underbrace{t^2_{i,2}\cdots t^2_{i,2}}_{m_{k_0}(i,2)}\cdots t^{i-1}_{i,i-1}t^{i-1}_{i,i-1}\cdots\underbrace{t^{|N|}_{i,|N|}\cdots t^{|N|}_{i,|N|}}_{m_{k_0}(i,|N|)},$$

the network state is

$$\gamma_{k_0+F_{k_0}(i)} = \begin{bmatrix} 0 & m_{k_0+F_{k_0}(i)}(1,2) & \cdots & m_{k_0+F_{k_0}(i)}(1,|N|) \\ & \cdots & \cdots & \\ m_{k_0+F_{k_0}(i)}(i-1,1) & \cdots & \cdots & m_{k_0+F_{k_0}(i)}(i-1,|N|) \\ 0 & 0 & \cdots & 0 \\ m_{k_0+F_{k_0}(i)}(i+1,1) & \cdots & \cdots & m_{k_0+F_{k_0}(i)}(i+1,|N|) \\ & \cdots & \cdots & \\ m_{k_0+F_{k_0}(i)}(|N|,1) & m_{k_0+F_{k_0}(i)}(|N|,2) & \cdots & 0 \end{bmatrix}.$$

Because G is path-wise controllable, for any $n_j \in N$, $n_j \neq n_i$, there is path from n_j to n_i. Let $|u_j|$ be the length of the shortest path from n_j to n_i in terms of the number of links. n_j is said to be $|u_j|$ *step reachable to* n_i. Let K be the maximum of $|u_j|$ for all $1 \leq j \leq |N|$, $j \neq i$.

First, suppose the node n_j is one step reachable to n_i. For a message to n_k stored at n_j,
(1) if $k = i$, the control action $t^i_{j,i}$ makes that message reach its destination and leave the network;
(2) otherwise, it may be sent to n_k via n_i by taking $t^k_{j,i}t^k_{i,k}$.
In this way, all messages at the nodes one step reachable to n_i can be forwarded to their destinations.

Recursively drain off the messages in the buffers at the nodes n step reachable to n_i, $1 < n \leq K$, the BNS(G) arrives at γ_0. Therefore, no deadlock state exists for this network.

$\Longleftarrow$

Suppose no such n_i exists, that is to say, for any node $n_j \in N$, there is a node n_{k_j} which is 2 step reachable from n_j. In a network state such that $B(n_j)$ messages with destination n_{k_j} are in the buffer n_j for all $n_j \in N$, the network state is in deadlock. This is because any control action $t^{k_j}_{j,l}$ ($k_j \neq l$ as assumed) will cause the overflow of the buffer of n_l. □

From the proof to Lemma 5.4, it is evident that if there is $n_i \in \mathcal{N}$ such that n_i is reachable to any other nodes in $\mathcal{N}$ in one step, the network state system BNS(G) is live-lock free.

Let $\rightarrow$ represent a one step transition in E. A *ring* is a non-trivial circuit $R(n) = x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n \rightarrow x_1$, $n > 1$, such that for all $1 \leq i, j \leq n$, if $i \neq j$, $x_i \neq x_j$. An *arc* from x_i to x_j is the part of $R(n)$ from x_i to x_j, i.e., $Arc(x_i, x_j) = x_i \rightarrow \ldots \rightarrow x_j$. If at time k_0, $\sum_{j=1}^{|\mathcal{N}|} m_{k_0}(i,j) < B(n_i)$ for a node $n_i \in \mathcal{N}$, then an *empty buffer place* is said to exist in n_i. Moreover, $B(n_i) - \sum_{j=1}^{|\mathcal{N}|} m_{k_0}(i,j)$ empty buffer places are said to exist in n_i at time k_0.

Lemma 5.5. *For a ring $R(k) = n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_k \rightarrow n_1$, $n_i \in \mathcal{N}$ such that $i \neq j$, $n_i \neq n_j$, $1 \leq i, j \leq k$, if at time instant k_0, $\sum_{i=1}^{k} (\sum_{j=1}^{|\mathcal{N}|} m_{k_0}(i,j) < \sum_{i=1}^{k} B(n_i)$, then the following conditions are true:*

(1) For any $1 \leq i \leq k$, there is a finite sequence of control actions in T such that at a time instant $k' \geq k_0$, $(\sum_{j=1}^{|\mathcal{N}|} m_{k'}(i,j) < B(n_i)$.

(2) For any message $msg(j)$ in $R(k)$ at k_0, $1 \leq j \leq k$, there is a finite sequence of control actions in T such that at a time instant $k' > k_0$, it is sent to n_j.

(3) For any message $msg(j)$ in $R(k)$ at k_0, $j > k$, and for all $1 \leq l \leq k$, there is a finite sequence of control actions in T such that at a time instant $k' > k_0$, it is sent to n_l.

Proof:

Suppose at time instant k_0, $\sum_{i=1}^{k} (\sum_{j=1}^{|\mathcal{N}|} m_{k_0}(i,j) < \sum_{i=1}^{k} B(n_i)$. That is to say, there exists n_j in $R(k)$, such that there is an empty buffer place in n_j at time instant k_0, i.e., $(\sum_{l=1}^{|\mathcal{N}|} m_{k_0}(j,l) < B(n_j)$.

(1) For any $1 \leq i \leq k$, if $i = j$, then let $k' = k_0$. Lemma 5.5(1) is obviously true. Otherwise, $R(k)$ consists of two paths $Arc(n_i, n_j)$ and $Arc(n_j, n_i)$. Rename the nodes in $\mathcal{N}$ such that $Arc(n_i, n_j)$, the part of $R(k)$ from n_i to n_j, is denoted by

$$v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_{n-1} \rightarrow v_n.$$

where $v_1 = n_i$ and $v_n = n_j$.

Let $l = n - 1$. $k' = k_0$. If $\sum_{j=1}^{|N|} m_{k'}(l, j) = B(v_l)$. send an arbitrary message $msg(m_l)$ at v_l to v_{l+1}. i.e.. let $a_l = t_{l,l+1}^{m_l}$: otherwise let $a_l = \epsilon$. Hence. by taking a_l. $\sum_{j=1}^{|N|} m_{k'+1}(l, j) < B(v_l)$. If $l > 0$. set $l = l - 1$. $k' = k' + 1$. repeat the above procedure. Therefore. after taking $a_{n-1}a_{n-2} \ldots a_1$. at time $k' = k_0 + n - 1$. $(\sum_{l=1}^{|N|} m_{k'}(1, l)$ $< B(v_1)$. An empty place is thus generated in $n_i = v_1$.

This process creates an empty buffer place in n_i based on an empty buffer place in n_j on $R(k)$ and will be denoted by $CE(n_j \rightarrow n_i, R(k))$.

(2) For any message $msg(m)$ in $R(k)$ at k_0. $1 \leq m \leq k$. without loss of generality. suppose it is in n_i for some $1 \leq i \leq k$. For any node $n_j \neq n_m$ in $R(k)$. rename the nodes in N such that $Arc(n_i, n_j)$ is denoted by

$$v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_{n-1} \rightarrow v_n.$$

where $v_1 = n_i$ and $v_n = n_m$.

According to (1). if at k_0. the empty buffer place in v_l. then $CE(v_l \rightarrow v_i, R(k))$ will create an empty place in the buffer of v_i. $1 \leq i \leq n$ after a finite time. First. do $CE(v_l \rightarrow v_2, R(k))$. This will not force $msg(n)$ to leave v_1 because as the immediate predecessor of v_2 in $R(k)$. v_1 can not be an intermediate node on $Arc(v_2, v_l)$ and thus no $t_{1,2}^n$ is taken in $CE(v_l \rightarrow v_2, R(k))$. Then take $t_{1,2}^n$. At time $k_2 + 1$. there is an empty buffer place in v_1. Therefore. take a sequence of actions $CE(v_1 \rightarrow v_3, R(k))t_{2,3}^n$ $CE(v_2 \rightarrow v_4, R(k))t_{3,4}^n \ldots CE(v_{n-2} \rightarrow v_n, R(k))t_{n-1,n}^n$. $msg(n)$ is sent to v_n and leave the network.

(3) Since $msg(m)$ will not reach n_m around $R(k)$. it will not leave the network. By an analogous argument to the above. $msg(m)$ can be sent to any node in $R(k)$. Denote this process that a message $msg(m)$ at n_i is sent to an arbitrary node n_j on $R(k)$ by $SO(n_i \rightarrow n_j, R(k))$. □

Corollary 5.1. *For $G = \{N, E, \delta, d, B\}$, if all nodes in N forms a ring, i.e., there is a circuit, $n_1 \to n_2 \to \dots \to n_{|N|} \to n_1$, $n_i \in N$ such that $i \neq j$, $n_i \neq n_j$, $1 \leq i, j \leq |N|$, then BNS(G) is free of live-locks.*

Proof:

At a time instant k_0, if the network G is in γ_{k_0}, which is not a deadlock state, there must be $t^k_{i,j} \in R$, $\gamma_{k_0-1} \in \Gamma$, such that $\nu(\gamma_{k_0}, t^k_{i,j}) = \gamma_{k_0-1}$. Therefore, $(\sum_{j=1}^{|N|} m_{k_0-1}(i,j)) < B(n_i)$. In other words, an empty buffer place in n_i is generated at time $k_0 - 1$.

Suppose that all nodes in N form a ring without the repetition of any node. If at time k_0' there is an empty buffer place in n_a, then by Lemma 5.5(2), any message with destination n_k at n_h at time k_0' can be sent to n_k in finite steps.

Hence, all messages will be sent to their destinations in finite time. That is to say, γ_{k_0} is reachable to γ_0. $\square$

Clearly, if G is Hamiltonian([26]), BNS(G) is live-lock free.

Define an operation *concatenation* over two paths: for any $p_1 = n_1 \to n_2 \to \dots \to n_k \in N^*$, $p_2 = m_1 \to m_2 \to \dots \to m_j \in N^*$, the concatenation of p_1 and p_2,

$$p_1 p_2 = \begin{cases} n_1 \to n_2 \to \dots \to n_k \to m_1 \to m_2 \to \dots \to m_j & \text{if } m_1 \neq n_k \text{ and} \\ & \exists e \in E. \delta(n_k, e) = m_1 \\ n_1 \to n_2 \to \dots \to n_k \to m_2 \to \dots \to m_j & \text{if } m_1 = n_k \\ \text{undefined} & \text{otherwise} \end{cases}$$

Lemma 5.6. *If G is path-wise controllable, then there is a circuit C in G covering all nodes in N, and C can be represented as a set of intersecting rings $\{R_i, 1 \leq i \leq n, R_i \cap R_{i-1} \neq \emptyset, 1 \leq i < n\}$.*

Proof:

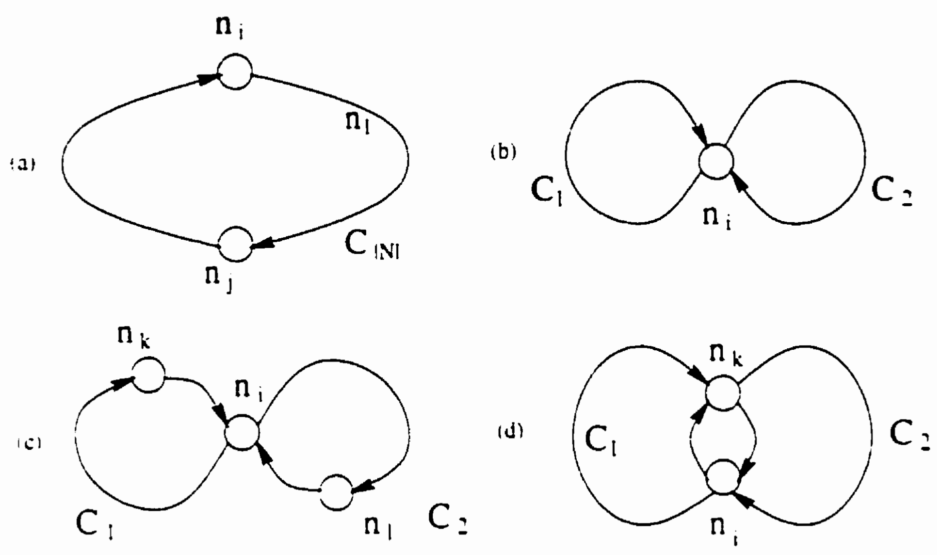

FIGURE 5.9. A circuit with repetition of nodes

Since G is path-wise controllable, for any $x, y \in \mathcal{N}$, there exists $u \in E^*$ such that $\delta(x, u) = y$. A circuit that covers all nodes in $\mathcal{N}$ can be constructed in the following way:

Let $C(1) = n_1$, $i = 1$. if n_{i+1} is not included by $C(i)$, find $u(i) \in E^*$ such that $\delta(n_i, u(i)) = n_{i+1}$, and denote $p_i = p(n_i, u_i)$; otherwise, let $p_i = \epsilon$. $C(i+1) = C(i)p_i$. If $i < |\mathcal{N}|$, let $i = i + 1$; repeat the above procedure. When $i = |\mathcal{N}|$, seek $u(|\mathcal{N}|) \in E^*$ such that $\delta(n', u(|\mathcal{N}|)) = n_1$, where $n' \in \mathcal{N}$ is the last node in $C(|\mathcal{N}|)$. $C(|\mathcal{N}| + 1) = C(|\mathcal{N}|)p((n', u(|\mathcal{N}|))$ is such a circuit.

Rename the nodes on $C(|\mathcal{N}| + 1)$ to be $v_i, 1 \le i \le |C_{\mathcal{N}+1}|$. Then if $v_i = v_j$, break down $C(|\mathcal{N}| + 1)$ into two circuits C_1 and C_2 such that $C_1 \cap C_2 = \{n_i\}$ (Figure 5.9(a) and (b)). If two nodes in C_1, $n_k = n_l$, C_1 may be further broken down into $C_{1,1}$ and $C_{1,2}$ such that $C_{1,1} \cap C_{1,2} = \{n_k\}$. If n_k is in C_1, n_l is in C_2, and $n_k = n_l$, then two circuits $C_1 \cap C_2 = \{n_i, n_k\}$ (Figure 5.9(c) and (d)). Thus, by breaking down circuits with repetition of n_i into intersecting circuits without repetitions of n_i, $C(|\mathcal{N}| + 1)$ can be represented by a union of intersecting rings $\{R_i, 1 \le i \le n \mid R_i \cap R_{i-1} \neq \emptyset, 1 \le i < n\}$. ◫

Theorem 5.3. *If all nodes in $\mathcal{N}$ are on $\{R_i, 1 \leq i \leq n | R_i \cap R_{i+1} \neq \emptyset, 1 \leq i < n\}$, then any network state γ_{k_0} with $\sum_{i=1}^{|\mathcal{N}|} \sum_{j=1}^{|\mathcal{N}|} m_i^j(k_0) \leq \sum_{i=1}^{|\mathcal{N}|} B(n_i) - n$, is a BNS controllable network state.*

Proof:

Suppose $\sum_{i=1}^{|\mathcal{N}|} \sum_{j=1}^{|\mathcal{N}|} m_{k_0}(i,j) \leq \sum_{i=1}^{|\mathcal{N}|} B(n_i) - n$. That is to say, there are n empty buffer places in $\mathcal{N}$ at time k_0.

By Lemma 5.5(1), for any v_i, v_j in the same ring R_k, if there is an empty buffer place in v_i, then after a finite time, by $CE(v_i \rightarrow v_j, R_k)$, there will be an empty place in v_j, i.e. any empty buffer place can be circulated around R_k.

Furthermore, if $v_k \in R_i \cap R_{i-1}$, and there is an empty buffer place in R_i at k_0, then an empty buffer place may be generated in v_k after a finite time, and by the same reason, an empty buffer place can be created in any node on R_{i-1} after a finite time.

Therefore, the n empty buffer places can be exchanged between intersecting rings. Assume at a time instant $k' > k_0$, each ring has an empty buffer place.

For a message msg in R_i with destination in another node in R_i, by Lemma 5.5(2), msg can be sent to its destination since there is an empty buffer place in R_i at k'.

For a message msg in $v_l \in R_i$ with destination in R_j, $j > i$. Circulate the empty buffer places such that there is an empty buffer place in $v_e(k) \in R_k - R_k \cap R_{k-1}$ for each $i < k \leq j$. Then, by Lemma 5.5(3), msg can be sent to a node in $v_c(i) \in R_i \cap R_{i-1}$ by $SO(v_l \rightarrow v_c(i), R_i)$. Since $v_e(i) \notin R_i \cap R_{i-1}$, $SO(v_l \rightarrow v_c(i), R_i)$ will not change the empty buffer place in $v_e(i)$. That is to say, any message in R_i can be circulated to $R_i \cap R_{i-1}$ and R_{i-1} still has an empty buffer place (see Figure 5.10). Hence, msg

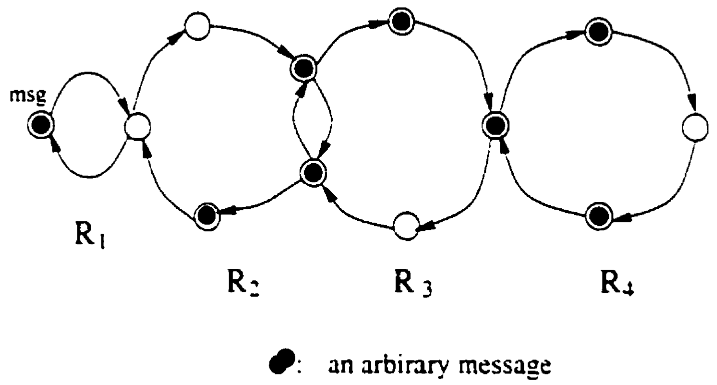

FIGURE 5.10. All nodes in $\mathcal{N}$ are on R_i

can be sent to $R_{j-1} \cap R_j$ through $R_{i-1}...R_{j-1}$ and then sent to its destination.

For a message *msg* in $v_l \in R_i$ with destination in R_j, $j < i$. Circulate the empty buffer places such that there is an empty buffer place in $v_e(k) \in R_k - R_k \cap R_{k-1}$ for each $j \leq k < i$. By an argument analogous to the above, *msg* can be sent to $R_j \cap R_{j-1}$ and then to its destination.

Therefore, all messages can be sent to their destinations, i.e., γ_{k_0} is BNS controllable.

□

Let $\alpha(G)$ be the minimal number of rings for which there exists a circuit C covering all nodes in $\mathcal{N}$ such that C is the set of rings. If at time k_0, the number of empty buffer places in G is strictly less than $\alpha(G)$, the network is possibly in a live-lock, e.g., the network state as shown in Figure 5.3.

5.4.3. High Level Network States of the Buffer Network Systems. In addition to directly applying HADP to the network state machine BNS(G), we present a formulation of hierarchical control by partitioning $\mathcal{N}$, the set of nodes of G into IBC blocks to reduce the computational complexity of seeking the optimal control (a shortest path problem for network state representation BNS(G)). This is because the number of network states is usually much greater than that of the nodes in the network.

Let $\pi = \{X_1, X_2, \dots, X_{|\pi|}\}$ be an IBC partition of N. Define the size of buffer for a block $X_i \in \pi$ by $B^h(X_i) \triangleq \sum_{n_j \in X_i} B(n_j)$. The number of messages stored in X_i with destination as nodes in X_j $(X_j \neq X_i)$ at time instant k is $M_k(i,j) = \sum_{n_l \in X_i, n_m \in X_j} m_k(l,m)$. A high level network $G^h = \{\pi, E^h, \delta^h, D^h, B^h\}$ is defined based on DC relations over π. A high level network state.

$$\gamma_k^h = \begin{bmatrix} 0 & M_k(1,2) & \dots & M_k(1,|\pi|) \\ M_k(2,1) & 0 & \dots & M_k(2,|\pi|) \\ & \dots & \dots & \\ M_k(|\pi|,1) & M_k(|\pi|,2) & \dots & 0 \end{bmatrix}.$$

where $F_k^h(X_i) = \sum_{j=1}^{|\pi|} M_k(i,j) < B^h(X_i)$, for all $1 \leq i \leq |\pi|$.

High level events include:
(1) $A^h = \{A_i^j \; i \neq j, 1 \leq i,j \leq |\pi|\}$: a new message with destination in X_j arrives at X_i, $X_i \neq X_j$, at time instant $k_0 + 1$, if the $F_{k_0}^h(X_i) < B(X_i)$.
(2) $T^h = \{T_{i,j}^l \; i \neq j, i \neq l, \langle X_i, X_j \rangle$ is DC $1 \leq i,j,l \leq |\pi|\}$: a message with destination in X_l is sent from X_i to X_j at time instant k_0+1, if $X_j = X_k$ or $F_{k_0}^h(X_j) < B(X_j)$.

For a block $X_i = \{n_1^i, n_2^i, \dots, n_{|X_i|}^i\}$, an *in-block network state* with respect to X_i at time instant k bears information about the number of messages with destination in $\{n_1^i, n_2^i, \dots, n_{|X_i|}^i\} \cup (\pi - \{X_i\})$ being processed at node n_j^i, $1 \leq j \leq |X_i|$.

$$\gamma_k(X_i) = \left[\begin{matrix} 0 & m_k(n_1^i, n_2^i) & \dots & m_k(n_1^i, n_{|X_i|}^i) \\ m_k(n_2^i, n_1^i) & 0 & \dots & m_k(n_2^i, n_{|X_i|}^i) \\ & \dots & \dots & \\ m_k(n_{|X_i|}^i, n_1^i) & m_k(n_{|X_i|}^i, n_2^i) & \dots & 0 \end{matrix} \right.$$

$$\left. \begin{matrix} m_k(n_1^i, X_1) & \dots & m_k(n_1^i, X_{|\pi|}) \\ m_k(n_2^i, X_1) & \dots & m_k(n_2^i, X_{|\pi|}) \\ & \dots & \\ m_k(n_{|X_i|}^i, X_1) & \dots & m_k(n_{|X_i|}^i, X_{|\pi|}) \end{matrix} \right].$$

$$F_k(n_j^i) = \sum_{l=1}^{|X_i|} m_k(n_j^i, n_l^i) + \sum_{l=1, l\neq i}^{|\pi|} m_k(n_j^i, X_l) < B(n_j^i).$$

where $m_k(n_j^i, X_l)$, $m_k(n_j^i, n_l^i)$ represent the number of messages being processed at node n_j^i at time k with destination in block X_l and node $n_l^i \in X_i$ respectively.

An control action $t_{i,j}^k \in T$ is said to be an *in-block control action with respect to* X_i if both n_i and n_j are in X_i.

Definition 5.8. An in-block network state of IBC block $X_i \in \pi$, $\gamma_k(X_i)$ is said to be a BNS in-block controllable state if, for all $n_l^i \in X_i$, all $n_e^i \in I(X_i)$ and all $n_o^i \in O(X_i)$, there is a finite sequence (length $k' - k$) of in-block control actions with respect to X_i such that
(1) $F_{k'}(n_e^i) < B(n_e^i)$, and
(2) if $M_k(i,j) > 0$, $m_o^i(X_j, k') > 0$, and
(3) $\sum_{l=1}^{X_i} \sum_{m=1}^{X_i} m_{k'}(n_l^i, n_m^i) = 0$. □

If an IBC block X_i is in a BNS in-block controllable state at a time instant k, after taking a finite sequence of in-block control actions, without the occurrence of events in A^h and A, the three conditions above will be true: (1) there will be an empty buffer place in any given in-set node, (2) any message to another block will be sent to any given out-set node, and (3) all messages with destinations in X_i will be sent to their destinations.

Theorem 5.4. *Given an IBC block X_i consisting of k rings, X_i is in a BNS in-block controllable state $\gamma_k(X_i)$ if the total number of messages in X_i at k is less than or equal to $B^h(X_i) - k$.*
Proof:
Similar to the proof of Theorem 5.3. □

Redefine the IBC partition machine $G_c^h = \{\pi, E^h, \delta^h, D^h, B_c^h\}$, where $B_c^h(X_i) = B^h(X_i) - \alpha(X_i)$ for all $X_i \in \pi$. The network corresponding to G_c^h, $M^h(G_c^h) =$

$\{\Gamma^{hc}, A^h \cup T^h, \nu^h\}$. The dynamics of $M^h(G_c^h)$ are defined by

$$\nu^h(\begin{bmatrix} 0 & M(1,2) & \dots & \dots & M(1,|\pi|) \\ & \dots & \dots & \dots & \\ M(i,1) & \dots & M(i,j) & \dots & M(i,|\pi|) \\ & \dots & \dots & \dots & \\ M(|\pi|,1) & \dots & \dots & \dots & 0 \end{bmatrix}, A_i^j) =$$

$$\begin{bmatrix} 0 & M(1,2) & \dots & \dots & M(1,|\pi|) \\ & \dots & \dots & \dots & \\ M(i,1) & \dots & M(i,j)+1 & \dots & M(i,|\pi|) \\ & \dots & \dots & \dots & \\ M(|\pi|,1) & \dots & \dots & \dots & 0 \end{bmatrix},$$

if $\sum_{k=0}^{|\pi|} M(i,k) < B_c^h(X_i)$,

$$\nu(\begin{bmatrix} 0 & M(1,2) & \dots & \dots & M(1,|\pi|) \\ & \dots & \dots & \dots & \\ M(k,1) & \dots & M(k,j) & \dots & M(k,|\pi|) \\ & \dots & \dots & \dots & \\ M(i,1) & \dots & M(i,j) & \dots & M(i,|\pi|) \\ & \dots & \dots & \dots & \\ M(|\pi|,1) & \dots & \dots & \dots & 0 \end{bmatrix}, T_{i,j}^k) =$$

$$\begin{bmatrix} 0 & M(1,2) & \dots & \dots & M(1,|\pi|) \\ & \dots & \dots & \dots & \\ M(k,1) & \dots & M(k,j)+1 & \dots & M(k,|\pi|) \\ & \dots & \dots & \dots & \\ M(i,1) & \dots & M(i,j)-1 & \dots & M(i,|\pi|) \\ & \dots & \dots & \dots & \\ M(|\pi|,1) & \dots & \dots & \dots & 0 \end{bmatrix},$$

if $k \neq j$ and $\sum_{l=0}^{|\pi|} M(k,l) < B_c^h(X_k)$.

$$\nu(\begin{bmatrix} 0 & M(1,2) & \dots & \dots & M(1,|\pi|) \\ & \dots & \dots & \dots & \\ M(k,1) & \dots & M(k,j) & \dots & M(k,|\pi|) \\ & \dots & \dots & \dots & \\ M(i,1) & \dots & M(i,j) & \dots & M(i,|\pi|) \\ & \dots & \dots & \dots & \\ M(|\pi|,1) & \dots & \dots & \dots & 0 \end{bmatrix} , T_{i,j}^{j}) =$$

$$\begin{bmatrix} 0 & M(1,2) & \dots & \dots & M(1,|\pi|) \\ & \dots & \dots & \dots & \\ M(k,1) & \dots & M(k,j) & \dots & M(k,|\pi|) \\ & \dots & \dots & \dots & \\ M(i,1) & \dots & M(i,j)-1 & \dots & M(i,|\pi|) \\ & \dots & \dots & \dots & \\ M(|\pi|,1) & \dots & \dots & \dots & 0 \end{bmatrix} .$$

if $\sum_{l=0}^{\pi} M(j,l) < B_c^h(X_j)$.

Otherwise, ν^h is undefined.

Theorem 5.5. *Let G^h be an IBC partition machine of G. if at time instant k_0, $\Gamma_{k_0}^{hc}$ of $M^h(G_c^h)$ is in a BNS controllable network state, $\gamma_{k_0}(X_i)$ is a BNS in-block controllable state for all $X_i \in \pi$, then γ_{k_0} is a BNS controllable state of $M(G)$.*

Proof:

Suppose at time instant k_0, $M^h(G_c^h)$ is in a BNS controllable network state $\Gamma_{k_0}^{hc}$, that is to say, there is a sequence of control actions in T^h that drives $M^h(G_c^h)$ from $\Gamma_{k_0}^{hc}$ to Γ_0. Also suppose all blocks $X_i \in \pi$ are in BNS in-block controllable states at k_0.

Because all messages in X_i with destination in X_i can be sent to their destinations, by Definition 5.8, it is equivalent to proving that each of the high level control commands is realisable by low level controllers and after completing a high level control command, $\gamma_{k_0}(X_i)$ is still a BNS in-block controllable state.

Let a message $msg(l)$ with $n_l \in X^k$ is held in X_i at time k'. By definitions, for a high level command $R^k_{i,j}$, it is defined at a state $\Gamma^{hc}_{k'}$ for some $k' > k_0$ if: (1) $M_{k'-1}(i,k) > 0$, (2) $\langle X_i, X_j \rangle$ is DC, and (3) $F^h_{k'-1}(X_j) < B^h_c(X_j)$.

Since $\gamma_{k'-1}(X_i)$ is a BNS in-block controllable state, by Definition 5.8, there is an outset node $n^i_o \in O_j(X_i)$, such that at time $k_1 > k'$, after the in-block controller takes a sequence of control actions with respect to X_i, $m^i_o(X_j, k_1) > 0$. In other words, there is a message at n^i_o with destination in X_k at k_1.

Since $\langle X_i, X_j \rangle$ is DC, $n^i_o \in O_j(X_i)$, there is $e \in E$, $n^j_e \in I_i(X_j)$, such that $\delta(n^i_o, e) = n^j_e$.

Because $\gamma_{k'-1}(X_j)$ is a BNS in-block controllable state, by Definition 5.8, $n^j_e \in I_i(X_j)$, such that at time $k_2 > k_1 > k'$, after the in-block controller takes a sequence of control actions with respect to X_j, $F_{k_2}(n^j_e) < B(n^j_e)$.

Therefore, at time $k_3 = k_2 - 1$, $r^l_{n^i_o, n^j_e}$ is well-defined. Let the low level controller take this control action $r^l_{n^i_o, n^j_e}$, then $msg(l)$ is a sent to block X_j at k_3.

At k_3, $F^h_{k_3}(X_i) = F^h_{k_0}(X_i) - 1$, evidently $\gamma_{k_3}(X_i)$ is BNS controllable after removing a message from a BNS controllable state $\gamma_{k_0}(X_i)$. $F^h_{k_3}(X_j) = F_{k_0}(X_j) + 1$ $\leq B^h_c(X_j) \leq B^h(X_j) - \alpha(X_j)$, by Theorem 5.4, $\gamma_{k_3}(X_j)$ is a BNS in-block controllable state.

In case $k = j$, by Definition 5.8, $msg(l)$ can be sent to its destination by low level controller and hence leave the network, X_j remains in BNS controllable state.

Thus, γ_0 is reachable. □

CHAPTER 6

Future Research

In this thesis we present a hierarchical control framework based on the notion of state aggregation via the dynamic consistency relations over the partition blocks of a given state space. Some suggestions for future research are given below.

Throughout this thesis the mathematical framework is deterministic. If the system state transitions are allowed to be probabilistic, for instance Markovian, then both the fundamental issues of the construction of the DC relations and the associated partition machines, and the solution of optimal control problems become challenging issues within the framework of stochastic system theory.

6.1. Research Related to HADP

A relation to be investigated is the size of partition blocks and the ratio of computational times taken by HADP and atomic search (i.e. search for an optimal control in the original system). This will result in an optimal partition in the sense that one multiple level hierarchy has the greatest magnitude of acceleration by HADP among all hierarchies with the same number of levels.

The relationship between the quality of a partition and the sub-optimality of the HADP method is given in Chapter 3. It is of great interest to investigate how to generate a partition within a given level of tolerance of optimality. Recall that all IBC partitions of a base finite state machine form a lattice. Through investigating the relation of the degree of optimality (statistical measurements might be used to represent this degree) and the position of a partition in the lattice, it is hoped that optimal partitions can be determined for which a two level HADP (consisting of the original system and a partition machine) will achieve solutions closest (on average) to the global optimum.

In order to judge the quality of partition before applying HADP, a good estimate of the cost of high level optimal solution becomes necessary. The combinatorial nature of in-block paths may make these estimates very loose. An improvement of estimation of the error bounds will definitely lead to the improvement of the HADP methodology.

6.2. Research Related to Multi-agent Systems

In the analysis of multi-agent systems in Chapter 4, the relation $\mathcal{R}$ is taken to be in a general form. If R is chosen to be a more specific relation (for instance, mutually exclusive events), more specific results will be derivable about the construction of an IBC partition.

For multi-agent systems, due to the huge state space of the product machine, in addition to the hierarchical control presented here, a hierarchical-decentralised control is worth further investigation. In the proposed configuration, each of a set of low controllers works on each of the agents, while a high level controller co-ordinates the operations of low level controllers in a feedback fashion. This will in turn shed some light on how to partition the state space of each agent separately in order to obtain an IBC partition of the product system.

6.3. Research Related to Network Routing

In Chapter 5, we discussed both link network systems and buffer network systems separately. A possible duality of these two classes of networks should be formulated and analysed. Moreover, the more complicated systems which combine the constraints on both links and buffers are of course deserve further investigation.

In real-time network. the information exchange sometimes is insufficient for the precise update of the network states. In the presence of uncertainty. the problem poses itself as to how an improved IHADP should be formulated achieve an applicable routing reliably.

Because of the multi-agent nature of the network. the concurrency of events is inevitable. In practice. a set of protocols are applied to each node in a network. This is a version of decentralised control. Through the analysis with the models proposed in Chapter 5. one of the objectives might be to find a set of protocols which combine the decentralised and hierarchical controls to efficiently decide the route for a request at a node dependent on the current network state.

REFERENCES

[1] R. Alur and D. Dill. *A theory of timed automata.* Theoretical Computer Science **126** (1994). 183–235.

[2] R. Alur and T.A. Henzinger. *Computer-aided verification.* Lecture Notes. Depatment of Electrical Engineering and Computer Science. UC Berkeley. 1996.

[3] P.J. Antsaklis. K. M. Passino. and S.J. Wang. *A system and control theoretic perspective on artificial intelligence planning systems.* Applied Artificial Intelligence **3** (1989). 1–32.

[4] J.J. Arrow and L. Hurwitz. *Decentralization and computation in resource allocation.* Essays in Economics and Econometrics (R.W. Pfouts. ed.). University of North Carolina Press. 1960.

[5] E. Asarin. O. Maler. and A. Pnueli. *Reachability analysis of dynamical systems having piecewise-constant derivatives.* TCS **138** (1995). no. 1. 35–66. Special issue on hybrid systems.

[6] G.R. Ash. *Design and control of networks with dynamic nonhierarchical routing.* IEEE Communication Magazine (October. 1990). 34–40.

[7] J.C. Bean. J.R. Birge. and R.L. Smith. *Aggregation in dynamic programming.* Operations Research **35** (1987). no. 2. 215–220.

[8] D. Bertsekas and D. Castanon. *Adaptive aggregation methods for infinite horizon dynamic programming.* IEEE Transactions on Automatic Control **34** (1989). no. 6. 589–598.

[9] B.A. Brandin and W.M. Wonham. *Supervisory control of timed discrete-event systems.* IEEE Transactions on Automatic Control **39** (1994). no. 2. 329–42.

[10] M.S. Branicky. *Topology of hybrid systems.* Proceedings of the 32nd IEEE Conference on Decision and Control (St.Antonio. Texas). 1993. pp. 2309–2314.

[11] M.S. Branicky, V.S. Borkar, and S.K. Mitter. *A unified framework for hybrid control.* Proceedings of the 33rd IEEE Conference on Decision and Control (Lake Buena Vista, FL), 1994, pp. 4228–4234.

[12] Y. Brave and M. Heymann. *Control of discrete event systems modeled as hierarchical state machines.* IEEE Transactions on Automatic Control **38** (1993), no. 12, 1803–1819.

[13] P. E. Caines, P. Hubbard, and G. Shen. *A state aggregation approach to hierarchical supervisory control with applications to a transfer line example.* Proc. of the 36th IEEE Conf. on Decision and Control (San Diego, FL), December 1997, pp. 3590–3591.

[14] P.E. Caines, V. Gupta, and G. Shen. *The hierarchical control of ST finite state machines.* Proceedings of the 36th IEEE Conference on Decision and Control (San Diego, CA), 1997, pp. 3584–3589.

[15] ———. *The hierarchical control of ST finite state machines.* Systems and Control Letters **32** (1997), 185–192.

[16] P.E. Caines and E. Lemch. *On the global controllability of hamiltonian and other nonlinear systems: fountains and recurrence.* Proceedings of the 37th IEEE Conference on Decision and Control (Tampa, FL), 1998, pp. 3575–3580.

[17] P.E. Caines and Y-J. Wei. *The hierarchical lattices of a finite machine.* Systems and Control Letters (1995), 257–263.

[18] ———. *On dynamically consistent hybrid systems.* Proceedings of the 1994 Cornell University Workshop on Hybrid Systems & Autonomous Control (NYC) (W. Kohn and A. Nerode, eds.), Lecture Notes in Computer Science, Springer Verlag, 1995.

[19] P.E. Caines and Y-J Wei. *Hierarchical hybrid control systems.* Proceedings of the Block Island Workshop of Control Using Logic-Based Switching (S. Morse, ed.), LNCIS 222, Springer-Verlag, 1996, pp. 39–48.

[20] P.E. Caines and Y-J. Wei. *Hierarchical hybrid control systems: A lattice theoretic approach.* IEEE Transactions on Automatic Control (37(4), 1998), 501–8.

[21] C. Cassandras. *Discrete event systems: Modeling and performance analysis.* Irwin Publ., 1993.

[22] B.A. Davey and H.A. Priestley. *Introduction to lattice and order.* Cambridge University Press, 1990.

[23] R. Diestel. *Graph theory*. Springer. 1997.

[24] U. Dudley, *Elementary number theory*. W.H. Freeman and Company. 1969.

[25] J. R. Evans and E. Minieka. *Optimization algorithms for networks and graphs*. Marcel Dekker Inc., New York. 1993.

[26] A. Gibbons. *Algorithmic graph theory*. Cambridge University Press. 1985.

[27] H. Haken. *The intractability of resolution*. Theoretical Computer Science **39** (1985). 297–308.

[28] F. Harary. R.Z. Norman. and D. Cartwright. *Structural models: An introduction to the theory of directed graphs*. John Wiley & Sons. New York. 1965.

[29] D. Harel. *Statechart: A visual formalism for complex systems*. Science of computer programming **8** (1987). 231–274.

[30] Y. C. Ho and S. K. Mitter (eds.). *Directions in large-scale systems: Many - person optimization and decentralized control*. Plenum Press. 1976.

[31] J.E. Hopcroft and J.D. Ullman. *Introduction to automata theory. languages. and computation*. Addison-Wesley. 1979.

[32] P. Hubbard and P. E. Caines. *Trace-DC hierarchical supervisory control with applications to transfer-lines*. Proceedings of the 37th IEEE Conference on Decision and Control (Tampa. Florida). 1998. pp. 3293–98.

[33] P. Hubbard and P.E. Caines. *Initial investigations of hierarchical supervisory control for multi-agent systems*. submitted to the 38th IEEE CDC. Pheonix. AZ. 1999.

[34] K.M. Inan and P.P. Varaiya. *Algebras of discrete event models*. Proceedings of IEEE **77** (1989). no. 1. 24–38.

[35] P. B. Key and G. A. Cope. *Distributed dynamic routing schemes*. IEEE Communications Magazine (1990). 54–64.

[36] K. Krishnan. *Markov decision algorithms for dynamic routing*. IEEE Communications Magazine (1990). 66–69.

[37] R. Kumar and V. Garg. *Optimal supervisory control of discrete event dynamical systems*. SIAM Journal on Control and Optimization **33** (1995). no. 2. 419–39.

[38] B. Lennartson. M. Tittus. B. Egardt. and S. Pettersson. *Hybrid systems in process control*. IEEE Control Systems (1996). 45–56.

[39] E. Leonardi, F. Neri, M. Gerla, and P. Palnate, *Congestion control in asychronous, high-speed wormhole routing networks*, IEEE Communication Magazine (1996), 58–69.

[40] Y. Li and W.M. Wonham, *Concurrent vector discrete-event systems*, IEEE Trans. on Automatic Control **40** (1995), no. 14, 628–38.

[41] F. Lin, A.F. Vaz, and W.M. Wonham, *Supervisor specification and synthesis for discrete event systems*, International Journal of Control **48** (1988), no. 1, 321–332.

[42] F. Lin and W.M. Wonham, *Decentralized control and coordination of discrete-event systems with partial observation*, IEEE Trans. on Automatic Control **35** (1990), no. 12, 1330–1337.

[43] J. Lygeros, D. Godbole, and S. Sastry, *Verified hybrid controllers for automated vehicles*, IEEE Transactions on Automatic Control **43** (1998), no. 4, 522–539.

[44] M.D. Mesarović, D. Macko, and Y. Takahara, *Theory of hierarchical, multilevel systems*, Academic, New York, 1970.

[45] G.L. Newhauser, *Introduction to dynamic programming*, John Wiley and Sons, 1966.

[46] K.M. Passino and Ümit Özgüner, *Modelling for hybrid systems: examples*, Proceedings of the 1991 IEEE International Symposium on Intelligent Control (Arlington, VA), 1991, pp. 251–256.

[47] P.J. Ramadge and W.M. Wonham, *The control of discrete event systems*, Proceedings of IEEE **77** (1989), no. 1, 81–97.

[48] J. Regnier and W.H. Cameron, *State-dependent dynamic traffic management for telephone networks*, IEEE Communications Magazine (10.1990), 42–53.

[49] K. Rudie and W.M. Wonham, *Think globally, act locally: Decentralized supervisory control*, IEEE Trans. on Automatic Control **37** (1992), no. 11, 1692–1708.

[50] M. Sampath, S. Lafortune, and D. Teneketzis, *Active diagnosis of discrete event systems*, The 36st IEEE Conference on Decision and Control (San Diego, CA), 1992, pp. 2976–83.

[51] R. Sengupta and S. Lafortune, *An optimal control theory for discrete event systems*, SIAM Journal on Control and Optimization **36** (1998), no. 2, 488–541.

[52] S. Sethi and Q. Zhang, *Multilevel hierarchical decision making in stochastic marketing-production systems*, SIAM J. Control and Optimization **33** (1995), no. 2, 528–553.

[53] G. Shen and P. Caines, *Hierarchically accelerated dynamic programming for discrete event systems*, submitted to IEEE Transactions on Automatic Control, 1999.

[54] ———, *On the application of hierarchically accelerated dynamic programming to multi-agent networks*, Technical Report, Department of Electrical and Computer Engineering, McGill University, 1999.

[55] G. Shen and P. E. Caines, *Control consistency and hierarchically accelerated dynamic programming*, Proceedings of the 37th IEEE Conference on Decision and Control (Tampa, Florida), 1998, pp. 1686–91.

[56] ———, *Hierarchically accelerated dynamic programming with applications to transportation networks*, Proceedings of the 14th World Congress of IFAC (Beijing, China), vol. L, 1999, pp. 285–290.

[57] J. A. Stiver, P.J. Antsaklis, and M.D. Lemmon, *Digital control from a hybrid perspective*, Proceedings of the 33rd IEEE Conference on Decision and Control (Lake Buena Vista, FL), 1994, pp. 4241–4246.

[58] A.S. Tanenbaum, *Computer networks*, Prentice Hall, 1989.

[59] J.G. Thistle, *Logical aspects of control of discrete event systems: a survey of tools and techniques*, 11th International Conference on Analysis and Optimization of Systems (Sophia-Antipolis) (Guy Cohen and Hean-Pierre Quadrat, eds.), 1994, Lecture Notes in Control and Information Sciences, 199, pp. 3–15.

[60] ———, *Supervisory control of discrete event systems*, Mathl. Comput. Modelling **23** (1996), no. 11/12, 25–53.

[61] J.G. Thistle, R.P. Malhamé, H.-H. Hoang, and S. Lafortune, *Supervisory control of distributed systems part I: Modelling, specification and synthesis*, Research report, École Polytechnique de Montréal, 1997.

[62] M. Tittus and K. Akesson, *Deadlock avoidance in batch processes*, IFAC World Congress 1999 (Beijing, China), 1999, A-1c-05-5.

[63] P.P. Varaiya, *Smart cars on smart roads: problems of control*, IEEE Trans. on Automatic Control **38** (1993), no. 2, 195–207.

[64] F. Wang and G.N. Saridis. *A coordination theory for intelligent machines*. Automatica **26** (1990), no. 5, 833–844.

[65] Y-J Wei. *Logic control: Markovian fragments, hierarchy and hybrid systems*. Ph.D. thesis, Department of Electrical Engineering, McGill University, Montreal, Canada, October, 1995.

[66] Y-J. Wei and P.E. Caines. *Hierarchical COCOLOG for finite machines*. Proceedings of the 11th INRIA International Conference on the Analysis and Optimization of Systems (Sophia Antipolis, France) (A. Bensoussan and J. Lions, eds.), Lecture Notes in Control and Information Sciences, vol. 199, INRIA, Springer Verlag, June, 1994, pp. 29–38.

[67] R. Wolf, P. Chemouil, and K. Mase. *Advanced traffic control methods for circuit-switched telecommunications networks*. IEEE Communications Magazine (1990), 12-14.

[68] K.C. Wong and W.M. Wonham. *Hierarchical and modular control of discrete-event systems*. Proc. of Thirtieth Annual Allerton Conference on Communication, Control and Computing, September, 1992, pp. 614-623.

[69] K.C. Wong and W.M. Wonham. *Hierarchical control of timed discrete-event systems*. Discrete event dynamic systems **6** (1996), no. 3, 275-306.

[70] H. Zhong and W.M. Wonham. *On the consistency of hierarchical supervision in discrete-event systems*. IEEE Trans. on Automatic Control **35** (1990), no. 10, 1125-1134.

APPENDIX A

C++ Header File

The application of the HADP methodology to the broken Manhattan grid systems (see Section 3.7) is realised by C++ code. Its header file is listed below.

```
#include"iostream.h"
#include"fstream.h"
#include <math.h>
#include <time.h>
#include <sys/times.h>
#include <sys/types.h>
#include <stdlib.h>

const int MAXEDGES=50;
const int BOUNDARYNODES=60;
const int MAX=2147483647;
```

The maximal number of edges starting from one node is set to be 50. and the maximal number of in-set nodes of a block with respect to a DC relation is set to be 60. In other words. for any $X_i \in \pi$. we assume there are no more than 50 blocks $X_j \in \pi$ such that each $\langle X_i, X_j \rangle$ is DC. We also assume $|I_i(X_j)| \leq 60$.

```
struct Path{
       int current_node;
       Path *next_node;
}path;
struct Node{
       int label;
       int number_of_edges;
       int edge[MAXEDGES];      //the index of an immediate successor
       float cost[MAXEDGES];       //D+/- cost for high level nodes
       float Mcost[MAXEDGES];      //D+ cost for high level nodes
       float mcost[MAXEDGES];      //D- cost for high level nodes
}node;
struct Block
{      int number_of_nodes;
       Node node[100];
       int number_of_insets;      //the number of predecessor DC blocks
       int inset[MAXEDGES];      //the index of a predecessor DC block
       int number_of_innodes[MAXEDGES];
       int innode[MAXEDGES][BOUNDARYNODES];
       int number_of_transitions;      //the number of successor DC blocks
       int transition[MAXEDGES];      //the index of a successor DC block
       int number_of_outnodes[MAXEDGES];
       int outnode[MAXEDGES][BOUNDARYNODES];
}block;
```

A block is an array of nodes. A structure Block also bears the information about the DC relations (transition) and the in-set. out-set nodes of a block.

```
int Rand(int);     //generate a random integer between 0 and int


//************************************************************
//          Grid:  a size*size array of vertices
//          vertex[i]=0:  it is removed
//************************************************************
class Grid
{
        public:
            int size;
            int *vertex;
        public:
            Grid(int length)
            { size=length; vertex=new int[size*size]; }
            void Create_Random_Grid(float);
};
//********************************************************************
//          (directed) Graph:  an array[number_of_nodes] of nodes
//          node[i]:  absolute index in the Grid
//*******************************************************************
class Graph
{
        public:
            int number_of_nodes;
            Node *node;
        public:
            Graph(int);      //constructor
            void Copy_Graph(Graph);      // copy a graph
            Graph(Grid);      //turn from a random grid
             Graph();
};
```

A grid is an array of integers. for which 1 means that a vertex exists. A broken grid is created by randomly removing a certain percentage of nodes (Create_Random_Grid (float)) from a regular grid. Then this broken grid is converted into a graph (Graph(Grid)).

```
//*******************************************************************
//          PartitionGraph:  a Graph used to generate a partition
//          first generate an array of constraint (a set of nodes)
//          then find IBC isolated blocks in each constraint
//          node[i].in_which_block:  the index of the block
//                containing this node
//*******************************************************************
class PartitionGraph :  public Graph
{
        public:
            int *in_which_block;
            int number_of_blocks;
            int *in_which_constraint;
            int *constraint;
            int nodes_in_constraint;
        public:
            PartitionGraph(Graph,int,int);
                  // int, int :  number of constraints, Grid.size
            PartitionGraph(int);      //int :  Graph.number_of_nodes
            void Partition(int,int);
             // int, int :  Grid.size, number of constraints
        void Find_IBC(int);      //int :  seed
    // find an IBC block containing the seed in the current constraint
        void Label_Predecessors(int);     //int :  seed
};
```

To obtain an IBC partition of a graph, we first partition this graph into a given number of regions (specified by constraints). Then, within every region, we randomly and sequentially choose seed nodes which are used for the construction of maximal IBC blocks containing them in the remaining region. We next label every node with the block index (in_which_block) which corresponds to the index of the IBC block containing it.

```
//**********************************************************************
//          WorkGraph:  a Graph used to calaculate shortest distances
//          label[node_index] used in the process of Dijkstra's algorithm
//                 to determine whether a label is permanent
//**********************************************************************
class WorkGraph :  public Graph
{       public:
            int *distance;
        protected:
            int *label;
            // label:  -1, unlebelled 0, temporary 1:  permanent
        public:
            WorkGraph(Graph);     //turn a Graph into a WorkGraph
            WorkGraph(int );     //create a WorkGraph with int nodes
            Path *Shortest_Path(int,int);     //(int int):  (source target)
            void Shortest_Distance(int, int);     //labelling process
            int Distance(int,int);     //return the shortest distance
};
```

Dijkstra's algorithm is used to find the shortest distance between a pair of start and target nodes (value returned by Distance(int,int)). A shortest path is then given by Shortest_Path(int,int).

```
//******************************************************************
//          TransferGraph:  an array[number_of_blocks] of blocks
//          based on a PartitionGraph, find in-block structure
```

```
//******************************************************************
class TransferGraph
{
        public:
            int number_of_blocks;
            Block *block;
        public:
            TransferGraph(int);      //int :  number_of_blocks
            void TransferBlock(PartitionGraph);
            int AdjustBlock();      //turn indices into local
            int Number_of_Highernodes();
            //return the number of high level nodes in case of semidual
            WorkGraph BlockGraph(int);
            //int :  the number of nodes in a block
            //turn a block into a WorkGraph to calculate costs
};
```

In `TransferGraph`, we record the in-block structures.
`PartitionGraph::Partition(int,int)` only labels each node with the name of the block containing it.
In `TransferBlock(PartitionGraph)`, the internal structure of a block (in-set/out-set) is set up by `TransferBlock(PartitionGraph)`.

The calculation of D^{+}, D^{+-} and D^{-} costs of the high level transitions is performed by `HigherGraph::Costs()`.

```
//******************************************************************
//          HighPartitionGraph:  used to partition a high level
//          difference from PartitionGraph:  how to decide a block
//                in which high level constraint
//******************************************************************
class HighPartitionGraph :  public PartitionGraph
{       public:
            HighPartitionGraph(TransferGraph,int,int);
             //int, int :  Grid.size, number of high level constraints
};
//*********************************************************************
// HigherGraph:  graph-like structure of middle/highest levels
// block_from/block_to :  indices of a pair of DC blocks
```

```
//*********************************************************************
class HigherGraph :  public WorkGraph
{
        public:
            int *block_from;
            int *block_to;
        public:
            HigherGraph(TransferGraph);
            void Costs(TransferGraph,int,int,int,int,int);
            //cost from I_i(X_j) to I_j(X_k)
            //j:  index of the current block, (i,j):  index of this node
//(j,k):  index of this edge, i:  from which block, k:  to which block
            Block HighBlockGraph(int, PartitionGraph, TransferGraph,
                  WorkGraph,int, int);
            TransferGraph HighTransferGraph(PartitionGraph,TransferGraph,
            WorkGraph, int, int);
            WorkGraph Modify_Graph(TransferGraph,int,int,int,int);
            //recalculate costs concerning Xs and Xt
            Path *Find_High_Path(TransferGraph,WorkGraph,
                  WorkGraph,int, int, int,int);
                  //int,int,int,int:  Xs, Xt, s, t
            Path *Find_Low_Path( TransferGraph, Path *, int,int);
                  //int, int:  s, t
            Path *Low_Path(TransferGraph,int,int,int,int &);
             //int,int,int,int:block_from, block_to, s, &t(next s)

            void GraphFile(); //write the graph into dot file
};
```

The conversion from a high level graph (an array of Blocks) to its semi-dual graph is carried out by HigherGraph(TransferGraph). For every node in a semi-dual, there are two correspondent integers: block_from and block_in, which together represent a pair of DC blocks.

Document Log:

Manuscript Version 0 — 28 August 1999
Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX — 28 August 1999

GANG SHEN

CENTER FOR INTELLIGENT MACHINES, MCGILL UNIVERSITY, 3480 UNIVERSITY ST., MONTRÉAL (QUÉBEC) H3A 2A7, CANADA

E-mail address: `shg@cim.mcgill.ca`

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX