Directed control of discrete event systems

Jing Huang
Iowa State University

Directed control of discrete event systems

by

Jing Huang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Program of Study Committee:
Ratnesh Kumar, Major Professor
Nicola Elia
Chris Chong-Nuen Chu
Samik Basu
Robert J. Weber

Iowa State University

Ames, Iowa

2009

DEDICATION

To my parents, who give me so much and ask for so little.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my cordial thanks to my major professor, Dr. Ratnesh Kumar, for his guidance. His insights and devotion to work have inspired me and driven me to achieve one milestone after another. I extend special thanks to him for introducing me to this interesting topic of discrete event systems and teaching me the art of scientific writing. I am also grateful to him for providing financial support during my graduate years at Iowa State University.

I wish to convey my sincere thanks to the other members of my Program of Study committee: Dr. Nicola Elia, Dr. Chris Chong-Nuen Chu, Dr. Samik Basu, Dr. Robert J. Weber and Dr. Murti V. Salapaka, for dedicating their time to participate in the committee. Their feedback and comments were very helpful in improving the work presented in this dissertation.

I also cherish the friendship I made along the way, with those working in the same office: Changyan Zhou, Wenbin Qiu, Haifeng Liu, Licheng Jin, Qin Wen, Songyan Xu, Saayan Mitra, and many others. They made my journey colorful and enjoyable.

Last but not least, I thank my parents for their love, education, care and support, without which I would not have been able to complete this work.

ABSTRACT

For the control of discrete event systems, the notion of directed control refines that of supervisory control. A directed controller is one that selects at most one controllable event to be enabled at any state (without disabling any uncontrollable event), which is in fact how a discrete event control is implemented. In contrast, a supervisory controller computes a maximal allowable set of controllable events at each state, leaving undecided exactly which one is to be enabled.

We model discrete event systems using the automaton formalism. Under directed control, our first goal is to achieve logical correctness of the controlled system behavior as specified by safety and nonblocking. Subsequently we address the best performance issue by providing an optimization based framework. The optimization task is to direct a system in such a way that regardless of the history of evolution, it accomplishes a pending task in a minimal cost.

In a state-based setting, we formulate and study the existence and synthesis problems with the above objectives. We first show that the existence and the synthesis of a safe and nonblocking directed controller are both solvable in polynomial complexity. Then we present a novel approach with polynomial complexity for the synthesis (and the existence) of an optimal director, thus providing a complete solution to the problems in study.

CHAPTER 1. OVERVIEW

1.1 Introduction - From supervisor to director

For the control of discrete event systems, the notion of directed control refines that of supervisory control. A directed controller, simply referred to as *director*, is one that selects at most one controllable event to be enabled at any state (without disabling any uncontrollable event), which is in fact how a discrete event control is implemented. This is in contrast to *supervisory control* [21][15][8], where a supervisory controller, simply referred to as *supervisor*, computes a maximal allowable set of controllable events at each state, leaving undecided exactly which one is to be enabled.

Most prior work on logical control of discrete event systems deals with supervisory control. Only a few exceptions exist, e.g., the notion of "forcing control" [2][13][5][18]. A problem with the notion of forcing control is that through forcing, one is able to preempt the other events including any feasible uncontrollable events. In the timed setting, the forcing is also able to preempt the "tick" transition of an underlying discrete clock. A director, on the other hand, does not preempt feasible uncontrollable events, rather restricts the number of enabled controllable events to be at most one. The feasible uncontrollable events remain enabled. Another exception can be found in the framework of prioritized synchronization based supervisory control [14]. Under that framework, there is an added provision of driven events, which are events that a supervisor can execute with or without the participation of the system under control.

The design of a supervisor is meaningful for applications in which the plant is an autonomous *generator* of controllable events. However for many applications, the plant is an *executor* of controllable events, i.e., it does not autonomously generate such events, rather executes them when commanded by a controller.

In a transportation system, for example, a supervisory control action will specify a maximal set of permissible routes for a vehicle. However, what is more appropriate is a directed control action commanding the vehicle to follow a specific route. Similarly, while it may be legal to permit a certain conveyor motor to rotate forward as well as reverse, in an application the direction of rotation must be specified.

So for systems that are executor of events, it is more meaningful to issue a command consisting of at most one possible controllable event, rather than a set of controllable events as issued by a supervisor.

In [3], an antenna rotor control system (ARCS) has been designed where a controller enforces the given safety, liveness, and real-time control constraints, while selecting a single controllable event at each state of the system, i.e., the controller is a director. The controllable event is selected from the ones allowed by a maximally permissive supervisor, but on an *ad hoc* basis. Similar ad hoc selection of controllable events is made in another application consisting of an educational assembly line [9]. In [11], the authors pointed out some main issues facing the implementation of supervisors on programmable logic controllers (PLCs), one of which is the need to choose one controllable event among alternatives. The suggested solution, however, is either making the choice explicitly at the compile time, or letting a PLC choose one based on the ordering of its rungs at the run time.

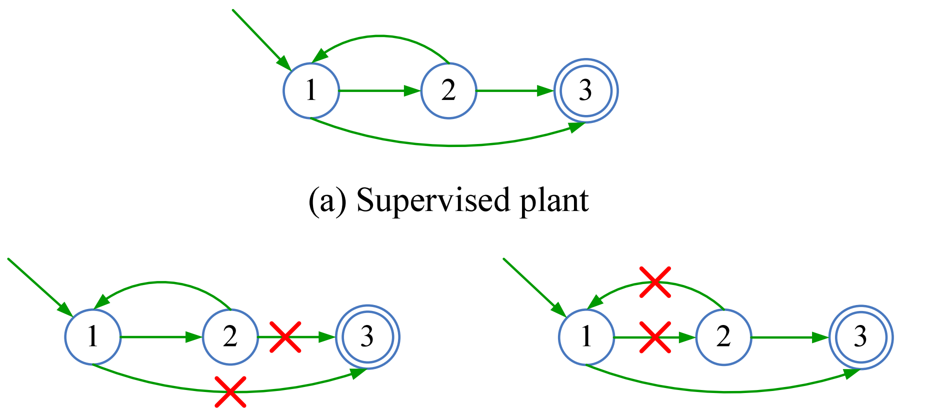
The problem of computing a director can also be viewed as a generalization of the classical planning problem considered in the AI community [12][10], where the objective is to compute a plan, which is a map that selects control actions at each state, so as to steer the system from any of the given initial states to the desired final or goal states. In that setting, the notion of uncontrollability of events is missing, so our setting of directed control is more general. Thus the approach developed in the work extends the scope of AI-planning problems to the settings where uncontrollable events are also present.

1.2 Objectives - From logic correctness to best performance

We model a discrete event system to be controlled, called a plant, using the automaton formalism. The control goal under directed control can be the same as that in the setting of supervisory control, e.g., logical correctness as specified by safety and nonblocking. We refer to safety as the system property that something “bad” will never happen, i.e., any undesirable or illegal state will never be reached from the initial state or other accessible states under directed control. In contrast, we refer to nonblocking as the system property that something “good” will eventually happen, i.e., any accessible state in the directed controlled plant, or simply directed plant, is extendable to a marked or final state.

While it is possible to arbitrarily select one of the controllable events among the ones enabled by a supervisor to “extract” a director, such an ad hoc selection can lead to blocking, i.e., a generated trace of the directed plant is not extendable to a marked or final state. For example, consider a plant under the control of a supervisor shown in Figure 1.1(a). An arbitrary disablement of all but one controllable event to obtain a director may result in blocking, as shown in Figure 1.1(b). On the other hand, another way of disabling all but one controllable event results in a nonblocking directed plant, as shown in Figure 1.1(c). Thus it is clear that one needs an algorithmic approach to search for a director.

After a system has been controlled so as to ensure proper behaviors, it is natural and logical to address the best performance issue. We formulate an optimization based framework for such an objective. Note that there may be many controls that are able to guarantee the proper system behavior whereas usually only select few, if not only one,



(a) Supervised plant

(b) Blocking directed plant (c) Nonblocking directed plant

Figure 1.1 From supervisor to director

is able to achieve the optimality.

Under such a framework, we define a cost associated with each event that is a function of the trace it follows. In practice, this cost will typically depend on a bounded history of executed events. For an uncontrollable event, this cost represents the cost of executing the event and the payoff of reaching the resulting state, whereas for a controllable event, this represents the cost of executing the event and the payoff of reaching the resulting state, *together with* the cost of disabling other feasible controllable events. Thus a single cost function is able to capture both the "path cost" and the "control cost" [23][24]. The cost function can represent completion time, production cost, etc.

The optimization task is to direct a system in such a way that regardless of the history of evolution, it accomplishes a pending task in a minimal cost. In the absence of uncontrollable events, this amounts to finding the shortest cost path between any state and its nearest reachable marked states. In the presence of uncontrollable events, control is exercised in a manner that the worst cost over all surviving paths between any state and its nearest reachable marked states is minimized.

In a flexible manufacturing system, for example, products need to be assembled in an optimal way using different permutations of the resources available. The inherent uncertainties associated with the manufacturing process such as breakdowns can be modeled as uncontrollable events, and the framework developed here can be used to obtain an optimal control strategy.

Note that the director synthesized for logic correctness can be seen as a special case under such a framework. In other words, an optimal director should be safe and nonblocking.

1.3 Results - From existence to synthesis

Given a plant, with control objectives described above, we proceed to determine whether there exists a director satisfying our specifications, and if yes, then how to synthesize one.

Without loss in generality, we formulate and study the existence and synthesis problems in a state-based setting. In this setting the search space for nonblocking directors and optimal directors are both exponential in the number of plant states. Thus the formulated problems are indeed solvable. An exhaustive search, however, would have a complexity exponential in the size of plant. So computationally efficient approaches are desired.

We will first show that a safe and nonblocking director exists if and only if a safe and nonblocking supervisor exists, thereby proving the polynomiality of verifying existence. (Recall that existence of a safe and nonblocking supervisor is polynomially verifiable.) Then we will provide a set of algorithms of polynomial complexity to compute a safe and nonblocking director (whenever one exists).

Next we will show a necessary and sufficient condition for the existence of an optimal director. For systems that are cycle-free, we provide an algorithm of polynomial complexity to compute an optimal director, which is also proved to be dynamic-programming optimal.

Last we will present a novel approach for the synthesis (and the existence) of an optimal director for general plants. i.e., plants with or without cycles, thus providing a complete solution to the above problems, with the complexity of the approach remains polynomial in the size of plant.

CHAPTER 2. NOTION OF DIRECTED CONTROL

2.1 Notation and preliminaries

A DES to be controlled, called a plant, is modeled as an automaton, denoted as $G := (X, \Sigma, \alpha, x_0, X_m)$, where X denotes the set of states, Σ denotes the finite set of events, $\alpha : X \times \Sigma \to X$ denotes the partial deterministic state transition function and is extended in a natural way to $\alpha : X \times \Sigma^* \to X$, $x_0 \in X$ denotes the initial state, and $X_m \subseteq X$ denotes the set of marked states.

For $x \in X$, we use $\Sigma(x) \subseteq \Sigma$ to denote the set of events defined at x, i.e.,

$$\Sigma(x) := \{\sigma \in \Sigma \mid \alpha(x, \sigma) \text{ is defined}\}$$

Σ^* is used to denote the set of all finite-length sequences of events, called traces, which includes the zero-length trace ϵ. A subset of Σ^* is called a language. The generated language of G is defined as,

$$L(G) := \{s \in \Sigma^* \mid \alpha(x_0, s) \text{ is defined}\}$$

whereas the marked language of G is defined as,

$$L_m(G) := \{s \in L(G) \mid \alpha(x_0, s) \in X_m\}$$

$X_t := \{x \in X \mid \Sigma(x) = \emptyset\}$ is used to denote the set of all terminating states and $X_{tm} := X_t \cap X_m$ is used to denote the set of all *terminating marked* states. We use $L_t(G) := \{s \in L(G) \mid \alpha(x_0, s) \in X_t\}$ to denote the set of terminating traces of G.

$X(G)$ is used to denote the state set of G. A state $x \in X$ is called *accessible* if there exists a trace $s \in \Sigma^*$ such that $x = \alpha(x_0, s)$. A state $x \in X$ is called *coaccessible* to X_m, or simply coaccessible, if there exists a trace $s \in \Sigma^*$ such that $\alpha(x, s) \in X_m$. We denote the operation to delete the states in G that are not accessible (resp. coaccessible) as $Ac(G)$ (resp. $CoAc(G)$). An automaton G is called accessible (resp. coaccessible) if $G = Ac(G)$ (resp. $G = CoAc(G)$). An automaton G that is both accessible and coaccessible is said to be *trim*.

For a language $K \subseteq \Sigma^*$, the notation $pr(K)$, called the prefix-closure of K, denotes the set of all prefixes of traces in K. K is said to be prefix-closed if $K = pr(K)$. We use $K \backslash s$ to denote the set of traces that occur in the language K after the trace s has occurred, i.e., $K \backslash s := \{t \in \Sigma^* \mid st \in K\}$.

The notation $s \leq t$ is used to denote that the trace $s \in \Sigma^*$ is a prefix of the trace $t \in \Sigma^*$. When s is a proper prefix of t (i.e., $s \leq t$ and $s \neq t$), it is denoted as $s < t$.

For control purposes, the event set of G is partitioned into the set of controllable events $\Sigma_c \subseteq \Sigma$ and the set of uncontrollable events $\Sigma_u \subseteq \Sigma$. We define, $\Sigma_c(x) := \Sigma(x) \cap \Sigma_c$ and $\Sigma_u(x) := \Sigma(x) \cap \Sigma_u$.

The uncontrollable events are generally of two kinds: disturbance inputs and sensor outputs. An example of disturbance inputs is system failure and an example of sensor outputs is change in motion detector output. Occurrence of a disturbance input is uncertain while that of a sensor output is something can be expected. The set of uncontrollable events that are disturbance inputs is denoted as $\Sigma_d \subseteq \Sigma_u$. (The remaining uncontrollable events in $\Sigma_u - \Sigma_d$ are the sensor outputs.)

A supervisory controller is a map $S : L(G) \rightarrow 2^{\Sigma - \Sigma_u}$ that determines the set of events $S(s) \subseteq (\Sigma - \Sigma_u)$ to be disabled after the occurrence of a trace $s \in L(G)$. Events not belonging to the set $S(s)$ remain enabled at trace s. In particular, the uncontrollable events remain enabled. The supervised plant is denoted as G^S, and its generated and marked languages are defined using:

$\epsilon \in L(G^S)$

$[s \in L(G^S), s\sigma \in L(G), \sigma \notin S(s)] \Leftrightarrow [s\sigma \in L(G^S)]$

$L_m(G^S) := L(G^S) \cap L_m(G)$

It holds that $L_m(G^S) \subseteq L(G^S) = pr(L(G^S)) \neq \emptyset$. S is said to be nonblocking if $L(G^S) \subseteq pr(L_m(G^S))$. Given a nonempty specification language $K \subseteq L_m(G)$, there exists a nonblocking supervisor if and only if K is controllable, i.e., $pr(K)\Sigma_u \cap L(G) \subseteq pr(K)$, and relative-closed, i.e., $pr(K) \cap L_m(G) = K$.

2.2 Notion of directed control

A directed controller, or simply a director, enables *at most* one controllable event following each trace. This control selection is what distinguishes a director from a supervisor.

It should be noted that following certain traces, disabling all controllable events is not a good option. These consist of

- Traces $s \in L_m(G) - L_t(G)$ such that $L(G)\backslash s \cap \Sigma_u = \emptyset$. These traces are non-terminating marked traces and not followed by any uncontrollable events (so disabling all controllable events at such traces will block system from performing future tasks), and

- Traces $s \in L(G)$ such that $L(G)\backslash s \cap \Sigma_c \neq \emptyset$ and $\emptyset \subset L(G)\backslash s \cap \Sigma_u \subseteq \Sigma_d$. These traces are followed by at least one controllable event (so a control can be exercised at such traces) while all uncontrollable events occur after the traces are disturbance inputs and at least one such disturbance input is present (so disabling all controllable events will make the system wait for a disturbance input to occur in order to evolve further).

We use $L_e(G)$ to denote the set of traces mentioned above, i.e.,

$$L_e(G) := \{s \in L_m(G) - L_t(G) \mid L(G)\backslash s \cap \Sigma_u = \emptyset\} \cup$$
$$\{s \in L(G) \mid (L(G)\backslash s \cap \Sigma_c \neq \emptyset) \wedge (\emptyset \subset L(G)\backslash s \cap \Sigma_u \subseteq \Sigma_d)\}$$

Definition 1 *A director is a map $D : L(G) \rightarrow 2^{\Sigma_c}$ such that*

$$\forall s \in L(G) : |D(s)| \leq 1 \text{ and } \forall s \in L_e(G) : |D(s)| = 1$$

Following the execution of a trace $s \in L(G)$, the director enables at most one controllable event unless the trace belongs to $L_e(G)$, in which case the director enables exactly one controllable event. Also note that no control decision is defined with respect to uncontrollable events; such events remain enabled. Thus the set of events enabled by a director following a trace $s \in L(G)$ is given by $D(s) \cup \Sigma_u$.

The directed plant is denoted by G^D, and the languages generated and marked by the directed plant are denoted by $L(G^D)$ and $L_m(G^D)$ respectively, which are defined as follows:

$$\epsilon \in L(G^D)$$
$$[s \in L(G^D), \sigma \in D(s) \cup \Sigma_u, s\sigma \in L(G)] \Leftrightarrow [s\sigma \in L(G^D)]$$
$$L_m(G^D) := L(G^D) \cap L_m(G)$$

It should be clear that $pr(L_m(G^D)) \subseteq L(G^D)$. A director D is said to be nonblocking if $pr(L_m(G^D)) = L(G^D)$.

For simplicity, we will consider state-based specification and control. It is known that a language-based specification (resp. control) can be converted to a state-based specification (resp. control) on a suitably refined plant model. A state-based specification for safety, for example, is given using a set of illegal states $X_i \subseteq X$ that must never be visited, whereas a director D is state-based if it computes control action as a function of plant state, i.e., $D : X \rightarrow 2^{\Sigma_c}$.

Definition 2 *A state-based director is a map $D : X \to 2^{\Sigma_c}$ such that*

$$\forall x \in X : |D(x)| \leq 1 \textit{ and } \forall x \in X_e : |D(x)| = 1$$

where

$$X_e := \{x \in X_m - X_t \mid \Sigma_u(x) = \emptyset\} \cup \{x \in X \mid (\Sigma_c(x) \neq \emptyset) \wedge (\emptyset \subset \Sigma_u(x) \subseteq \Sigma_d)\}$$

Under the control of a state-based director D, the controlled plant is a subgraph of the plant graph, $G^D := (X, \Sigma, \alpha^D, x_0, X_m)$, where the state-transition function is defined as follows:

$$\forall x \in X, \sigma \in \Sigma : \alpha^D(x, \sigma) := \begin{cases} \alpha(x, \sigma) & \text{if } \sigma \in D(x) \cup \Sigma_u \text{ and } \alpha(x, \sigma) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

For a state-based director to be nonblocking, the following must hold: if $x \in X$ is such that there exists $s \in \Sigma^*$ with $\alpha^D(x_0, s) = x$, then there exists $t \in \Sigma^*$ such that $\alpha^D(x_0, st) \in X_m$.

CHAPTER 3. DIRECTED CONTROL FOR SAFETY AND NONBLOCKING

3.1 Introduction

As discussed above, the control goal under directed control can be logical correctness as specified by safety and nonblocking. In this chapter, we will show how we can achieve this goal with an algorithmic approach. We will begin with the notation and preliminaries specific to this topic, then present a set of algorithms to verify the existence and subsequently perform the synthesis of a safe and nonblocking director in time polynomial in the number of the plant states. Some examples will be provided to aid the understanding of the algorithms. We will conclude this chapter with an application example to demonstrate the results.

3.2 Notation and preliminaries

Definition 3 Given a plant $G := (X, \Sigma, \alpha, x_0, X_m)$, a component $(\hat{X}, \hat{\alpha})$ of G is a subgraph of G satisfying $\hat{X} \subseteq X$ and $\hat{\alpha} \subseteq \alpha|_{\hat{X}}$, where $\alpha|_{\hat{X}}$ denotes the transition function α restricted to the domain $\hat{X} \times \Sigma$.

The set of all possible directors for a component $(\hat{X}, \hat{\alpha})$ is denoted as $\mathcal{D}(\hat{X}, \hat{\alpha})$.

Central to our algorithm for synthesizing a nonblocking director is the observation that any given graph, including a controlled plant graph, can be partitioned into a set of strongly-connected components (SCCs) [25], over which the given graph possesses a tree structure. In case of a directed plant graph, the leaf nodes of such a tree must be *strongly-connected*, *legal*, *invariant*, *nonblocking* and *control-consistent* components ($SLINC$s) while the non-leaf nodes must be *strongly-connected*, *legal*, *$SLINC$-attractable* and *control-consistent* components ($SLAC$s). These notions are formalized as follows.

Definition 4 A component $(\hat{X}, \hat{\alpha})$ is called

1. *strongly-connected* if there exists a path lying entirely within the component between any pair of states of the component, i.e.,

$$\forall x_1, x_2 \in \hat{X}, \exists s \in \Sigma^* \text{ s.t. } \hat{\alpha}(x_1, s) = x_2 \text{ and } \forall t \leq s : \hat{\alpha}(x_1, t) \in \hat{X}$$

2. *legal* if there is no illegal state inside the component, i.e.,

$$\hat{X} \cap X_i = \emptyset.$$

3. *control-consistent* if there is at most one controllable event defined at each state and exactly one controllable event defined at each state belongs to X_e, i.e.,

$$\forall x \in \hat{X} : |\Sigma_c^{\hat{\alpha}}(x)| \leq 1 \text{ and } \forall x \in \hat{X} \cap X_e : |\Sigma_c^{\hat{\alpha}}(x)| = 1$$

   where

$$\Sigma_c^{\hat{\alpha}}(x) := \{\sigma \in \Sigma_c \mid \hat{\alpha}(x, \sigma) \text{ is defined}\}$$

4. *nonblocking* if from any state of the component, a marked state can be reached within the component, i.e.,

$$\forall x \in \hat{X}, \exists s \in \Sigma^* \text{ s.t. } \hat{\alpha}(x, s) \in X_m \text{ and } \forall t \leq s : \hat{\alpha}(x, t) \in \hat{X}$$

5. *invariant* if state transitions of $(\hat{X}, \hat{\alpha})$ can be confined within $\hat{X}$ under directed control, i.e.,

$$\hat{\alpha}(\hat{X}, \Sigma_u) \subseteq \hat{X} \text{ and } \forall x \in \hat{X} \cap X_e : \hat{\alpha}(x, \Sigma_c) \cap \hat{X} \neq \emptyset$$

Otherwise, the component is called *variant*. We represent the set of states in $\hat{X}$ violating any of the above conditions, i.e. the set of *variant* states, as $V(\hat{X}, \hat{\alpha}) \subseteq \hat{X}$.

6. X_r*-attractable*, where X_r is a reference state set, if there exists a component $(\widetilde{X}, \widetilde{\alpha}) \supseteq (\hat{X}, \hat{\alpha})$ such that every state of $(\widetilde{X}, \widetilde{\alpha})$ can reach a state of X_r over transitions within $\widetilde{X} \cup X_r$, and the state transitions of $(\widetilde{X}, \widetilde{\alpha})$ can be confined within $\widetilde{X} \cup X_r$ under directed control, i.e., $\exists(\widetilde{X}, \widetilde{\alpha}) \supseteq (\hat{X}, \hat{\alpha})$ such that

   (a) $\forall x \in \widetilde{X}, \exists s \in \Sigma^*$ such that $\widetilde{\alpha}(x, s) \in X_r$ and $\forall t < s : \widetilde{\alpha}(x, t) \in \widetilde{X}$, and

   (b) $\widetilde{\alpha}(\widetilde{X}, \Sigma_u) \subseteq \widetilde{X} \cup X_r$, and

   (c) $\forall x \in \widetilde{X} \cap X_e : \widetilde{\alpha}(x, \Sigma_c) \cap [\widetilde{X} \cup X_r] \neq \emptyset$.

   In case $(\widetilde{X}, \widetilde{\alpha})$ can be chosen as the component $(\hat{X}, \hat{\alpha})$ itself, $(\hat{X}, \hat{\alpha})$ is said to be *singularly* X_r*-attractable*. Also, the notion $U((\hat{X}, \hat{\alpha}), X_r) \subseteq \hat{X}$ is used to denote the set of states in $\hat{X}$ violating any of the 3 conditions in the definition of *singular* X_r-attractability.

Remark 1 The notion of attraction was introduced in [6] and is generalized in the above definition.

Definition 5 Given a component $(\hat{X}, \hat{\alpha})$ and a reference state set $X_r \subseteq X$, we represent the set of *maximal* sub-components of $(\hat{X}, \hat{\alpha})$ that are

1. strongly-connected as, $\mathcal{S}(\hat{X}, \hat{\alpha})$;

2. strongly-connected, legal, invariant and nonblocking as, $\mathcal{SLIN}(\hat{X}, \hat{\alpha})$;

3. strongly-connected, legal and singularly X_r-attractable as, $\mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$.

Example 1 We illustrate the above concepts in Figure 3.1, where we represent illegal states by crossing them. An edge with double arrows represents a transition on an uncontrollable event and an edge with single arrow represents a transition on a controllable event.

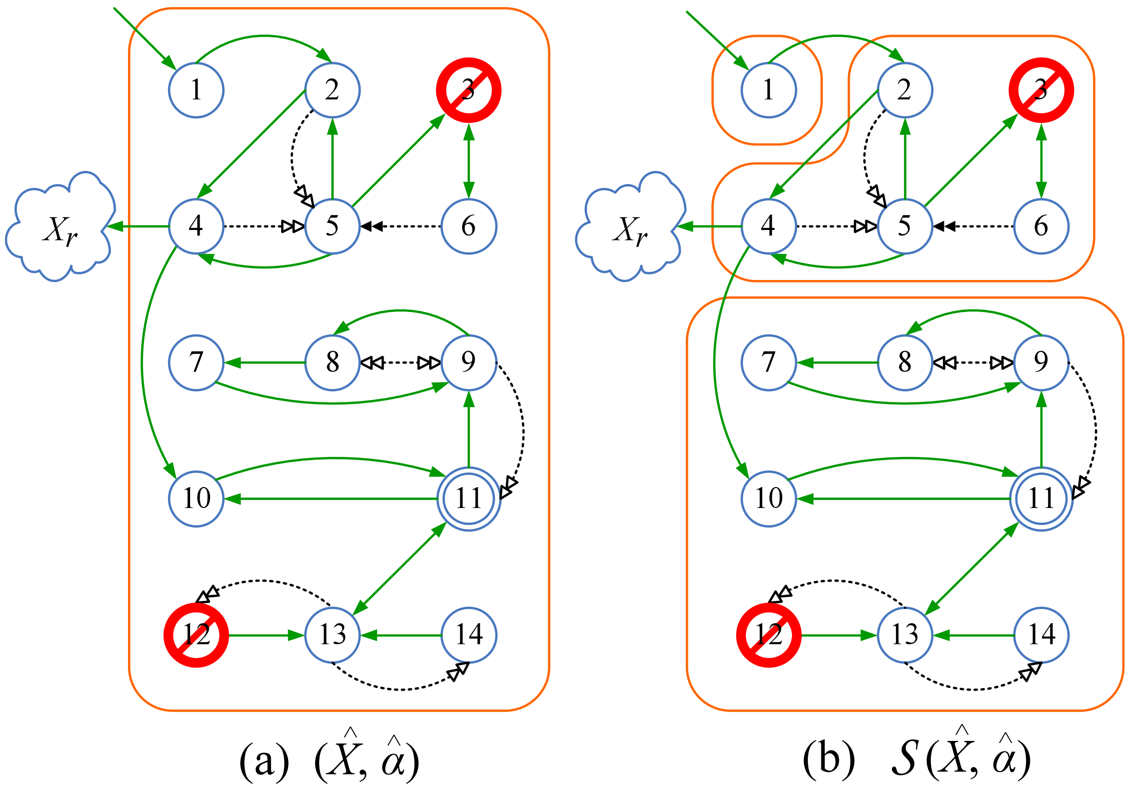
For the component $(\hat{X}, \hat{\alpha})$ shown in Figure 3.1(a), all its maximal SCCs are encircled in Figure 3.1(b). Figure 3.1(c) shows all maximal SCCs that are legal, invariant and nonblocking, whereas Figure 3.1(d) shows all maximal SCCs that are legal and singularly X_r-attractable.

Algorithmic computation of $\mathcal{S}(\hat{X}, \hat{\alpha})$ is well-known [25]; the algorithms to compute $\mathcal{SLIN}(\hat{X}, \hat{\alpha})$ and $\mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$ are presented in the Appendix for reference.

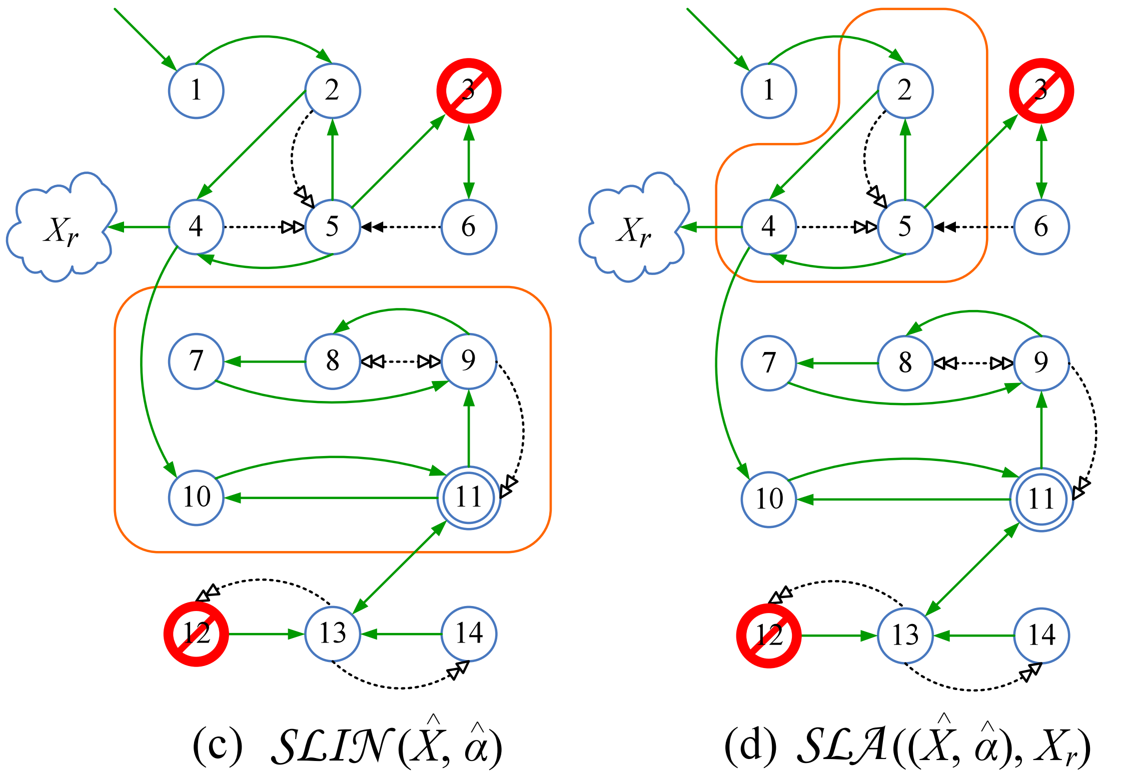
3.3 Existence of nonblocking director

Given a plant $G := (X, \Sigma, \alpha, x_0, X_m)$, the control goal is to find a state-based nonblocking director $D : X \rightarrow 2^{\Sigma_c}$ such that illegal states are never visited. It turns out that the existence of such a director can be determined by checking the existence of a nonblocking supervisor. Since a nonblocking supervisor exists if and only if a maximally permissive nonblocking supervisor exists [21], we first present an algorithm to compute such a supervisor, taken from [17]. Again the central idea is that the graph of a maximally permissive nonblocking supervised plant is partitionable into SCCs, over which it possesses a tree-structure (see Figure 3.2(a)). The leaf nodes of the tree are SCCs that are legal, invariant and nonblocking. Other SCCs are legal and attractable to leaf nodes.

The algorithm first identifies strongly-connected, legal, invariant and nonblocking components ($SLIN$s) as the leaf nodes, and then iteratively searches backward to identify strongly-connected, legal and $SLIN$-attractable components (SLAs) for non-leaf nodes. The iterative backward search terminates when either the *root* node (i.e., a $SLA/SLIN$ containing the initial state) is found, or no further SLAs can be added as nodes to the tree. In the former case, a nonblocking supervisor exists, whereas in the latter case, a nonblocking supervisor does not exist.



X_r
1
2
3
4
5
6
7
8
9
10
11
12
13
14
(a) $(\hat{X}, \hat{\alpha})$
(b) $\mathcal{S}(\hat{X}, \hat{\alpha})$



X_r
1
2
3
4
5
6
7
8
9
10
11
12
13
14
(c) $\mathcal{SLIN}(\hat{X}, \hat{\alpha})$
(d) $\mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$

Figure 3.1 Illustration of Definition 5

Algorithm 1 Given a plant graph (X, α) containing some marked and illegal states, the following steps computes a maximally permissive nonblocking supervisor [17].

1. Compute $\mathcal{SLIN}(X, \alpha)$. Let X_0 denote the set of states of all those components; $k = 0$.

2. (a) Compute singularly X_k-attractable region in the remainder of the plant, i.e., compute $\mathcal{SLA}((X - X_k, \alpha|_{X-X_k}), X_k)$; Let $\widetilde{X}$ denote the set of states of all those components.

   (b) Augment X_k with $\widetilde{X}$ to get X_{k+1}, i.e., $X_{k+1} = X_k \cup \widetilde{X}$.

3. Repeat Step 2 with $k = k + 1$ until

   (a) $x_0 \in X_k$, in which case a maximally permissive nonblocking supervisor is found, or

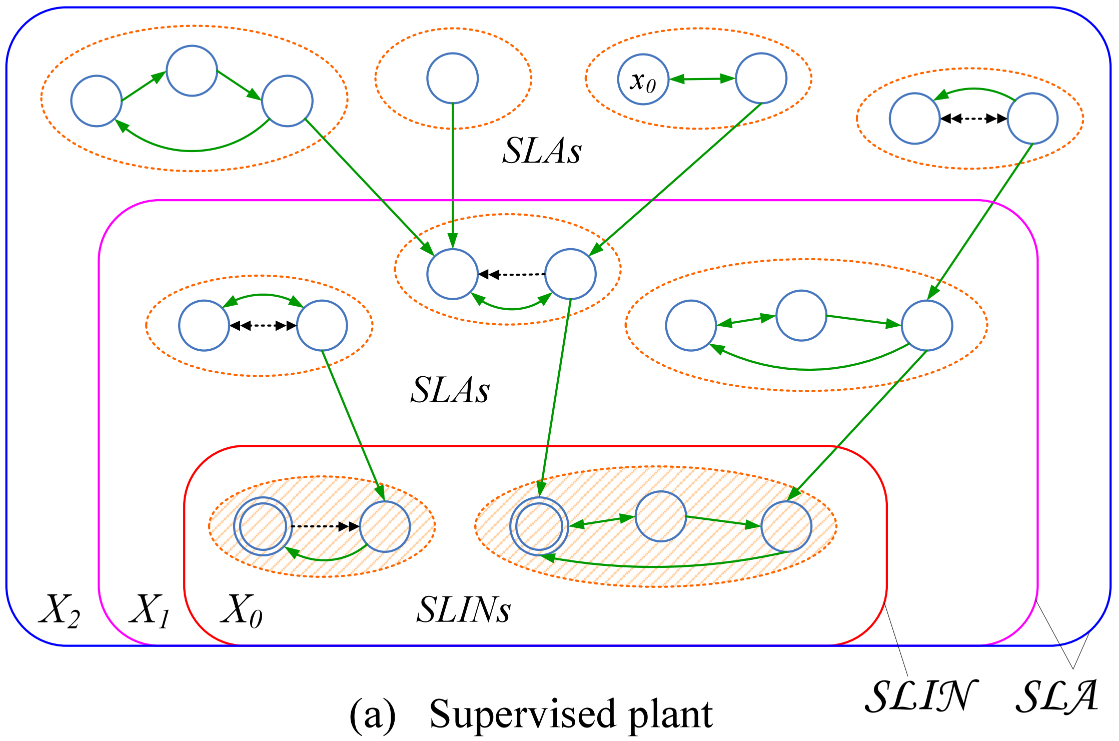
   (b) $X_{k+1} = X_k$, in which case no nonblocking supervisor exists.

In the remainder of this section, we show that the director existence is equivalent to the supervisor existence. We need the following lemmas to establish our existence result.

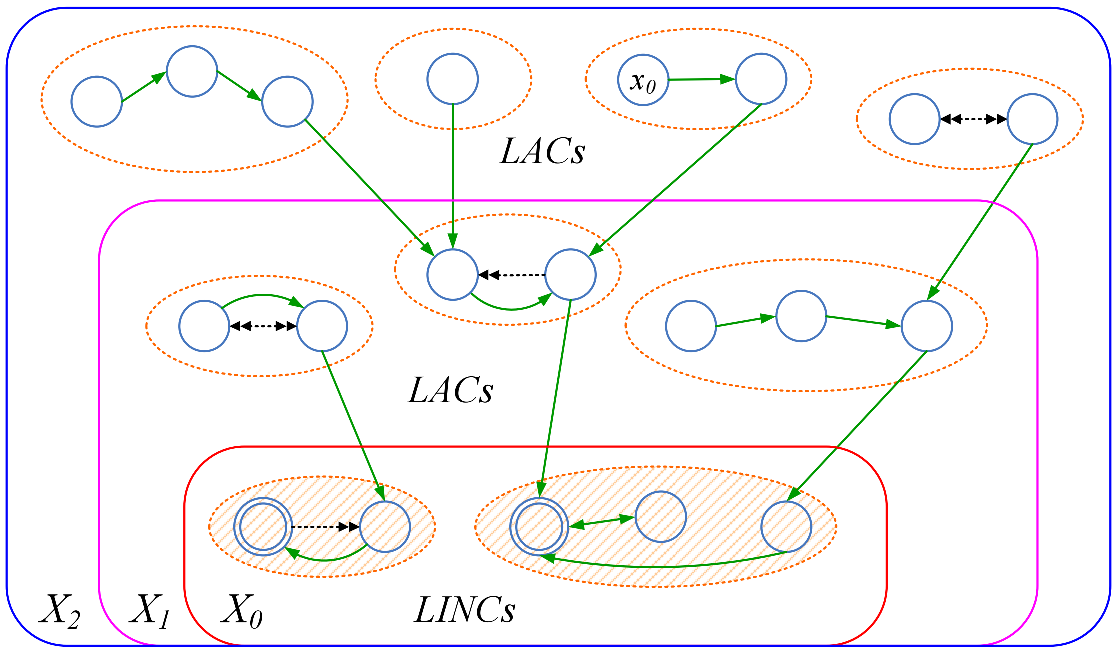
Lemma 1 For any strongly-connected, legal, invariant and nonblocking component $(\hat{X}, \hat{\alpha})$, there exists one $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, invariant, nonblocking and control-consistent.

Proof: Our proof obligation is to show there exists a director D for $(\hat{X}, \hat{\alpha})$ such that under the control of D, the resulting sub-component $(\hat{X}, \hat{\alpha}^D)$ remains legal, invariant and nonblocking. (Note that $(\hat{X}, \hat{\alpha}^D)$ may not be strongly-connected. Also $(\hat{X}, \hat{\alpha}^D)$ has the same state space as $(\hat{X}, \hat{\alpha})$.)

Clearly, for any $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$, $(\hat{X}, \hat{\alpha}^D)$ is legal and control-consistent.



(a) Supervised plant



(b) Directed plant

Figure 3.2 Structure of plants under control

Next, since $(\hat{X}, \hat{\alpha})$ is strongly-connected, we can partition $\hat{X}$ into a chain of disjoint state sets: $\hat{X} = \bigcup_{k=0}^{n} X_k$, where $X_0 := \hat{X} \cap X_m$ and $\forall 0 < k \leq n : X_k := \{x \in \hat{X} \mid \hat{\alpha}(x, \Sigma) \cap X_{k-1} \neq \emptyset\}$. In other words, X_k is composed of all the states that can reach X_{k-1} in one step. Note that since $(\hat{X}, \hat{\alpha})$ is nonblocking, $X_0 = \hat{X} \cap X_m$ is nonempty.

For $x \in X_0$, a director D enables any controllable event σ defined at x which makes transition inside the component $(\hat{X}, \hat{\alpha})$, i.e., $\hat{\alpha}^D(x, \sigma) \in \hat{X}$. Next for each state $x \in X_k$, there exists at least one $\sigma \in \Sigma$ such that $\hat{\alpha}(x, \sigma) \in X_{k-1}$. If this $\sigma \in \Sigma_c$, then the director D enables it; otherwise any controllable event σ defined at x which makes transition inside the component $(\hat{X}, \hat{\alpha})$ can be enabled by D. In this manner, the desired $(\hat{X}, \hat{\alpha}^D)$ is obtained. ■

Lemma 2 For any strongly-connected, legal and singularly X_r-attractable component $(\hat{X}, \hat{\alpha})$, there exists one $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, singularly X_r-attractable and control-consistent.

The proof of Lemma 2 is similar to that of Lemma 1 and hence omitted for brevity. It also requires partitioning the state space $\hat{X}$ in a manner described in the proof of Lemma 1. The only difference is the definition of X_0, which in the present case is defined as, $X_0 := \{x \in \hat{X} \mid \hat{\alpha}(x, \Sigma) \cap X_r \neq \emptyset\}$.

Theorem 1 There exists a nonblocking director for a plant G if and only if there exists a maximally permissive nonblocking supervisor for G.

Proof: Suppose a maximally permissive nonblocking supervisor exists and is computed by Algorithm 1. Then from Lemma 1, we can transform each component $(\hat{X}, \hat{\alpha}) \in \mathcal{SLIN}(X, \alpha)$ to a legal, invariant, nonblocking and control-consistent component $(LINC)$. Similarly from Lemma 2, we can transform each component $(\hat{X}, \hat{\alpha}) \in \mathcal{SLA}((X - X_k, \alpha|_{X - X_k}), X_k)$ to a legal, $LINC$-attractable and control-consistent component (LAC). Following such transformations, a graph such as in Figure 3.2(a) is converted to a graph such as in Figure 3.2(b), yielding a tree structure with leaf nodes consisting of $LINC$s (which are partitionable into $SLINC$s and $SLAC$s) and non-leaf nodes consisting of LACs (which are partitionable into $SLAC$s), thereby yielding a desired nonblocking directed plant graph.

Conversely, if a nonblocking director exists, then since a director is also a supervisor, a nonblocking supervisor exists. Finally since a nonblocking supervisor exists if and only if the maximally permissive nonblocking supervisor exists (because controllability and relative-closure are preserved under union), the assertion of the theorem follows. ■

Remark 2 The existence of a nonblocking director can be intuitively understood since a nonblocking supervisor guarantees that there exists a path to the marked states, although we do not know what the exact path is from any given state. The formal proof provided above is rather intended for the synthesis solution discussed below since the proof is given in a constructive way, which can be used to extract a director from a maximally permissive supervisor.

3.4 Synthesis of nonblocking director

We use the ideas developed in the previous section to present a set of algorithms for synthesizing a nonblocking director. The first algorithm transforms a $SLIN$ to a $LINC$; the second transforms a SLA to a LAC, and the final algorithm performs backward search over the tree of SCCs in the graph of a maximally permissive nonblocking supervised plant to find a desired director.

Algorithm 2 Given a strongly-connected, legal, invariant and nonblocking component $(\hat{X}, \hat{\alpha})$, the following steps compute a director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, invariant, nonblocking and control-consistent.

1. $\widetilde{X} = \hat{X}$; $X_0 = \emptyset$; $X_1 = \hat{X} \cap X_m$; $k = 1$;

2. For each $x \in X_k$, set the control action as

$$D(x) := \begin{cases} \{\sigma\},\ \text{where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable} \\ \qquad\qquad \text{event s.t. } \hat{\alpha}(x,\sigma) \in X_{k-1} & \text{if } \hat{\alpha}(x,\Sigma_c) \cap X_{k-1} \neq \emptyset \\ \emptyset & \text{if } \hat{\alpha}(x,\Sigma_c) = \emptyset \\ \{\sigma\},\ \text{where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable} \\ \qquad\qquad \text{event s.t. } \hat{\alpha}(x,\sigma) \in \hat{X} & \text{otherwise} \end{cases}$$

3. $\widetilde{X} = \widetilde{X} - X_k$; If $\widetilde{X} = \emptyset$, then terminate the algorithm, else continue the following steps;

4. $X_{k+1} = \{x \in \widetilde{X} \mid \hat{\alpha}(x,\Sigma) \cap X_k \neq \emptyset\}$; $k = k+1$

5. Go back to Step 2.

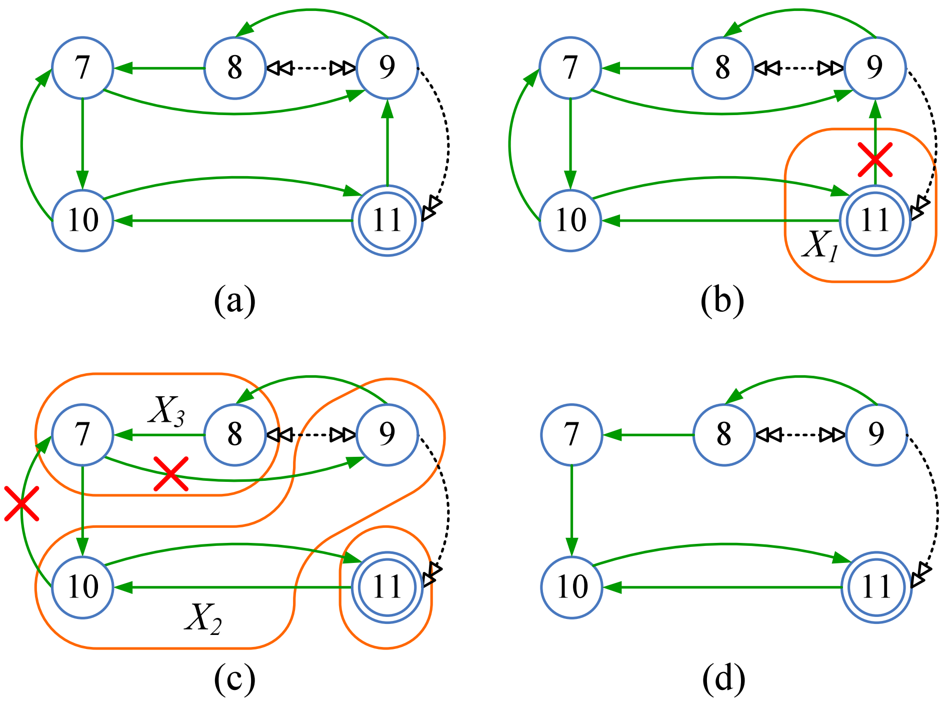


Figure 3.3 Transformation from $SLIN$ to $LINC$

We present an example to aid the understanding of Algorithm 2.

Example 2 Consider a $SLIN$ $(\hat{X}, \hat{\alpha})$ shown in Figure 3.3(a). For simplicity of discussion, we represent a component of interest by its state set. The associated state transition function can be readily identified from the figures. The following steps show how one director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is computed by Algorithm 2 such that $(\hat{X}, \hat{\alpha}^D)$ is a $LINC$. Note that a controllable event, if disabled by some underlying director, is omitted from the corresponding figures.

1. $\widetilde{X} = \hat{X} = \{7, \ldots, 11\}$; $X_0 = \emptyset$; $X_1 = \hat{X} \cap X_m = \{11\}$, as circled in Figure 3.3(b); $k = 1$;

2. We enable one controllable event for the state 11. Suppose we pick the one from the state 11 to the state 10, as shown in Figure 3.3(b).

3. $\widetilde{X} = \widetilde{X} - X_1 = \{7, \ldots, 10\} \neq \emptyset$, so we continue the following steps.

4. $X_2 = \{x \in \widetilde{X} \mid \hat{\alpha}(x, \Sigma) \cap X_1 \neq \emptyset\} = \{9, 10\}$, as circled in Figure 3.3(c); $k = 2$;

5. We return to Step 2.

2. We enable the controllable event from the state 9 to the state 8 and the one from the state 10 to the state 11.

3. $\widetilde{X} = \widetilde{X} - X_2 = \{7, 8\} \neq \emptyset$, so we continue the following steps.

4. $X_3 = \{x \in \widetilde{X} \mid \hat{\alpha}(x, \Sigma) \cap X_2 \neq \emptyset\} = \{7, 8\}$, as circled in Figure 3.3(c); $k = 3$;

5. We return to Step 2.

2. We enable the controllable event from the state 7 to the state 10 and the one from the state 8 to the state 7, as shown in Figure 3.3(c).

3. $\widetilde{X} = \widetilde{X} - X_3 = \emptyset$, so we terminate the algorithm. The resulting $(\hat{X}, \hat{\alpha}^D)$ is shown in Figure 3.3(d).

Next we prove the correctness of Algorithm 2.

Theorem 2 Consider the notation of Algorithm 2, then the algorithm terminates with one director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, invariant, nonblocking and control-consistent.

Proof: Algorithm 2 computes the same chain of disjoint state sets as presented in the proof of Lemma 1, and its correctness follows from that of Lemma 1. ■

Remark 3 Let $|\hat{X}|$ be the number of state in a $SLIN$. Due to determinism, there are at most $|\hat{X}||\Sigma|$ transitions, then it can be verified that complexity of Algorithm 2 is of order $O(|\hat{X}||\Sigma|)$.

Next we present an algorithm to transform a SLA to a LAC.

Algorithm 3 Given a reference set $X_r \in X$ and a strongly-connected, legal, singularly X_r-attractable component $(\hat{X}, \hat{\alpha})$, the following steps compute a director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, singularly X_r-attractable and control-consistent.

1. $\widetilde{X} = \hat{X}$; $k = 0$; $X_0 = X_r$;

2. $X_{k+1} = \{x \in \widetilde{X} \mid \hat{\alpha}(x, \Sigma) \cap X_k \neq \emptyset\}$; $k = k + 1$;

3. For each $x \in X_k$, set the control action as

$$D(x) := \begin{cases} \{\sigma\}, \text{ where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable} & \\ \qquad \text{event s.t. } \hat{\alpha}(x,\sigma) \in X_{k-1} & \text{if } \hat{\alpha}(x, \Sigma_c) \cap X_{k-1} \neq \emptyset \\ \emptyset & \text{if } \hat{\alpha}(x, \Sigma_c) = \emptyset \\ \{\sigma\}, \text{ where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable} & \\ \qquad \text{event s.t. } \hat{\alpha}(x,\sigma) \in \hat{X} & \text{otherwise} \end{cases}$$

4. $\widetilde{X} = \widetilde{X} - X_k$; If $\widetilde{X} = \emptyset$, then terminate the algorithm, else go back to Step 2.

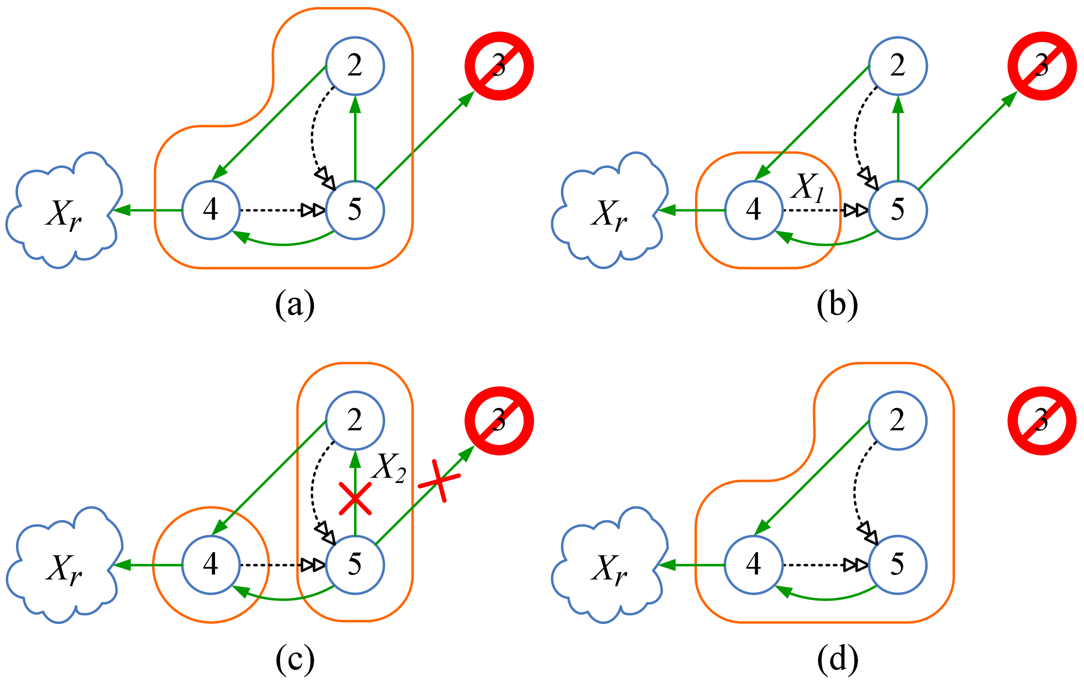


Figure 3.4 Transformation from SLA to LAC

We present an example to aid the understanding of Algorithm 3.

Example 3 Consider a SLA $(\hat{X}, \hat{\alpha})$ circled in Figure 3.4(a) and a reference state set X_r. The following steps shows how one director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is computed by Algorithm 3 such that $(\hat{X}, \hat{\alpha}^D)$ is a LAC.

1. $\widetilde{X} = \hat{X} = \{2, 4, 5\}$; $k = 0$; $X_0 = X_r$.

2. $X_{k+1} = X_1 = \{x \in \hat{X} \mid \hat{\alpha}(x, \Sigma) \cap X_r \neq \emptyset\} = \{4\}$, as circled in Figure 3.4(b); $k = 1$;

3. We enable the controllable event from the state 4 to X_r, as shown in Figure 3.4(b).

4. $\widetilde{X} = \widetilde{X} - X_1 = \{2, 5\} \neq \emptyset$, so we go back to Step 2.

2. $X_2 = \{x \in \widetilde{X} \mid \hat{\alpha}(x, \Sigma) \cap X_1 \neq \emptyset\} = \{2, 5\}$, as circled in Figure 3.4(c); $k = 2$;

3. We enable the controllable event from the state 2 to the state 4 and the one from the state 5 to the state 4, as shown in Figure 3.4(c).

4. $\widetilde{X} = \widetilde{X} - X_2 = \emptyset$, so we terminate the algorithm. The resulting $(\hat{X}, \hat{\alpha}^D)$ is circled in Figure 3.4(d).

The following theorem states the correctness of Algorithm 3, the proof of which is similar to that of Algorithm 2 and hence omitted.

Theorem 3 Consider the notation of Algorithm 3, then the algorithm terminates with one director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, singularly X_r-attractable and control-consistent.

Remark 4 Similar to the complexity of Algorithm 2, the complexity of Algorithm 3 is also of order $O(|\hat{X}||\Sigma|)$, where $|\hat{X}|$ is the number of states in a SLA.

Now we are ready to present the final algorithm that performs backward search over the tree of SCCs in the graph of a maximally permissive nonblocking supervised plant to obtain a nonblocking director.

Algorithm 4 Given a plant $G := (X, \Sigma, \alpha, x_0, X_m)$, the following steps compute a nonblocking director.

(I) **Initiation:**

1. Compute $\mathcal{SLIN}(X, \alpha)$. If $\mathcal{SLIN}(X, \alpha) = \emptyset$, then go to Step III.1; else do the following.
2. Set $k = 0$ and let $(X_k, \alpha_k) = \bigcup_{(\hat{X}, \hat{\alpha}) \in \mathcal{SLIN}(X,\alpha)} (\hat{X}, \hat{\alpha}^D)$, where $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is a director computed by Algorithm 2 for each $(\hat{X}, \hat{\alpha}) \in \mathcal{SLIN}(X, \alpha)$.

(II) **Iteration:**

1. If $x_0 \in X_k$, then go to Step III.2; else do the following.

2. Let $X' = X - X_k$; $\alpha' = \alpha|_{X'}$. Compute $\mathcal{SLA}((X', \alpha'), X_k)$. If $\mathcal{SLA}((X', \alpha'), X_k) = \emptyset$, then go to Step III.1; else do the following.
3. $(\widetilde{X}, \widetilde{\alpha}) = \bigcup_{(\hat{X}, \hat{\alpha}) \in \mathcal{SLA}((X', \alpha'), X_k)} (\hat{X}, \hat{\alpha}^D)$, where $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is a director computed by Algorithm 3 for each $(\hat{X}, \hat{\alpha}) \in \mathcal{SLA}((X', \alpha'), X_k)$;
4. $(X_{k+1}, \alpha_{k+1}) = (X_k, \alpha_k) \cup (\widetilde{X}, \widetilde{\alpha})$;
5. Go back to Step II.1 with $k = k + 1$.

(III) **Termination:**

1. Stop since no nonblocking director exists.
2. Stop since a nonblocking director is found.

We present an example to aid the understanding of Algorithm 4.

Example 4 Consider a plant $G = (X, \Sigma, \alpha, x_0, X_m)$ shown in Figure 3.5(a). The following steps show how a nonblocking director is computed by Algorithm 4. Note that this plant includes the components used in Example 2 and 3 (so some constructions performed previously are reused to facilitate the computation). Note that the edges with dashed line and double *solid* arrows represent the transitions on (uncontrollable) sensor output while those with dashed line and double *hollow* arrows represent the transitions on (uncontrollable) disturbance input. From the example, the transition from the state 6 to the state 5 is on (uncontrollable) sensor output.

Initiation:

1. According to the algorithm that computes $\mathcal{SLIN}(x, \alpha)$ (see Algorithm 10 in the Appendix), the state 3 and 12 are excluded from computation because they are illegal states.

   Then the state 13 becomes variant and therefore is excluded, too, as shown in Figure 3.5(b). Note that the state 6 is not variant because $6 \notin X_e$.

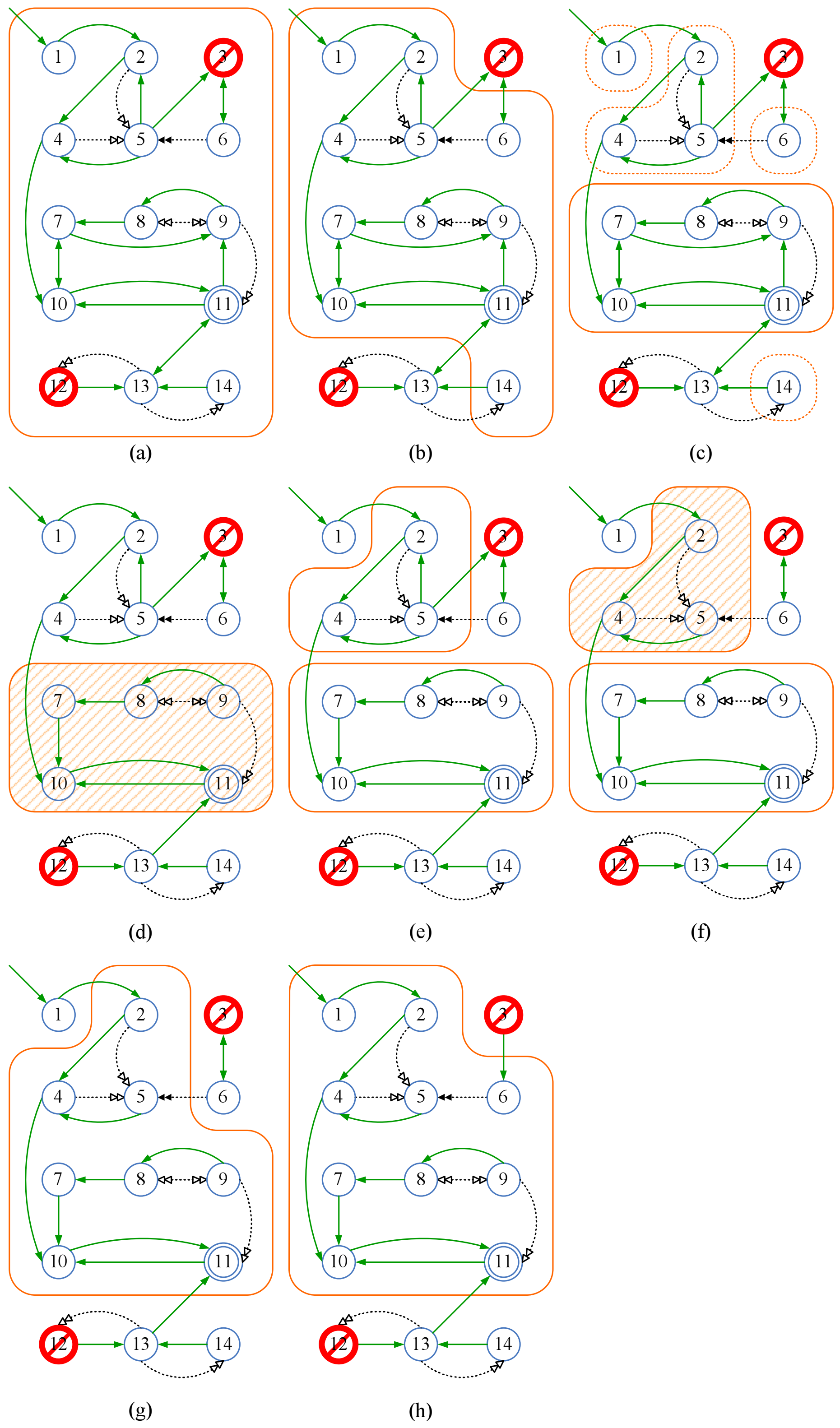


Figure 3.5 Computing a nonblocking director

Next the maximal SCCs of the remaining plant are identified and only $\{7, \ldots, 11\}$ is invariant and nonblocking, as shown in Figure 3.5(c). So $\mathcal{SLIN}(x, \alpha) = \{\{7, \ldots, 11\}\}$.

The result in Example 2 is reused to obtain a $LINC$, $\{7, \ldots, 11\}$, which is circled and shaded in Figure 3.5(d).

2. $(X_0, \alpha_0) = \{7, \ldots, 11\}$.

Iteration 1: $k = 0$.

1. Since $x_0 = 1 \notin X_0$, we continue the following steps.
2. $X' = \{1, \ldots, 6, 12, 13, 14\}$. By the algorithm that computes $\mathcal{SLA}$ (see Algorithm 11 in the Appendix), we get $\mathcal{SLA}((X', \alpha'), X_0) = \{\{2, 4, 5\}\}$, as circled in Figure 3.5(e). The result in the Example 3 is reused to obtain a LAC, $\{2, 4, 5\}$, circled and shaded in Figure 3.5(f).
3. It follows that $(\widetilde{X}, \widetilde{\alpha}) = \{2, 4, 5\}$; and
4. $X_1 = X_0 \cup \{2, 4, 5\} = \{2, 4, 5, 7, \ldots, 11\}$, as circled in Figure 3.5(g).

Iteration 2: $k = 1$.

1. Since $x_0 = 1 \notin X_1$, we continue the following steps.
2. $X' = \{1, 3, 6, 12, 13, 14\}$. By the algorithm that computes $\mathcal{SLA}$ and Algorithm 3, we get two LACs, $\{1\}$ and $\{6\}$.
3. It follows that $(\widetilde{X}, \widetilde{\alpha}) = \{1, 6\}$; and
4. $X_2 = X_1 \cup \{1, 6\} = \{1, 2, 4, \ldots, 11\}$, as circled in Figure 3.5(h).

Termination: Because $x_0 = 1 \in X_3$, the algorithm stops with a nonblocking director having been found.

The following theorem proves the correctness of Algorithm 4.

Theorem 4 Consider the notation of Algorithm 4, then Algorithm 4 terminates with $x_0 \in X_k$ if and only if $G|_{(X_k,\alpha_k)}$ is a plant controlled by a nonblocking director.

Proof: It is easy to see Algorithm 4 implements the computation discussed in the proof of Theorem 1 and hence is correct. ■

Remark 5 From [17], we know the overall complexity of computing a maximally permissive nonblocking supervisor is quadratic, namely, of order $O(|\hat{X}|^2|\Sigma|^2)$. In addition, we also show in the Remark 3 and 4 that the complexity of additional steps to transform $SLIN/SLA$ to $LINC/LAC$ is linear. Thus the complexity of the above set of algorithms that computes a nonblocking director is also of order $O(|\hat{X}|^2|\Sigma|^2)$.

Remark 6 The nonblocking director we synthesize is deterministic in the sense that it computes for each state a fixed control action to perform. It is also possible to consider a director that uses randomization for selecting control actions. Such a director may perform different control selections on different visits of the same state. By the rules of probability, the system would eventually reach a marked state, if a nonblocking director exists. Since we don't define any performance index for logic correctness, this randomization approach also works. However, such an approach makes the system evolution path unpredictable and aimless, which is often undesired. Instead, we take the "deterministic" approach and the notion of attraction, which brings the system one step closer to the desired final states with each execution of the selected control action.

Remark 7 Depending on the application, the optimality of the director synthesized can also be of interest, though not always necessary. For example, in real-time systems where tasks are scheduled to meet deadlines, a popular scheduling scheme is the static priority based scheduling, where the priority of each task is fixed relative to other tasks.

By no means this is optimal in terms of processor utilization, but nonetheless widely practiced. Such a scheme is guaranteed to be logically correct, i.e., meet deadlines, if and only if a deadline meeting scheme exists.

3.5 Application example

We provide an application example to demonstrate our result. The application is of controlling train traffic over a set of track sections.

As shown in Figure 3.6(a), six sections of tracks labeled from 1 to 6 are separated by some traffic lights and a switch. The traffic lights can be used to stop the traffic in the directions as indicated by the arrows above the lights, but have no effect for the traffic in the opposite directions. A switch connects the section 2, 3 and 4. A train coming from the section 2 can go to the section 3 or 4 while a train coming from the section 3 or the section 4 can go to the section 2 via the switch.

Suppose initially two trains T_1 and T_2 are in the section 1 and 2, respectively. We are required to synthesize a nonblocking director to control the traffic lights and the switch to ensure T_1 and T_2 eventually swap their positions without running into each other.

We model the movement of T_1 and T_2 by two automata G_1 and G_2, respectively, shown in the Figure 3.6(b). The combined movement of two trains is given by $G_1||G_2$. The corresponding graph is denoted as (X, α), shown in Figure 3.6(c), where the scenarios that two trains are in the same section are represented by illegal states.

Applying Algorithm 4, we first compute $\mathcal{SLIN}(X, \alpha)$, which yields only one $SLIN$ in this example. Since this $SLIN$ includes the initial state x_0, it turns out to be the maximally permissive nonblocking supervisor for the plant (X, α), as shown in Figure 3.6(d).

Next Algorithm 2 is applied to transform this $SLIN$ into a $LINC$. The corresponding chain of disjoint state sets have been identified and the control actions for each state

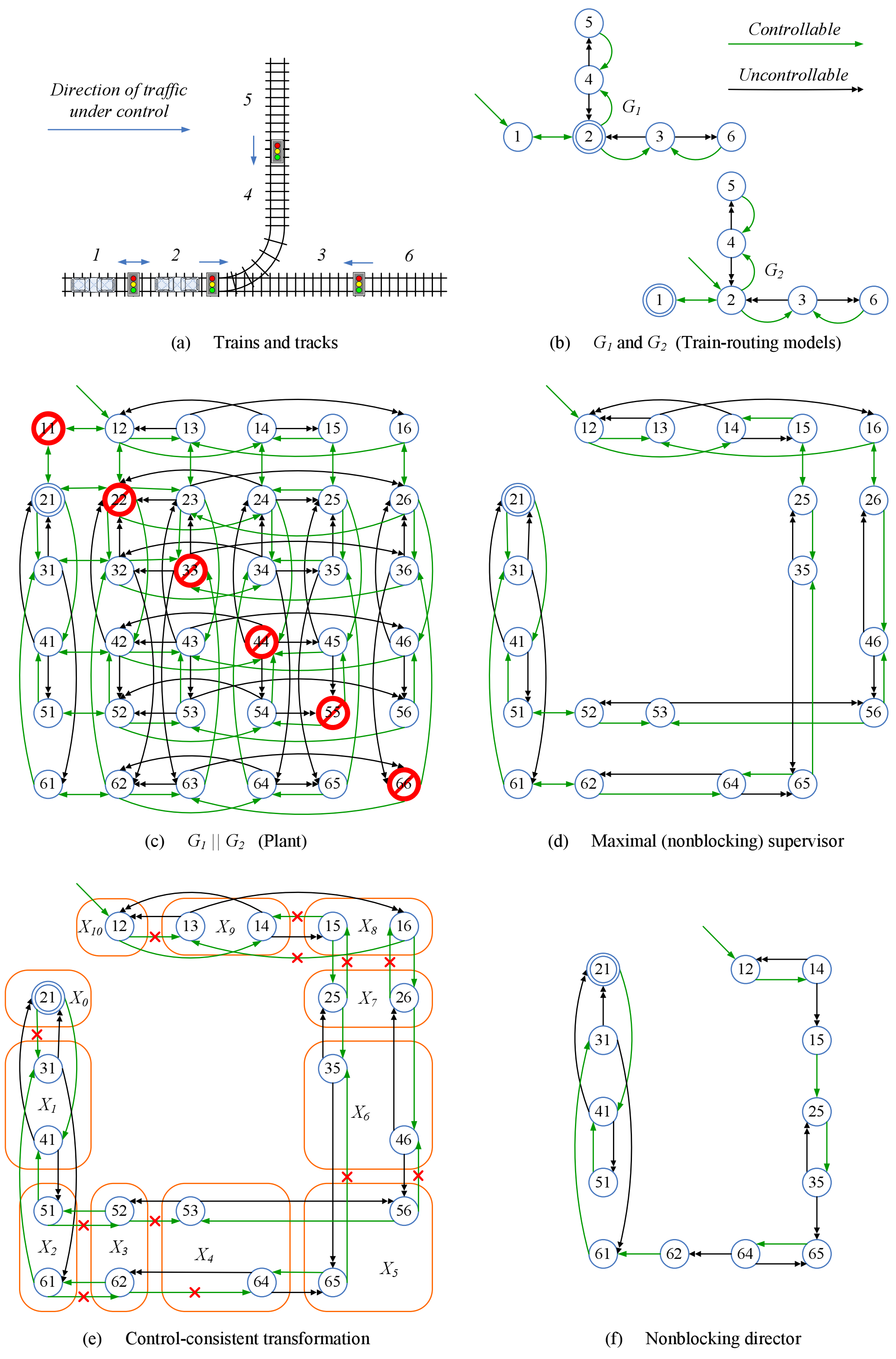

Direction of traffic under control
1
2
3
4
5
6
Controllable
Uncontrollable
G_1
G_2
(a) Trains and tracks
(b) G_1 and G_2 (Train-routing models)
(c) $G_1 \,||\, G_2$ (Plant)
(d) Maximal (nonblocking) supervisor
(e) Control-consistent transformation
(f) Nonblocking director
X_0
X_1
X_2
X_3
X_4
X_5
X_6
X_7
X_8
X_9
X_{10}

Figure 3.6 Synthesis of a nonblocking director for the application example

are obtained. In our example, the chain contains 11 disjoint state sets, as shown in Figure 3.6(e). Note that the disabled events are removed from the figure. After trimming, the resulting nonblocking director is shown in Figure 3.6(f).

The control strategy with this nonblocking director is summarized as follows. We first guide T_2 from the section 2 to the section 5 via the section 4. Then we guide T_1 from the section 1 to the section 6 via the section 2 and 3. Next we release T_2 from the section 5 and move it to its destination, the section 1, via the section 4 and 2. In the end we move T_1 back from the section 6 to its destination, the section 2, via the section 3.

CHAPTER 4. OPTIMIZATION BASED FRAMEWORK

4.1 Introduction

In the previous chapter, we have addressed the control existence and synthesis problems for the logically correct behavior of the systems under directed control. Starting from this chapter, we will address the best performance, or optimality issue of directed control. We begin with an optimization based framework.

This framework differs from the prior work on optimal supervisory control, where optimization is with respect to the set of supervisors in contrast to the set of directors. Also, the optimization criterion we consider is different from those considered in prior works. In [20], a cost function is defined on the set of transitions, and the control objective is to restrict the plant behavior in such a way that after starting from *a given* initial state, the plant reaches one of the marked states along a trajectory of optimal cost. In [7], a cost function is defined on the set of transitions and the control objective is to restrict the plant behavior so that after starting from *any* state the plant reaches one of the marked states along a trajectory of optimal cost. In contrast to the distance function of [7] which computes the worst-case cost to a *fixed set of states*, we consider distance function to be the worst-case cost to the *set of nearest reachable marked states.*

In [23], two types of cost are defined: a path cost and a control cost on the graph representing a plant. The control objective is to determine a subgraph of the graph in which the worst case path (in terms of cost) among all possible paths in that subgraph, is still the best compared to worst case paths of other subgraphs. While such a problem admits many solutions, the authors study maximal DP-optimal (dynamic-programming optimal) solutions. A DP-optimal solution has the property that its *admissible* subgraphs are also optimal in their corresponding subproblems. This formulation was later generalized to the setting of languages (rather than graphs) in [24] and to the case of partial observation in [19]. In [16], control costs are associated with transitions (these are the costs incurred in disabling the transitions). Moreover, the states are classified into desirable and undesirable states, and a certain cost is incurred if an undesirable state is reached or if a desirable state is not reached in the supervised plant. The objective in that work is to find a supervisor for which the net cost incurred in disabling events, reaching undesirable states, and missing desirable states, is minimized. The authors use network flow methods, namely the max-flow min-cut theorem, to synthesize such a supervisor. Further by selecting appropriate cost functions, the authors are able to compute a variety of languages that are of interest in the control of discrete event systems by reducing the corresponding supervisory control problems to an instance of an optimal control problem of their setting.

In the subsequent sections, we will formulate our optimization based framework, introduce the notion of dynamic-programming optimality, and address the existence problem of optimal director. Some examples will be provided to aid the understanding of the notions and an algorithm to compute director cost.

4.2 Optimal directed control problem

Let $c : \Sigma^* \times \Sigma \rightarrow \mathcal{R}^+$ denote a cost function of control, where $\mathcal{R}^+$ denotes the set of positive reals, *including* infinity. For each $s \in \Sigma^*$ and $\sigma \in \Sigma$, the cost $c(s, \sigma)$ represents the cost of "selecting" the event $\sigma \in \Sigma$ following the execution of trace s.

Remark 8 For a trace $s \in \Sigma^*$ and an uncontrollable event $\sigma \in \Sigma_u$, $c(s, \sigma)$ represents the execution cost of σ following the execution of s and the payoff of reaching the resulting state $\alpha(x_0, s\sigma)$, whereas for a controllable event $\sigma \in \Sigma_c$, $c(s,\sigma)$ represents the execution cost of σ following the execution of s and the payoff of reaching the resulting state $\alpha(x_0, s\sigma)$, *together with* the disablement cost of all other controllable events feasible at trace s. Thus a single cost function suffices to capture both the "control cost" and "path cost" introduced in [24]. Also, the cost $c(s,\sigma)$ can include the cost of reaching the plant state $\alpha(x_0, s\sigma)$. For example, if the state $\alpha(x_0, s\sigma)$ is an illegal state (or equivalently, the trace $s\sigma$ does not belong to the specification language), then we can set $c(s,\sigma) = \infty$.

By induction, the cost function can be extended to a mapping $c : \Sigma^* \times \Sigma^* \to \mathcal{R}^+$ as follows:

$$\forall s, t \in \Sigma^*, \sigma \in \Sigma : \begin{cases} c(s,\epsilon) := 0 \text{ and} \\ c(s, t\sigma) := c(s,t) + c(st,\sigma) \end{cases}$$

We next introduce the notion of frontier traces needed for defining the cost of a director.

Definition 6 Given a trace $s \in \Sigma^*$, its set of *frontier* traces in a language K, denoted as $(K\backslash s)_f$, is given by:

$$\forall s \in \Sigma^*\text{: } (K\backslash s)_f := \{t \neq \epsilon \mid st \in K \text{ and } \forall u < t : su \notin K\}$$

The frontier traces of s are the minimal extensions of s with non-zero length such that the extensions belong to K. If K is the set of marked traces, then execution of any such extension implies accomplishing a task that is pending to be completed.

Definition 7 Given a plant G, a director $D : L(G) \to 2^{\Sigma_c}$, a trace $s \in L(G^D)$ and a marked frontier (trace) extension $t \in (L_m(G^D)\backslash s)_f$, $c(s,t)$ denotes the cost of "completing the task" along the trace t following the trace s. We consider the costs of all such marked frontier extensions t of s in $L(G^D)$, and determine the worst possible cost $\max_{t\in(L_m(G^D)\backslash s)_f} c(s,t)$. We do this for all traces s of $L(G^D)$, and use the largest worst case cost among all traces s as the cost $P(D)$ of the director D, i.e.,

$$P(D) := \begin{cases} \max_{s \in L(G^D)}[\max_{t \in (L_m(G^D) \backslash s)_f} c(s,t)] & \text{if } \forall s \in L(G^D) : |(L_m(G^D) \backslash s)_f| < \infty \\ \infty & \text{otherwise} \end{cases}$$

where for any $s \in L(G^D)$ such that $(L_m(G^D) \backslash s)_f = \emptyset$, we define

$$\max_{t \in (L_m(G^D) \backslash s)_f} c(s,t) := \begin{cases} 0 & \text{if } s \in L_t(G) \cap L_m(G^D) \\ \infty & \text{otherwise} \end{cases}$$

Since executing a marked trace amounts to finishing a pending task, $P(D)$ thus represents the worst case cost of finishing a pending task under the control of D. It follows from the definition that if D is blocking, i.e., if there exists $s \in L(G^D) - L_m(G^D)$ such that $(L_m(G^D) \backslash s)_f = \emptyset$, then $P(D) = \infty$. Thus any director with finite cost must be nonblocking.

The objective of the optimal control is to find a director with the least cost.

Optimal directed control problem: Determine whether there exists

$$D^* \in \arg\left\{\min_D P(D)\right\} \text{ with } P(D^*) < \infty$$

and if yes, then find one such D^*.

The motivation behind the above optimal control problem is that under the control of an optimal director, the system should accomplish a pending task (by executing a marked frontier extension) in a minimal possible cost for all possible histories of its evolution. Since the cost of a blocking director is infinity by definition, an optimal director, when it exists, is nonblocking. It may be noted that the optimal solution need not be unique.

In our framework, once a non-terminating marked state is reached, the system gets "re-initialized" with the marked state just reached acting as a new initial state and the existing marked states continuing to act as the final states.

This enriched view of non-terminating marked states is formulated in the following definition:

Definition 8 Given a plant $G := (X, \Sigma, \alpha, x_0, X_m)$, I(G) denotes the set of all possible states from which the plant can "initialize" or "re-initialize", i.e.,

$$I(G) := \{x_0\} \cup (X_m - X_t)$$

It is possible that a re-initialization results in a completely new set of marked states. Our framework can easily accommodate this case by defining an extended model as the concatenation of a suitable number of appropriately initialized plant models.

Remark 9 In general, we treat all marked states "equally", i.e., we don't favor some marked states over the others in terms of completing a task. But sometimes differentiating marked states is useful and desired, e.g., some marked states may be more important or preferred. We can incorporate this preference in our framework by the following additional modeling steps:

1. Create a dummy state for each "less favorable" marked state;

2. Redirect all the incoming transitions to a less favorable marked state to its corresponding dummy state;

3. Move all the outgoing transitions from a less favorable marked state to start from its corresponding dummy state instead;

4. Create a new transition from each dummy state to its corresponding marked state, and assign an appropriate "penalty" cost for the transition.

With the above steps, the cost to reach a less favorable marked state has been increased, thus the solution to the optimal control problem would "favor" those preferred marked states.

In the above optimal directed control problem, the cost function is "trace-based" for the sake of generality. In a practical setting one would typically start with a "state-based" representation of the cost function. Also, it is possible to suitably refine the plant model so that the cost function can be viewed as "state-based" (with respect to the states of the refined plant) by defining a certain equivalence relation over the set of traces. While in general such a refinement may not preserve the finiteness of the state space, it will do so in any practical setting where typically the given trace-based cost function will depend only on a bounded history of the system evolution.

With this understanding, we formulate and study the optimal director problem in a state-based setting. A cost function is state-based if for all $s, t \in \Sigma^*$ such that $\alpha(x_0, s) = \alpha(x_0, t) := x$, it holds that for all $u \in \Sigma^*$ such that $\alpha(x, u)$ is defined, $c(s, u) = c(t, u)$. In such a case, the cost function can be specified as a map, $c : X \times \Sigma \rightarrow \mathcal{R}^+$ so that for $x \in X$ and $\sigma \in \Sigma$, $c(x, \sigma)$ is the cost of executing σ at state x.

A plant $G := (X, \Sigma, \alpha, x_0, X_m)$ can also be viewed as a weighted directed-graph $G := (X, E, X_m)$, where $E \subseteq X \times \Sigma \times X$ is the set of transitions, called the set of edges. An edge of G is an ordered triple $e = (x_1, \sigma, x_2) \in E$ with $\alpha(x_1, \sigma) = x_2$, and is said to be labeled by event σ, and directed from state x_1 to state x_2. The edge e is assigned a control cost, denoted $c(e)$, with value $c(e) = c(x_1, \sigma)$.

We use $E(x)$ to denote the set of edges defined at x, and $E_c(x)$ (resp. $E_u(x)$) to denote the set of edges labeled with controllable (resp. uncontrollable) events at x. The *trim* directed plant under a state-based director D is given by $G^D = (X^D, E^D, X_m^D)$, where $E^D \subseteq E$ represents the set of transitions enabled by the director D, and X^D (resp. X_m^D) represents the set of states (resp. the set of marked states) in the *trim* directed plant. Thus the set of all possible initialization states of G^D is given by $I(G^D) := \{x_0\} \cup (X_m^D - X_t)$. For $x \in X$, $D \backslash x$ is used to denote the director D with x treated as the initial state, and we call it a *director rooted at* x. Note that $D \backslash x_0 = D$.

A *path* p is a *finite* sequence of edges $p = (x_1, \sigma_1, x_2)(x_2, \sigma_2, x_3) \cdots (x_{n-1}, \sigma_{n-1}, x_n)$, in which p is said to start from x_1 and end at x_n. The cost of path p is the sum of the costs of its edges, and is denoted by $c(p)$. For each $x \in X$, let $\Pi^f_G(x)$, called the *set of marked frontier paths* from x in G, denote the set of all paths of G of the form $p = (x_1, \sigma_1, x_2)(x_2, \sigma_2, x_3) \cdots (x_{n-1}, \sigma_{n-1}, x_n)$ such that $x_1 = x, x_n \in X_m$, and for all $i = 2, \cdots, n-1$, $x_i \notin X_m$. The states reached from x via the marked frontier paths of $\Pi^f_G(x)$ are called x's marked frontier states.

Definition 9 *The distance of x (to its marked frontier states in G)*, denoted $d_G(x)$, is defined as the worst cost of all paths in $\Pi^f_G(x)$, i.e.,

$$d_G(x) := \begin{cases} \max\{c(p) \mid p \in \Pi^f_G(x)\} & \text{if } 0 < |\Pi^f_G(x)| < \infty \\ 0 & \text{if } \Pi^f_G(x) = \emptyset \text{ and } x \in X_{tm} \\ \infty & \text{otherwise} \end{cases}$$

It is clear that only terminating marked states have the distance of 0. Also note that when $\Pi^f_G(x) \neq \emptyset$,

$$\begin{aligned} d_G(x) &= \max\{d_G(x') + c(e) \mid e = (x, \sigma, x') \in E\} \\ &= \max\{d_1, d_2\} \end{aligned} \tag{4.1}$$

where

$$d_1 := \max\{d_G(x') + c(e) \mid e = (x, \sigma, x') \in E, x' \notin X_m\}$$

$$d_2 := \max\{0 + c(e) \mid e = (x, \sigma, x') \in E, x' \in X_m\}$$

From Definition 7 and 9, it follows that the cost of a director D is given by,

$$P(D) = \max_{x \in X^D} d_{G^D}(x) \tag{4.2}$$

Note that if two states x_1 and x_2 are such that every path in $\Pi^f_G(x_1)$ is a suffix of some path in $\Pi^f_G(x_2)$, then $d_G(x_1) \leq d_G(x_2)$. So Equation 4.2 is equivalent to the following:

$$P(D) = \max_{x \in I(G^D)} d_{G^D}(x) \tag{4.3}$$

It follows that the optimal control problem is to find

$$D^* \in \arg\left\{\min_D P(D)\right\} = \arg\left\{\min_D \max_{x \in I(G^D)} d_{G^D}(x)\right\}$$

Remark 10 Note that the definition of distance function $d_G(\cdot)$ given above computes the worst case distance from a state to *it's set of marked frontier states.* This is different from the distance function consider in [7], which computes the worst case distance of a state to *a fixed set of marked states.* Another difference between our work and the work in [7] is that we optimize over the set of directors, whereas the optimization in [7] is performed over the set of all "stabilizing" supervisors.

Remark 11 In our setting, we do not explicitly define "illegal" states, but they can be identified by suitably defining the cost function. Any state that is deemed illegal should have an infinite cost for all incoming edges to the state. As a result, none of such edges should be present in a trim optimally directed plant, ensuring the unreachability of that state.

If a plant G is not trim or contains edges with infinite cost, then such a plant can be reduced to a trim one having no edges with infinite cost while preserving the solution to the optimal director problem. We provide an algorithm below for such a reduction.

Algorithm 5 Input a plant G, the following steps generate a reduced trim plant G' such that an optimal director for G' exists if and only if an optimal director for G exists.

1. For each $x \in X$, remove the transitions on $\sigma \in \Sigma_c(x)$ if $c(x, \sigma) = \infty$ and $x \in X_e \Rightarrow \exists \sigma' \in \Sigma_c(x) : c(x, \sigma') \neq \infty$. (Note that this step may result in new terminating states.)

2. $k = 0$; $X_k := \{x \in X \mid \exists \sigma \in \Sigma_u(x) \text{ s.t. } c(x,\sigma) = \infty\} \cup \{x \in X_e \mid \forall \sigma \in \Sigma_c(x) : c(x,\sigma) = \infty\} \cup \{x \in X \mid \forall s \in \Sigma^* : \alpha(x,s) \notin X_m\}$.

3. $X_{k+1} := X_k \cup \{x \in X - X_k \mid \exists \sigma \in \Sigma_u(x) : \alpha(x,\sigma) \in X_k\} \cup \{x \in (X - X_k) \cap X_e \mid \forall \sigma \in \Sigma_c(x) : \alpha(x,\sigma) \in X_k\} \cup \{x \in X - X_k \mid \forall \sigma \in \Sigma(x) : \alpha(x,\sigma) \in X_k\}$.

4. Repeat Step 3 with $k = k+1$ until $X_{k+1} = X_k$.

5. Remove all the incoming transitions to the states of X_k.

6. Trim the plant and the resulting plant is G'.

The following theorem establishes the correctness of Algorithm 5.

Theorem 5 Given a plant G, consider the reduced plant G' computed by Algorithm 5. An optimal director for G' exists if and only if an optimal director for G exists.

Proof: First note that the set of directors of G can be divided into two classes, one that restricts G to a subgraph of G' and another that doesn't. It suffices to show that the second class of directors cannot be optimal. Next note that state set of G' is either a strict subset of that of G, or it equals the state set of G. First consider the latter, i.e., $X_0 = \emptyset$. Then there are two cases: either edge set of G is a strict subset of that of G', or the two edge sets are the same. If it is the case of the latter, then there is nothing to prove since $G = G'$. On the other hand, if it is the case of the former, then some edges are removed in Step 1 of the algorithm. Since these edges are of infinite cost, they must not belong to any optimally directed G. This proves that in case $X_0 = \emptyset$, the class of directors that do not restrict G to a subgraph of G' are not optimal. Next consider the case when $X_0 \neq \emptyset$. Then any state in X_0 must not be reached in any optimally directed G, since otherwise the cost of the corresponding director will be infinity. (This is because every state in X_0 has an uncontrollable transition of infinite cost, or is an element of X_e where all controllable transitions are of infinite cost, or is not coaccessible.

Recall that (i) uncontrollable transitions cannot be disabled; (ii) at a state of X_e, at least one controllable transition must be enabled; (iii) all optimal directors are nonblocking.) Finally we claim that when $X_0 \neq \emptyset$, any state in $X_{k(\geq 1)}$ must not be reached in an optimally directed G. This is because if a state is reached in $X_{k(\geq 1)}$, then there exists no director that can prevent G from reaching a state in X_0. (This is because each state in X_k has an uncontrollable transition to X_{k-1}, or is an element of X_e and all its controllable transitions lead to X_{k-1}, or simply has all its transitions leading to X_{k-1}.) It should also be clear that the states being trimmed out are inaccessible under the control of any optimal director of G. This completes the proof. ■

Remark 12 With the above reduction result of Algorithm 5 in place, it follows that there is no loss in generality to consider only the plants that are trim and all edges are of finite cost.

Central to our director existence and synthesis is the notion of attraction introduced in [6], which is presented below.

Definition 10 Given a plant G, consider a state set $\hat{X} \subseteq X$. A state $x \in X$ is said to be *1-step $\hat{X}$-attractable* in G if $\alpha(x, \Sigma) \cap \hat{X} \neq \emptyset$ and $\alpha(x, \Sigma) \subseteq \hat{X}$. $x \in X$ is said to be *1-step directively $\hat{X}$-attractable* in G if there exists a director D such that x is *1-step $\hat{X}$-attractable* in G^D. In other words, $x \in X$ is *1-step directively $\hat{X}$-attractable* in G if $\alpha(x, \Sigma) \cap \hat{X} \neq \emptyset$, $\alpha(x, \Sigma_u) \subseteq \hat{X}$ and $x \in X_e \Rightarrow \alpha(x, \Sigma_c) \cap \hat{X} \neq \emptyset$.

Note that if $x \in X$ is 1-step attractable to $\hat{X} \subseteq X$, then x is also 1-step directively attractable to $\hat{X}$, i.e., the notion of 1-step attractability is stronger than the notion of 1-step directive attractability. In the following definition, the notion of single-step attractability is generalized to multi-step attractability.

Definition 11 Given a reference state set $\hat{X} \subseteq X$, $x \in X$ is *$\hat{X}$-attractable* in G if there exists a non-negative integer N such that for all paths p from x of length greater than or equal to N, p visits $\hat{X}$. $x \in X$ is *directively* $\hat{X}$*-attractable* in G if there exists a director D such that x is $\hat{X}$*-attractable* in G^D. We use $\Omega(\hat{X})$, called the *region of attraction of* $\hat{X}$, to denote the set of all $\hat{X}$-attractable states, and $\Omega_d(\hat{X})$, called the *region of directive attraction* of $\hat{X}$, to denote the set of all directively $\hat{X}$*-attractable* states. A state set $\widetilde{X} \subseteq X$ is said to be *attractable* to $\hat{X}$ if $\widetilde{X} \subseteq \Omega(\hat{X})$, and *directively attractable* to $\hat{X}$ if $\widetilde{X} \subseteq \Omega_d(\hat{X})$.

Note that from the definition of multi-step attractability and the fact that the application of control decreases the number of enabled outgoing transitions for states, it follows that $\hat{X} \subseteq \Omega(\hat{X}) \subseteq \Omega_d(\hat{X})$ for any $\hat{X} \subseteq X$.

Remark 13 The notion of attractability given in Definition 11 is identical to that of strong attractability first introduced in [6]. On the other hand, the notion of directive attractability given in Definition 11 is stronger than the notion of weak attractability introduced in [6]. Directive attractability is relevant in our context (and not weak attractability) since we are considering directors and not supervisors.

4.3 Dynamic-programming optimality

Among the optimal directors, of special interest are those that are dynamic-programming optimal, which we define next:

Definition 12 Given a plant G, and cost of the control function $c : \Sigma^* \times \Sigma \to \mathcal{R}^+$, an optimal director D is said to be *dynamic-programming optimal* (DP-optimal) if for all $s \in L(G^D)$, $D\backslash s$ is also optimal for $G\backslash s$ with respect to the cost function $c\backslash s$, where for $t \in \Sigma^*, \sigma \in \Sigma$,

$$D\backslash s(t) := D(st); \quad c\backslash s(t, \sigma) := c(st, \sigma)$$

and $G\backslash s$ denotes the plant G with a new initial state given by $\alpha(x_0, s)$.

The DP-optimality property is not possessed by all optimal directors, i.e., there exist optimal directors but their "sub-directors" are not optimal for the "sub-problems". However, by replacing such sub-directors with optimal ones, one can always obtain a DP-optimal director.

The following example compares the notions of optimal and DP-optimal directors.

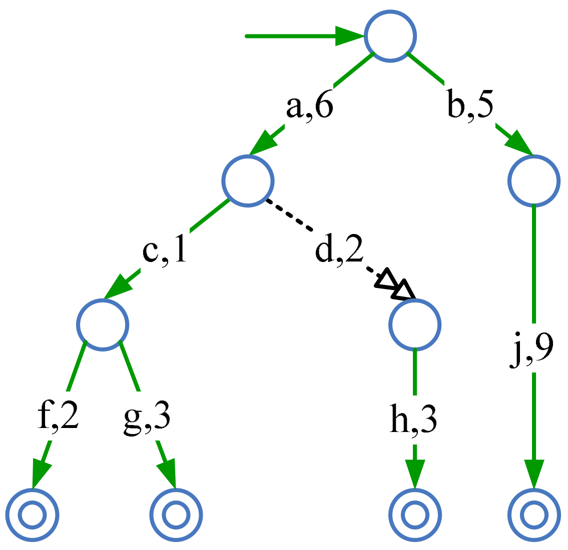


Figure 4.1 The plant of Example 5

Example 5 Consider a plant G with languages $L_m(G) = \{acf + acg + adh + bj\}$ and $L(G) = pr(L_m(G))$. The plant is shown in Figure 4.1, where an edge with dashed line and double arrows represents a transition on an uncontrollable event while an edge with solid line and single arrow represents a transitions on a controllable event. The edges are labeled with events as well as costs.

For illustration, we consider three possible directors D_1, D_2 and D_3: D_1 and D_2 select "a" in the initial state, whereas D_3 selects "b" in the initial state. In the state reached following the execution "ac", D_1 selects "f", whereas D_2 selects "g". Note that the uncontrollable event "d" remains enabled following "a" under both D_1 and D_2.

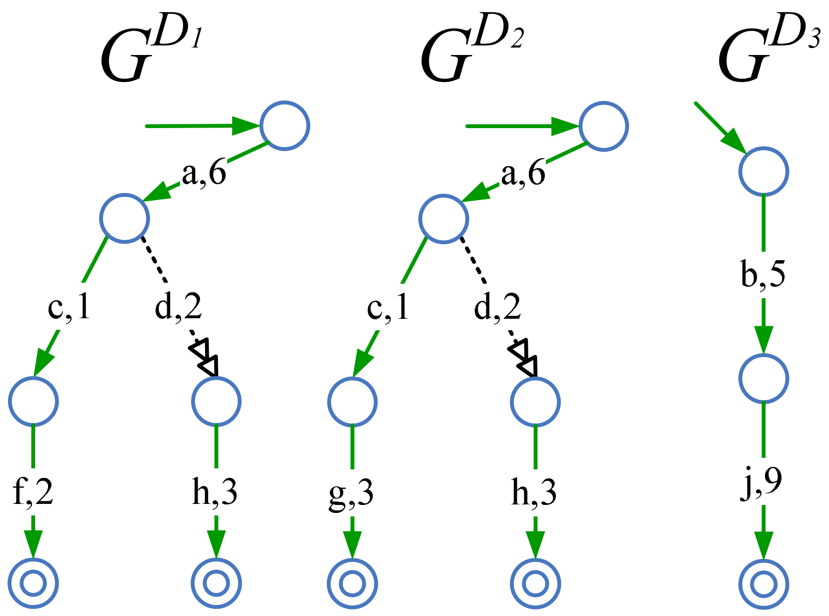


Figure 4.2 Directed plants G^{D_1}, G^{D_2} and G^{D_3} of Example 5

Then $L_m(G^{D_1}) = \{acf + adh\}$ and $L(G^{D_1}) = pr(L_m(G/D_1))$. Similarly, $L_m(G^{D_2}) = \{acg + adh\}$ and $L(G^{D_2}) = pr(L_m(G^{D_2}))$. Finally, $L_m(G^{D_3}) = \{bj\}$ and $L(G^{D_3}) = pr(L_m(G^{D_3}))$. All directed plants are shown in Figure 4.2. Note that none of the three directed behaviors is the same to each other.

As we can see, the cost associated with D_1 and D_2 is dominated by the cost incurred following the trace "*adh*", which is 11. The cost of D_3 can be seen to be $5 + 9 = 14$. Thus, out of the three possible directors D_1, D_2 and D_3, two are optimal, namely, D_1 and D_2. Clearly both D_1 and D_2 are solutions to the optimal control problem, but D_2 is not DP-optimal. After the system has executed the trace "*ac*", the worst case cost beyond "*ac*" is 3 under D_2, whereas it is 2 under D_1. So although both D_1 and D_2 have same overall cost of 11, the "sub-director" $D_1\backslash ac$ is optimal for $G\backslash ac$, whereas $D_2\backslash ac$ does not have this property. In other words, D_1 is DP-optimal, whereas D_2 is not DP-optimal.

4.4 Existence of optimal director

The number of directors is an exponential function of the number of plant states. This is because at each state there are at most $|\Sigma_c|+1$ choices for a director action, and so there are at most $(|\Sigma_c|+1)^{|X|}$ directors. When $|X|$ is finite, it is possible to search exhaustively over the set of finitely many directors for an optimal director, provided we know how to compute the cost of a state-based director. Thus, having an algorithm to compute the cost of a given state-based director will establish the solvability of the optimal control problem. In this section we give an algorithm of polynomial complexity to compute the cost of a given state-based director.

Given a state-based director D (or equivalently, the directed plant G^D), the following algorithm computes its cost, $P(D)$.

Algorithm 6 Input G^D

Initiation:

Set $k=0$, $\Omega_k = X^D_{tm}$, $\forall v \in \Omega_k : \rho(v)=0$, and $\forall v \in X^D_m : \lambda(v)=0$.

Iteration:

1. Let V be the set of all *1-step* $(\Omega_k \cup X^D_m)$*-attractable* states, and for all $v \in V$ compute, $\rho(v) = \max\{\lambda(v') + c(e) \mid e=(v,\sigma,v') \in E^D\}$

2. Let $\Omega_{k+1} = \Omega_k \cup V$, and for each $v \in V - X^D_m$, let $\lambda(v)=\rho(v)$.

Termination:

- If $\Omega_{k+1} = X^D$, then set $\Pi(D) = \max_{v\in X^D}\rho(v)$, and stop;

- If $\Omega_{k+1} \neq X^D$ and $V=\emptyset$, then set $k=k+1$, $\Omega_{k+1}=\Omega_k \cup X^D_m$, $\Pi(D)=\infty$, and stop;

- If $\Omega_{k+1} \neq X^D$ and $V \neq \emptyset$, then set $k=k+1$, and go to iteration step.

Remark 14 Note that Algorithm 6 computes the region of attraction of X_m^D in Ω_{k+1}, i.e., upon its termination, we have $\Omega_{k+1} = \Omega(X_m^D)$ (following from the algorithm for computation of region of attraction and its proof given in [6]). Also note that when Algorithm 6 terminates with $\Pi(D) = \infty$, we have $\Omega_{k+1} = \Omega(X_m^D) \neq X^D$.

Before we prove the correctness of Algorithm 6, we present an example to aid the understanding of the algorithm.

Example 6 Consider a directed plant G^D represented by the weighted digraph of Figure 4.3. As Algorithm 6 proceeds, it iteratively computes the values of Ω_k, which is shown in Figure 4.3 as the set of states encircled. For each state x in the plant, the value of $\rho(x)$ is also shown beside x in parentheses.

Initiation:

Since $X_{tm}^D = \emptyset$ and $X_m^D = \{x_2\}$, $\Omega_0 = \emptyset$, and $\lambda(x_2) = 0$.

Iteration 1:

x_1 is the only 1-step $(\Omega_0 \cup X_m^D)$-attractable state, and so $V = \{x_1\}$. We also obtain $\rho(x_1) = 2$. Therefore, $\Omega_1 = \Omega_0 \cup V = \{x_1\}$ and $\lambda(x_1) = 2$.

Iteration 2:

x_0 and x_3 both are 1-step $(\Omega_1 \cup X_m^D)$-attractable. From the iteration step 1, $V = \{x_0, x_3\}$, $\rho(x_0) = 5$ and $\rho(x_3) = 6$. From the iteration step 2, $\Omega_2 = \Omega_1 \cup V = \{x_0, x_1, x_3\}$, and $\lambda(x_0) = 5, \lambda(x_3) = 6$.

Iteration 3:

x_2 is 1-step $(\Omega_2 \cup X_m^D)$-attractable, and $V = \{x_2\}$, $\rho(x_2) = 9$. Therefore, $\Omega_3 = \Omega_2 \cup V = X^D$.

Termination:

Since $\Omega_3 = X^D$, the algorithm terminations, giving $\Pi(D) = \max_{v \in X^D} \rho(v) = \rho(x_2) = 9$, i.e., the cost of this director is 9.

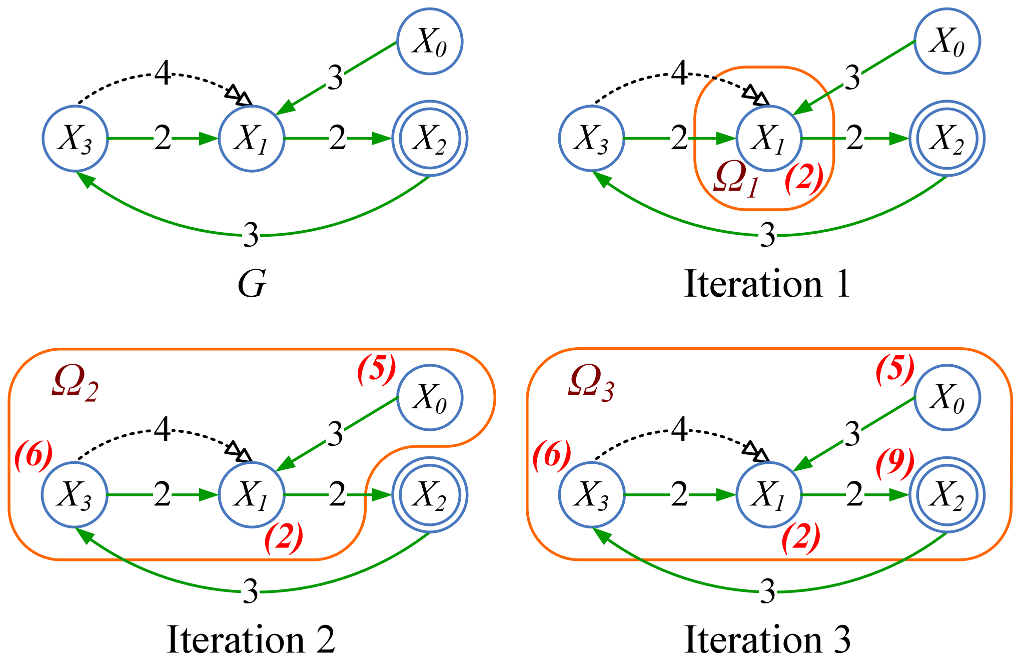


Figure 4.3 Computing the cost of a given director

We next prove the correctness of Algorithm 6. We need the following lemma which states that $\rho(v)$ computes the distance from v to the marked frontier states in G^D.

Lemma 3 For any iteration step k of Algorithm 6, it holds that $\forall v \in \Omega_k : \rho(v) = d_{G^D}(v)$.

Proof: We prove the lemma by induction on iteration steps of Algorithm 6. Since $\Omega_0 = X^D_{tm}$, and $\rho(v) = 0$ for all $v \in X^D_{tm}$, the base step trivially holds. Assume for induction that the lemma holds for the jth step, and consider a state $v \in \Omega_{j+1}$. Then from the definition of 1-step attractability, for every edge $e = (v, \sigma, v') \in E^D$, we have $v' \in \Omega_j \cup X^D_m$. From the induction hypothesis, we have

$$\rho(v') = d_{G^D}(v') \tag{4.4}$$

if $v' \in \Omega_j$. Further, if $v' \in \Omega_j - X^D_m$, then from the construction of the algorithm, $\lambda(v') = \rho(v')$, and so from Equation (4.4), $\lambda(v') = d_{G^D}(v')$ for each $v' \in \Omega_j - X^D_m$. On the other hand, if $v' \in X^D_m$, then $\lambda(v') = 0$. Using these values of $\lambda(v')$, and applying definition of $\rho(v)$ in iteration step 1, we obtain:

$$\begin{aligned}\rho(v) &= \max\{\lambda(v') + c(e) \mid e = (v, \sigma, v') \in E^D\}\\ &= \max\{d_1, d_2\},\end{aligned}$$

where

$$d_1 := \max\{d_{G^D}(v') + c(e) \mid e = (v, \sigma, v') \in E^D, v' \notin X_m^D\}$$

$$d_2 := \max\{0 + c(e) \mid e = (v, \sigma, v') \in E^D, v' \in X_m^D\}.$$

Then the result follows from Equation (4.1). ■

The following theorem establishes the correctness of Algorithm 6.

Theorem 6 It holds that, $\Pi(D) = P(D)$.

Proof: Consider first when Algorithm 6 terminates with $\Omega_{k+1} = X^D$. Then from Lemma 3, for each $v \in X^D$, $\rho(v) = d_{G^D}(v)$. So,

$$\Pi(D) = \max_{v \in X^D} \rho(v) = \max_{v \in X^D} d_{G^D}(v) = P(D),$$

where the last equality follows from Equation (4.2).

On the other hand, if Algorithm 6 does not terminate with $\Omega_{k+1} = X^D$, (so that $\Pi(D) = \infty$), then there exists a state $v \in X^D$ that is not attractable to X_m^D. So, either there exists a cycle of states in $X^D - \Omega(X_m^D)$ containing v, or there is no path from v to any states in X_m^D. In the first case, given any finite number, there exists a path starting from v, such that the cost of the path becomes infinite; in the second case, D is a blocking director. Hence, $P(D) = \infty = \Pi(D)$ for either of the cases. ■

Remark 15 At each iteration, at least one state is included in Ω_k and so there is a maximum of $|X^D|$ iterations. In addition, for every iteration, we perform certain computations for all those states that are 1-step $(\Omega_k \cup X_m^D)$-attractable. The complexity of this step is linear in the number of such states and the number of their edges. So, the overall complexity of Algorithm 6 is $O(|X^D| \times |\Sigma|)$.

The following theorem shows that a director has a finite cost if and only if the region of attraction of marked states in G^D is the entire state space X^D. So, if a director is such that $X^D \not\subseteq \Omega(X_m^D)$, then it can not be an optimal one.

Theorem 7 Given a directed plant G^D, $P(D)$ is finite if and only if $X^D \subseteq \Omega(X_m^D)$.

Proof: For sufficiency, note that upon the termination of Algorithm 6, we have $\Omega_{k+1} = \Omega(X_m^D)$. So, if $X^D \subseteq \Omega(X_m^D)$, then Algorithm 6 terminates with $\Omega_{k+1} = X^D$, and at that step $P(D) = \Pi(D)$ is finite.

For necessity, if $P(D) = \Pi(D)$ is finite, then Algorithm 6 terminates with $\Omega_{k+1} = X^D$. Since Ω_{k+1} consists of all the attractable states of X_m^D in G^D, we conclude that $X^D \subseteq \Omega(X_m^D)$, as desired. ■

The following theorem follows from Theorem 7 and provides a test for the existence of an optimal director (of finite cost). We need to introduce the notion of invariance first.

Definition 13 Given $\hat{X} \subseteq X$, $\hat{X}$ is called *invariant* if $\alpha(\hat{X}, \Sigma_u) \subseteq \hat{X}$ and $\forall x \in \hat{X} \cap X_e$: $\alpha(x, \Sigma_c) \cap \hat{X} \neq \emptyset$.

Note that $\hat{X} \subseteq X$ is invariant if and only if there exists a director D such that under the control of D, all the state transitions of $\hat{X}$ are confined within $\hat{X}$.

Theorem 8 Given a plant G, there exists an optimal director (with finite cost) if and only if there exists $\hat{X}_m \subseteq X_m$ s.t. $x_0 \in \Omega_d(\hat{X}_m)$ and $\Omega_d(\hat{X}_m)$ is invariant.

Proof: For sufficiency, suppose $x_0 \in \Omega_d(\hat{X}_m)$ and $\Omega_d(\hat{X}_m)$ is invariant, we can define a director D such that $X^D = \Omega_d(\hat{X}_m)$. Under the control of D, we have $X^D = \Omega(X_m^D)$, where $X_m^D \supseteq \hat{X}_m$. Then it follows from Theorem 7 that $P(D)$ is finite. Hence, an optimal director with finite cost also exists.

For necessity, suppose an optimal director exists, and let it be D. Then $P(D)$ is finite, and so it follows from Theorem 7 that $X^D \subseteq \Omega(X_m^D)$. Since $\Omega(X_m^D) \subseteq \Omega_d(X_m^D)$, we get $X^D \subseteq \Omega_d(X_m^D)$, which implies that $x_0 \in \Omega_d(X_m^D)$ and $\Omega_d(X_m^D)$ is invariant. So the result follows by letting $\hat{X}_m = X_m^D$. ■

Remark 16 Theorem 8 requires the computation of region of directive attraction. By replacing the phrase "1-step $(\Omega_k \cup X_m^D)$-attractable states" with the phrase "1-step directively $(\Omega_k \cup \hat{X}_m)$-attractable states" in Algorithm 6, we can compute the region of directive attraction in linear complexity. However, the number of subsets $\hat{X}_m$ of X_m is exponential in the number of states of X_m. So the complexity of checking the existence of an optimal director by this approach is $O(|X| \times |\Sigma| \times 2^{|X_m|})$, which is linear in the number of states and events of G but exponential in the number of marked states of G. Thus for a plant with a smaller number of marked states, this approach will be more manageable.

CHAPTER 5. OPTIMAL DIRECTOR FOR ACYCLIC PLANT

5.1 Synthesis of optimal director: Acyclic case

Since at any state $x \in X$, a director can enable one of the feasible controllable events in $\Sigma_c(x)$ or none of such events, there are $|\Sigma_c(x)| + 1 \leq |\Sigma_c| + 1$ choices for control per state. It is clear that the total number of state-based directors is upper bounded by $(|\Sigma_c| + 1)^{|X|}$, whereas the complexity of computing the cost of any given director D is known to be $O(|X^D| \times |\Sigma|)$, as shown in Chapter 4. So if an optimal director is identified by way of enumerating all directors and comparing their costs, the complexity of such an exhaustive search will be exponential in number of plant states. We show in this section that for acyclic plants, it is possible to compute an optimal director with complexity that is polynomial in the number of plant states.

We first show that an optimal director for a trim acyclic G always exists. We need the following lemma which states that the region of attraction of the terminating marked states in G is the entire state space X.

Lemma 4 For a trim acyclic G, it holds that $X \subseteq \Omega(X_{tm})$.

Proof: Since G is acyclic and has a finite state space, it suffices to show that for any state v in G, all paths from v can be extended to end in X_{tm}. Since, G is trim, $X_t \subseteq X_m$, and so $X_{tm} = X_t$. Pick a state v in G and a path p starting from v. If p ends in X_{tm}, then we are done, otherwise, suppose p ends in state $v_1 \notin X_{tm}$. Then since $X_{tm} = X_t$,

v_1 is not a terminating state, and further from acyclicity of G, $v_1 \neq v$. Since v_1 is not terminating, there exists an edge $e = (v_1, \sigma, v_2) \in E$ such that either $v_2 \in X_{tm}$, or v_2 is not a terminating state and $v_2 \neq v_1$. From acyclicity of G, it also follows that $v_2 \neq v_1 \neq v$. Continuing this line of reasoning, and invoking the finiteness of $|X|$, we can extend p such that it ends in X_{tm}. ■

The following corollary follows from Lemma 4 and establishes the existence of an optimal director.

Corollary 1 If G is trim, acyclic, and has finite states, then an optimal director for G exists.

Proof: From Lemma 4, $X \subseteq \Omega(X_{tm})$. Since $\Omega(X_{tm}) \subseteq \Omega_d(X_{tm})$, we get $X \subseteq \Omega(X_{tm}) \subseteq \Omega_d(X_{tm})$. This implies $x_0 \in \Omega_d(X_{tm})$, and further since $\Omega_d(X_{tm}) \supseteq X$, $\Omega_d(X_{tm})$ is invariant. So, from Theorem 8 it follows that an optimal director exists. ■

The following algorithm computes an optimal director, denoted D^*, which is also a DP-optimal director.

Algorithm 7 Input a trim acyclic plant G, the following steps computes an optimal director D^*.

Initiation:

Set $k = 1$, $\Omega_k = X_{tm}$, $\forall x \in \Omega_k : D^*(x) = \emptyset, \rho(x) = 0, \Pi(D^* \backslash x) = 0$, and $\forall x \in X_m$: $\lambda(x) = 0$.

Iteration:

1. Let U be the set of all 1-step Ω_k-attractable states, and for each $x \in U$, compute the control action:

$$D^*(x) = \begin{cases} \{\sigma^*\} & x \in X_e \text{ or } \Sigma_u(x) = \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

where $e^* = (x, \sigma^*, x') \in E_c(x)$ is an edge that belongs to the argument of

$$\min_{e \in E_c(x)} \{\max(\lambda(x') + c(e), \Pi(D^* \backslash x')) \mid e = (x, \sigma, x')\}$$

(Note that choice of σ^* in definition of $D^*(x)$ need not be unique, indicating non-uniqueness of optimal director.)

2. For each $x \in U$ compute:

$$\rho_1(x) = \begin{cases} \lambda(x') + c(e^*) & x \in X_e \text{ or } \Sigma_u(x) = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$\rho_2(x) = \begin{cases} \max\limits_{e \in E_u(x)} \{\lambda(x') + c(e) \mid e = (x, \sigma, x')\} & E_u(x) \neq \emptyset \\ 0 & E_u(x) = \emptyset \end{cases}$$

$$\rho(x) = \max(\rho_1(x), \rho_2(x))$$

3. For each $x \in U$, compute the set of edges in the optimally directed graph and the cost of the optimal director rooted at state x:

$$E^{D^*}(x) = \{(x, \sigma, x') \in E \mid \sigma \in D^*(x) \cup \Sigma_u(x)\}$$

$$\Pi(D^* \backslash x) = \max\{\max_{x' \in \Omega_k} \{\Pi(D^* \backslash x') \mid (x, \sigma, x') \in E^{D^*}(x)\}, \rho(x)\}$$

4. Set $\Omega_{k+1} = \Omega_k \cup U$, and for each $x \in U - X_m$, let $\lambda(x) = \rho(x)$.

Termination:

- If $x_0 \in \Omega_{k+1}$, then stop and $G^{D^*} := (X^{D^*}, E^{D^*}, X_m^{D^*})$ is the optimally directed plant, where $X^{D^*} := \Omega_{k+1}$ and $X_m^{D^*} := \Omega_{k+1} \cap X_m$.

- If $x_0 \notin \Omega_{k+1}$, then set $k = k + 1$, and go to iteration step.

Remark 17 It follows from analysis similar to that given in Remark 15 that the computational complexity of Algorithm 7 is $O(|X| \times |\Sigma|)$.

Before we prove the correctness of Algorithm 7, we present an example to aid the understanding of the algorithm.

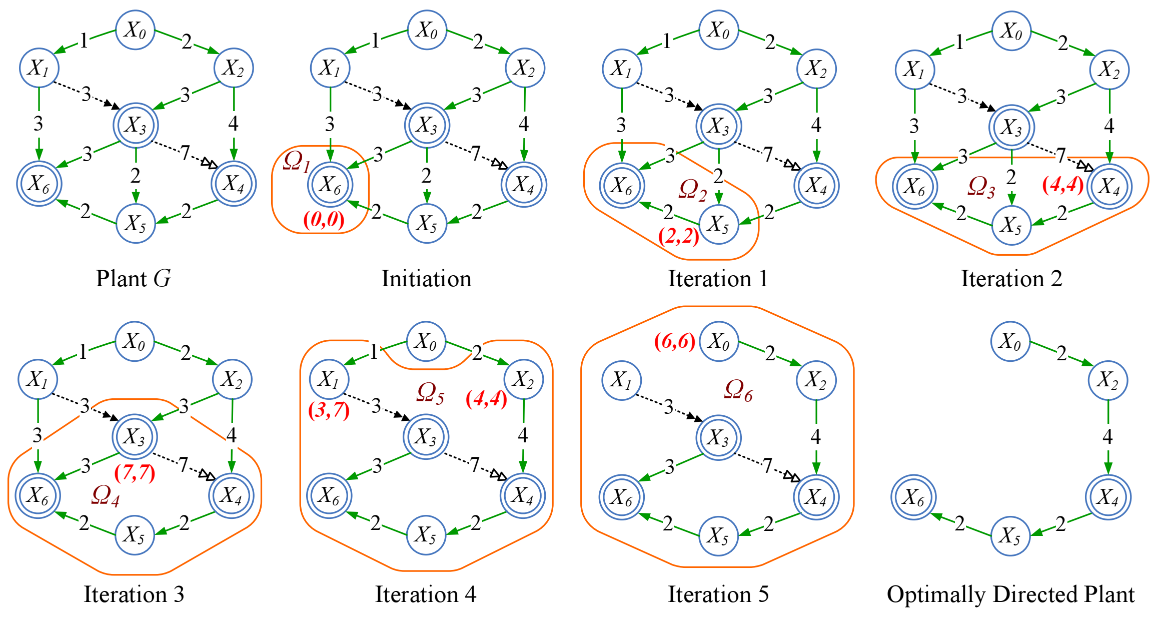


Figure 5.1 Illustration of Algorithm 7

Example 7 The plant is shown in Figure 5.1. Note that edges with dashed line and double *solid* arrows represent the transitions on sensor output while those with dashed line and double *hollow* arrows represent the transitions on disturbance input. As Algorithm 7 proceeds, it iteratively computes the values of Ω_k, which is shown in Figure 5.1 as the set of states encircled. For each state x in the plant, the values of $\rho(x)$ and $\Pi(D^*\backslash x)$ are also shown beside x as a pair of numbers in parentheses.

Initiation:

Since $X_{tm} = \{x_6\}$ and $X_m = \{x_6, x_4, x_3\}$, $\Omega_1 = \{x_6\}$, $D^*(x_6) = \emptyset$, $\rho(x_6) = 0$, $\Pi(D^*\backslash x_6) = 0$ and $\lambda(x_6) = \lambda(x_4) = \lambda(x_3) = 0$.

Iteration 1:

The state x_5 is the only *1-step* Ω_1-attractable state. Therefore $U = \{x_5\}$. Since no uncontrollable events exist at x_5, $\rho(x_5) = \rho_1(x_5) = 2$, and the control action at x_5 is the selection of the event causing the transition from x_5 to x_6. We also obtain $\Pi(D^*\backslash x_5) = 2$, $\Omega_2 = \{x_6, x_5\}$ and $\lambda(x_5) = \rho(x_5) = 2$.

Iteration 2:

The only 1-step Ω_2-attractable state is x_4, and reasoning similarly as in the previous iteration, we get $\rho(x_4) = 4$, and control action at x_4 is the selection of the event causing the transition from x_4 to x_5. We also obtain $\Pi(D^*\backslash x_4) = 4$, $\Omega_3 = \{x_6, x_5, x_4\}$.

Iteration 3:

x_3 is the only 1-step Ω_3-attractable state, and therefore $U = \{x_3\}$. $\rho_1(x_3) = 3$, the director selects the controllable transition from x_3 to x_6 even its cost is bigger than that of the transition from x_3 to x_5. It is clear that the uncontrollable event causing the transition from x_3 to x_4 remains enabled, and $\rho_2(x_3) = 7$. So $\rho(x_3) = \max(\rho_1(x_3), \rho_2(x_3)) = \rho_2(x_3) = 7$. $\Pi(D^*\backslash x_3)$ is the maximum of the values $\rho(x_3), \Pi(D^*\backslash x_4)$, and $\Pi(D^*\backslash x_6)$, which is $\rho(x_3)$. Accordingly $\Pi(D^*\backslash x_3) = 7$. Finally $\Omega_4 = \{x_3, x_4, x_5, x_6\}$.

Iteration 4:

Both the states x_1 and x_2 are 1-step Ω_4-attractable. Because $x_1 \notin X_e$ and $\Sigma_u(x_1) \neq \emptyset$, $\rho_1(x_1) = 0$ while $\rho_2(x_1) = \lambda(x_3) + c(e') = 3$, where e' is the uncontrollable transition from x_1 to x_3. So, $\rho(x_1) = \max(0, 3) = 3$. The control action at x_1 is to disable all the controllable transitions. On the other hand, $\rho(x_2) = 4$. Note that the transition from x_2 to x_4 is enabled even its cost is bigger than that of the transition from x_2 to x_3. This is because the cost of director $D\backslash x_2$ is dominated by the cost of its sub-directors instead of x_2's distance to the marked frontier states. Next, $\Pi(D^*\backslash x_1) = 7$, which is the maximum of the values $\rho(x_1)$ and $\Pi(D^*\backslash x_3)$. Similarly, $\Pi(D^*\backslash x_2) = 4$, which is the maximum of the values $\rho(x_2)$ and $\Pi(D^*\backslash x_4)$. Finally, $\Omega_5 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $\lambda(x_1) = \rho(x_1) = 3$, and $\lambda(x_2) = \rho(x_2) = 4$.

Iteration 5:

x_0 is 1-step Ω_5-attractable. Again, the cost of director $D\backslash x_0 = D$ is dominated by the cost of its sub-directors instead of x_0's distance to the marked frontier states. So the control action at x_0 is to select the controllable edge from x_0 to x_2 and we have $\rho(x_0) = \rho_1(x_0) = 6$. Also, $\Omega_6 = X$ and $\Pi(D^*\backslash x_0) = 6$.

Termination:

Since $x_0 \in \Omega_6$, the algorithm terminates, and the cost of an optimal director is 6. The final optimally directed system is shown in Figure 5.1.

In the remainder of this section we prove the correctness of Algorithm 7. We first show that Algorithm 7 terminates.

Theorem 9 Algorithm 7 terminates, i.e., $x_0 \in \Omega_{k+1}$ for some $k \geq 1$.

Proof: It suffices to show that at each iteration step $k \geq 1$ of Algorithm 7, at least one state is added in Ω_{k+1}. (So, from finiteness of $|X|$, $x_0 \in \Omega_{k+1}$ for some $k \geq 1$.) For $k = 1$, $\Omega_1 = X_{tm} \neq \emptyset$ since G is trim. Pick any $k > 1$, and consider a state $v \notin \Omega_k$. From Lemma 4, every path starting from v ends in X_{tm} within a finite number of steps. Since $X_{tm} \subseteq \Omega_k$, every path starting from v ends in Ω_k within a finite number of steps, for example, within at most N_v steps. Then every path from a successor of v ends in Ω_k within at most $N_v - 1$ steps (since there are no cycles in G). Continuing this argument further yields that there is a state reachable from v and it is 1-step attractable to Ω_k. Then this is a state that is added to Ω_{k+1} in the kth iteration step. ∎

We next show that for any $v \in X^{D^*}$, $\Pi(D^*\backslash v)$ indeed equals the cost of director $D^*\backslash v$. We need the result of the following lemma, whose proof is analogous to that of Lemma 3, and hence omitted.

Lemma 5 For all $v \in X^{D^*}$, it holds that $\rho(v) = d_{G^{D^*}}(v)$.

The following corollary is an immediate consequence of Lemma 5 and Theorem 6.

Corollary 2 For all $v \in X^{D^*}$, it holds that $P(D^*\backslash v) = \Pi(D^*\backslash v) = \max_{u \in X^{D^*\backslash v}} \rho(u)$.

By replacing v with x_0 in the above corollary, and noting that $D^*\backslash x_0 = D^*$, we obtain: $P(D^*) = \max_{u \in X^{D^*}} \rho(u)$.

Thus far we have shown that Algorithm 7 terminates, and correctly computes the cost of director D^* that is constructed. It remains to show that D^* is optimal, which we establish next.

Theorem 10 Given a plant G, the director D^* computed by Algorithm 7 is a DP-optimal director for G.

Proof: We need to show that for any $v \in X^{D^*}$, the director $D^*\backslash v$ is a DP-optimal director rooted at v. Since $v \in X^{D^*}$, there exists $k \geq 1$ such that $v \in \Omega_k$. The proof is based upon induction on k. If $k = 1$, then $v \in X_{tm}$, and so clearly $D^*(v) = \emptyset$ is the only and hence a DP-optimal control action at v. Next, suppose $v \in \Omega_{k+1}$ for some $k \geq 1$. From induction hypothesis, $D^*\backslash u$ is a DP-optimal director rooted at u for all $u \in \Omega_k$.

Suppose for contradiction that there exists an optimal director D such that

$$P(D\backslash v) < P(D^*\backslash v) = \Pi(D^*\backslash v), \tag{5.1}$$

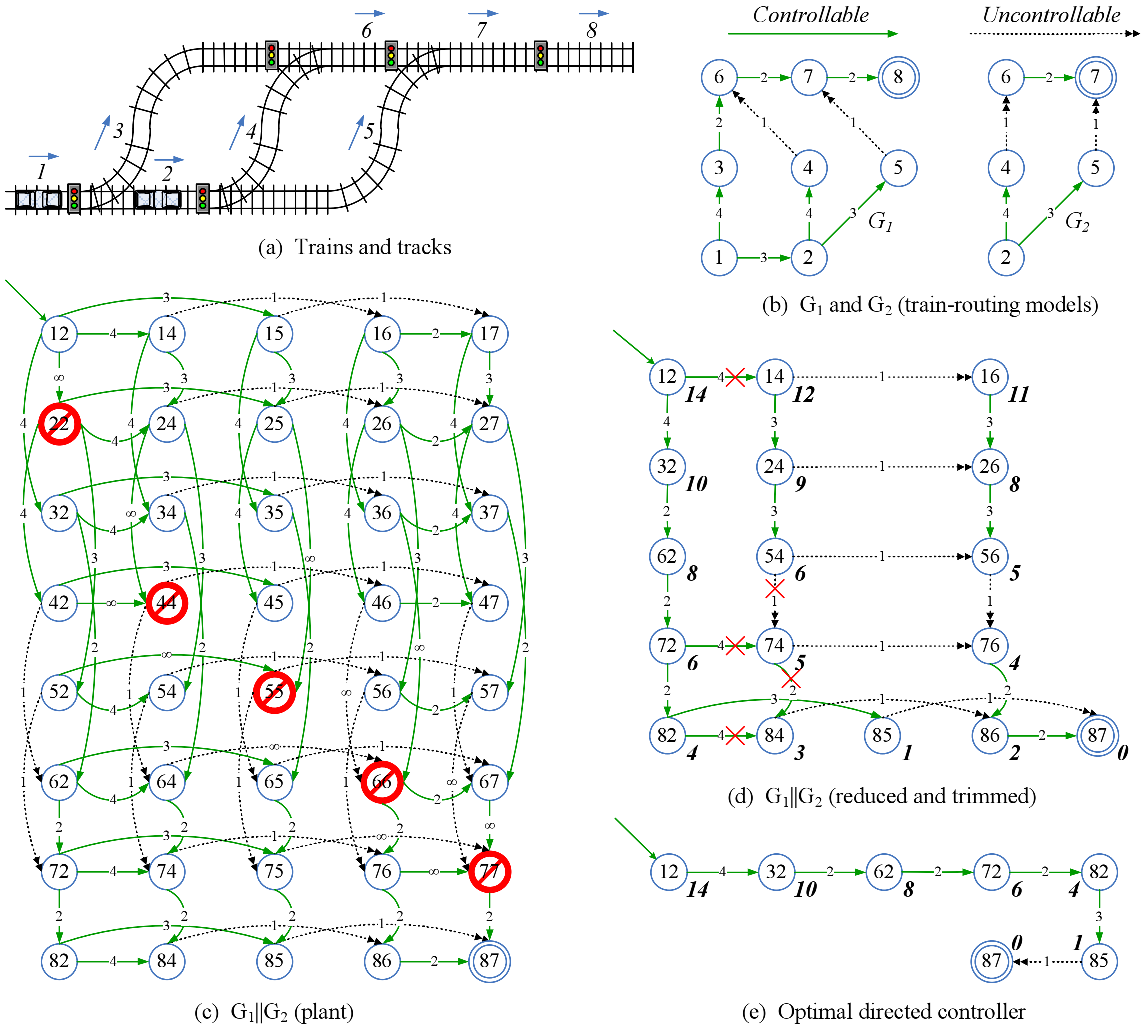
where the equality follows from Corollary 2. It must be that $D(v) \neq D^*(v)$, i.e., the two directors select different control actions at v, otherwise it would follow that $P(D\backslash v) = P(D^*\backslash v)$. When $v \in X_e$ or $\Sigma_u(v) = \emptyset$,

$$\begin{aligned} P(D\backslash v) = \max \Big(\big\{ \max \big(\lambda(v') + c(e), \Pi(D^*\backslash v')\big) \mid \\ e = (v, \sigma, v') \in E_c(v) \text{ and } D(v) = \{\sigma\} \big\}, \rho_2(v) \Big) \end{aligned}$$

$$P(D^*\backslash v) = \max \Big(\min_{e \in E_c(v)} \big\{ \max \big(\lambda(v') + c(e), \Pi(D^*\backslash v')\big) \mid e = (v, \sigma, v') \big\}, \rho_2(v) \Big)$$

When $v \notin X_e$ and $\Sigma_u(v) \neq \emptyset$, $P(D\backslash v) = P(D^*\backslash v) = \rho_2(v)$. In both cases, $P(D\backslash v) \geq P(D^*\backslash v)$, which is a contradiction. ∎

5.2 Application example


(a) Trains and tracks

(b) G_1 and G_2 (train-routing models)

(c) $G_1 \| G_2$ (plant)

(d) $G_1 \| G_2$ (reduced and trimmed)

(e) Optimal directed controller

Figure 5.2 Synthesis of an optimal director for the application example

We provide an application example to demonstrate our result. The application is of train-traffic control over a set of track sections. As shown in Figure 5.2(a), eight sections of tracks labeled from 1 to 8 are separated by some traffic lights and switches. In each section, trains can only travel in the directions as indicated by the arrows. Suppose initially two trains T_1 and T_2 are in the section 1 and 2, respectively. We are required to synthesize an optimal director to control the traffic lights and switches to ensure T_1

and T_2 eventually go to the section 8 and 7, respectively.

We model the movement of T_1 and T_2 by two automata G_1 and G_2, respectively, shown in the Figure 5.2(b). The cost associated with each edge in G_1 (resp. G_2) is the amount of time taken by the train T_1 (resp. T_2) to go from the source section to the destination section of the edge. Note that since both trains can only move forward, both G_1 and G_2 are acyclic. Also note that the way traffic lights and switches are placed, the events corresponding to transitions between the section 4 and 6, and between the section 5 and 7 for both trains are uncontrollable. Also these uncontrollable events are not disturbance inputs, and represent elements of the set $\Sigma_u - \Sigma_d$.

The combined movement of two trains is given by the composition of automata G_1 and G_2, denoted as $G_1||G_2$. The corresponding automaton is shown in Figure 5.2(c), where the state "12" is the initial state, the state "87" is the only marked state, and the scenarios that two trains are in the same section are represented by illegal states, which are identified by having costs of all incoming edges to be infinity. Note that in the composed automaton, each edge represents the movement of one of the two trains, and so the cost of such an edge is taken to be the same as the cost of corresponding edge in the corresponding train model. Also since both G_1 and G_2 are acyclic, so is $G_1||G_2$.

We apply Algorithm 5 on $G_1||G_2$ and obtain the automaton shown in Figure 5.2(d). This automaton has no infinite-cost edges and obviously is acyclic. Algorithm 7 is then applied to compute an optimal director. The synthesis process is also shown in Figure 5.2(d) where distance from each state to its marked frontier states has been identified. After trimming, the resulting optimal director is obtained and shown in Figure 5.2(e).

The control strategy implemented by this optimal director is summarized as follows. We first guide T_1 from the section 1 to the section 8 via the section 3, 6 and 7. Then we guide T_2 from the section 2 to the section 7 via the section 5. The cost of this director is shown to be 14.

CHAPTER 6. OPTIMAL DIRECTOR FOR GENERAL PLANT

6.1 Synthesis of optimal director for general plant

An algorithm (Algorithm 7) to compute an optimal director for acyclic plants with complexity that is polynomial in the number of plant states was presented in the previous chapter. It is based on the observation that the region of attraction of the set of terminating marked states is the whole state set in a trim acyclic plant, i.e., $X \subseteq \Omega(X_{tm})$. Algorithm 7 constructs the region of attraction of X_{tm} iteratively: it starts with $\Omega_1 = X_{tm}$; in the kth iteration, all the states that are *1-step attractable* to Ω_k are combined with Ω_k to form the state set Ω_{k+1}. For each added state x, Algorithm 7 optimizes the control action by comparing all the feasible transitions from x to the states in Ω_k. Note that the value of $\rho(x)$ and $\Pi(D^*\backslash x)$ computed by Algorithm 7 represents x's distance (to its marked frontier states) in G^{D^*} and the cost of the director rooted at x, respectively.

In general, a plant model can contain cycles and in which case Algorithm 7 is not applicable. This situation is illustrated by the following examples.

Example 8 Consider a plant G shown in Figure 6.1(a). Algorithm 7 starts with $\Omega_1 = \{x_3\}$. Due to the cycle between x_1 and x_2, neither x_1 nor x_2 is 1-step attractable to Ω_1 and thus Algorithm 7 can not proceed or terminate because $X \subseteq \Omega(X_{tm})$ no longer holds in this case. An optimal director for G, however, does exist, which yields the optimally directed plant as shown in Figure 6.1(b) with the director cost of 4.

Example 9 Consider another plant G shown in Figure 6.1(c). In this case there is no terminating marked state and so Algorithm 7 does not even start. Suppose we let Algorithm 7 start from all marked states (including the non-terminating marked states), then since both x_1 and x_2 are marked states, we will have $\Omega_1 = \{x_1, x_2\}$. Then Algorithm 7 will compute the values of $\rho(x)$ and $\Pi(D^*\backslash x)$ for each state x. Each state in Figure 6.1(c) is labeled with such a pair of numbers. When Algorithm 7 terminates, it will disable the transition from x_0 to x_1 while enable that from x_0 to x_2. The resulting directed plant is shown in Figure 6.1(d) with the director cost of 9. However, the optimally directed plant should be the one shown in Figure 6.1(e), with the director cost of 7.

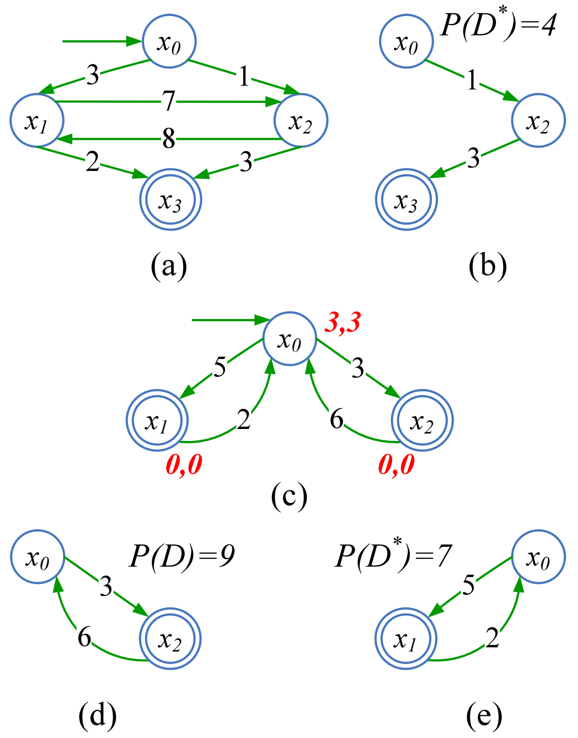


Figure 6.1 Application of Algorithm 7 on plants with cycles

Note that Algorithm 7 is “greedy” or “locally optimal” in the sense that when computing $\Omega_2 = \Omega_1 \cup \{x_0\}$, it picks the edge from x_0 to x_2 with the cost of 3 instead of the edge from x_0 to x_1 with the cost of 5. This ignores the fact that the distance of x_2 is 6, larger than the distance of x_1, which is 2. Since the cost of a director is determined by the worst distance among those of the initialization states (see Equation 4.3), the above "locally optimal" selection is not "globally optimal". If, however, the set of marked states to be present in a trim optimally directed plant could be identified, then Algorithm 7 could be forced to consider only those marked states. For example, by omitting x_2 in Ω_1, Algorithm 7 would be forced to pick the edge from x_0 to x_1 and yields an optimal director. Thus by trying all subsets of X_m as a choice for Ω_1 and applying a locally optimal algorithm, a globally optimal director could be obtained. However, such a strategy could have complexity that is exponential in the number of marked states. We will show that not all subsets of X_m need to be considered as a choice for Ω_1, which is the key to a polynomial complexity algorithm.

We present a two-step algorithm in this paper. We first provide a synthesis algorithm (Algorithm 8) that is applicable for general plant models, cyclic or acyclic. The director computed by Algorithm 8 is "locally optimal" in the sense that the distance of each state under directed control is minimized. We then present another algorithm (Algorithm 9) which iteratively applies Algorithm 8 on different input plants starting from G and compares the resulting directors from various iterations. During each iteration, Algorithm 9 removes the states with the maximum distance to produce the input plant for the next iteration. It will be shown that among all the directors computed during iterations of Algorithm 9, the one with the minimum cost is indeed optimal.

It will also be shown that Algorithm 8 is of polynomial complexity in the number of plant states, and the number of times in which Algorithm 8 is executed inside Algorithm 9 is at most one more than the number of non-terminating marked states. So the overall complexity of synthesis of an optimal director remains polynomial in the plant size.

We first present Algorithm 8.

Algorithm 8 Input a plant $G := (X, E, X_m)$, the following steps compute a "locally optimal" director $D^\diamond$ and the distance of each state $x \in X$ under the control of $D^\diamond$.

Initiation:

Set $k = 1$, $\Omega_k = X_m$, $\Omega'_k = X_{tm}$, $\forall x \in \Omega_k : \lambda(x) = 0$ and $\forall x \in \Omega'_k : D^\diamond(x) = \emptyset$ and $\rho(x) = 0$.

Iteration:

1. Let U_k be the set of all 1-step *directively* Ω_k-attractable states excluding those in Ω'_k, i.e.,

$$U_k := \{x \in X - \Omega'_k \mid \alpha(x, \Sigma) \cap \Omega_k \neq \emptyset, \alpha(x, \Sigma_u) \subseteq \Omega_k \text{ and } x \in X_e \Rightarrow \alpha(x, \Sigma_c) \cap \Omega_k \neq \emptyset\}$$

2. For each $x \in U_k$, we define

$$E_{c_k}(x) := \{e = (x, \sigma, x') \in E_c(x) \mid x' \in \Omega_k\}$$

and compute the following:

$$D_k(x) = \begin{cases} \{\sigma_k\} & x \in X_e \text{ or } \Sigma_u(x) = \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

such that $e_k = (x, \sigma_k, x') \in E_{c_k}(x)$ is *any* edge that belongs to the argument of

$$\min_{e \in E_{c_k}(x)} \{\lambda(x') + c(e) \mid e = (x, \sigma, x')\}$$

$$\rho_k^c(x) = \begin{cases} \lambda(x') + c(e_k) & x \in X_e \text{ or } \Sigma_u(x) = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$\rho_k^u(x) = \begin{cases} \max\limits_{e \in E_u(x)} \{\lambda(x') + c(e) \mid e = (x, \sigma, x')\} & E_u(x) \neq \emptyset \\ 0 & E_u(x) = \emptyset \end{cases}$$

$$\rho_k^\diamond(x) = \max(\rho_k^c(x), \rho_k^u(x))$$

$$E^{D_k}(x) = \{(x, \sigma, x') \in E \mid \sigma \in D_k(x) \cup \Sigma_u(x)\}$$

3. Let $V_k \subseteq U_k$ be the set of states that belong to the argument of $\min_{x \in U_k} \rho_k^\diamond(x)$. For each state $x \in V_k$, set $\rho(x) = \rho_k^\diamond(x)$, $D^\diamond(x) = D_k(x)$ and $E^{D^\diamond}(x) = E^{D_k}(x)$. If $x \notin X_m$, then let $\lambda(x) = \rho(x)$.

4. Set $\Omega_{k+1} = \Omega_k \cup V_k$ and $\Omega'_{k+1} = \Omega'_k \cup V_k$.

Termination:

- If $U_k = \emptyset$, then stop. Set $\rho(x) = \infty$ for each state $x \in X - \Omega'_k$ and define $P(D^\diamond) := \max_{x \in I(G^{D^\diamond})} \rho(x)$.
- If $U_k \neq \emptyset$, then continue iteration with $k = k + 1$.

Remark 18 At each iteration, at least one state is added into Ω'_k and so there is a maximum of $|X|$ iterations. Also for each iteration, we perform certain computations for those states that are 1-step directively Ω_k-attractable. The complexity of this step is linear in the number of such states and the number of their edges. So, the overall complexity of Algorithm 8 is $O(|X| \times |\Sigma|)$.

We present an example to aid the understanding of Algorithm 8.

Example 10 The plant is shown in Figure 6.2, where an edge with double arrows represents a transition on an uncontrollable event and an edge with single arrow represents a transition on a controllable event. Note that all the uncontrollable events in the plant are disturbance inputs, so we have $X_e = \{x_6\}$ and the only controllable transition at x_6 should not be disabled by any director. As Algorithm 8 proceeds, it iteratively computes the values of Ω_k, which is shown in Figure 6.2 as the set of states encircled. For each state x in the plant, the values of $\lambda(x), \rho(x)$ are also shown as a pair beside the state x.

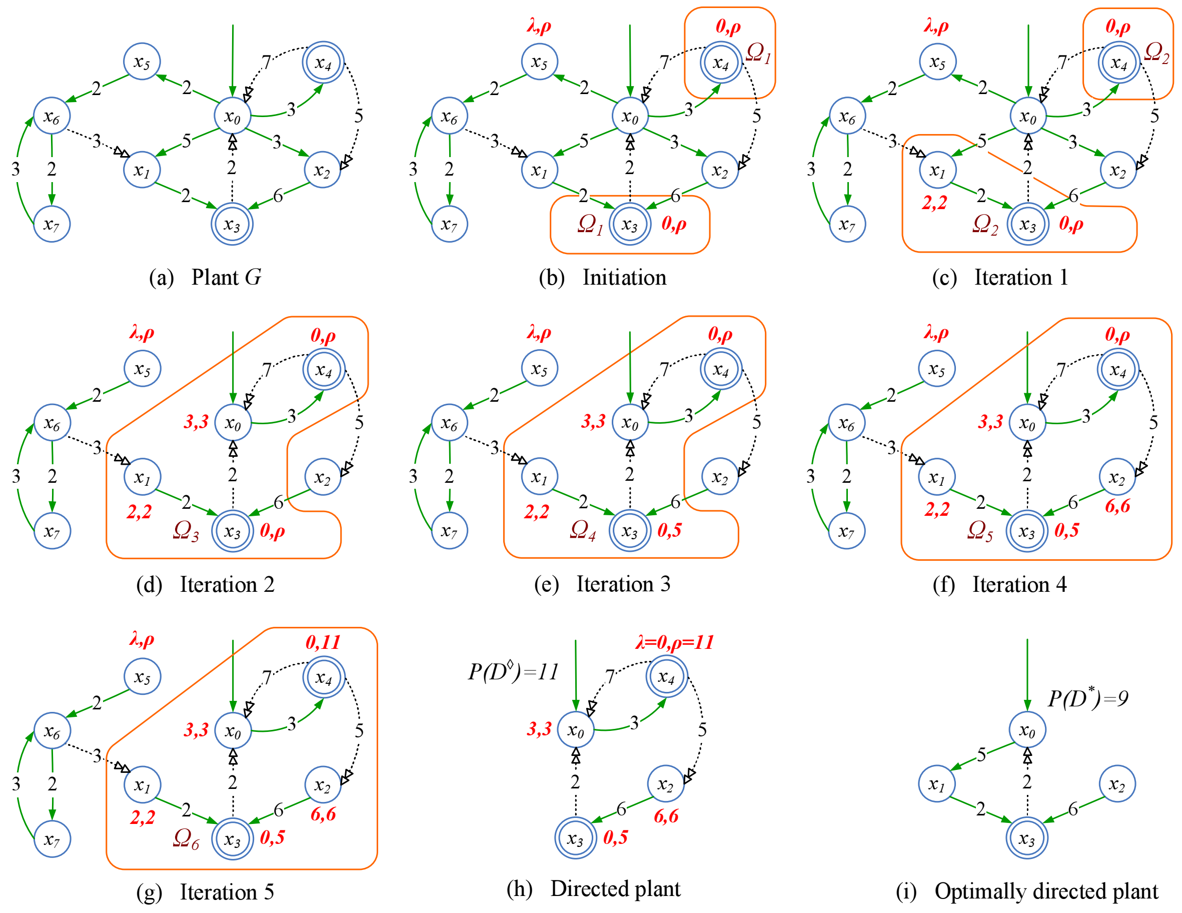

(a) Plant G (b) Initiation (c) Iteration 1 (d) Iteration 2 (e) Iteration 3 (f) Iteration 4 (g) Iteration 5 (h) Directed plant (i) Optimally directed plant

Figure 6.2 Example for Algorithm 8

Initiation:

Since $X_{tm} = \emptyset$ and $X_m = \{x_3, x_4\}$, $\Omega_1 = \{x_3, x_4\}$, $\Omega_1' = \emptyset$ and $\lambda(x_3) = \lambda(x_4) = 0$.

Iteration 1:

The states x_0, x_1 and x_2 are all *1-step* directively attractable to Ω_1. Therefore $U_1 = \{x_0, x_1, x_2\}$. Since $\rho_1^{\diamond}(x_1)$ is the minimum among $\rho_1^{\diamond}(x_0)$, $\rho_1^{\diamond}(x_1)$ and $\rho_1^{\diamond}(x_2)$, we get $V_1 = \{x_1\}$ and $\rho(x_1) = \lambda(x_1) = 2$. We have $\Omega_2 = \Omega_1 \cup V_1 = \{x_1, x_3, x_4\}$ and $\Omega_2' = \Omega_1' \cup V_1 = \{x_1\}$.

Iteration 2:

The states x_0 and x_2 are both *1-step* directively attractable to Ω_2. Therefore $U_2 = \{x_0, x_2\}$. Since $\rho_2^{\diamond}(x_0)$ is the minimum between $\rho_2^{\diamond}(x_0)$ and $\rho_2^{\diamond}(x_2)$, we get $V_2 = \{x_0\}$ and

$\rho(x_0) = \lambda(x_0) = 3$. The control selection at x_0 is to select the controllable transition from x_0 to x_4. We have $\Omega_3 = \Omega_2 \cup V_2 = \{x_0, x_1, x_3, x_4\}$ and $\Omega'_3 = \Omega'_2 \cup V_2 = \{x_0, x_1\}$.

Iteration 3:

The states x_2 and x_3 are both *1-step* directively attractable to Ω_3. Therefore $U_3 = \{x_2, x_3\}$. Since $\rho_3^\diamond(x_3)$ is the minimum between $\rho_3^\diamond(x_2)$ and $\rho_3^\diamond(x_3)$, we get $V_3 = \{x_3\}$ and $\rho(x_3) = 5$. We have $\Omega_4 = \Omega_3 \cup V_3 = \{x_0, x_1, x_3, x_4\}$ and $\Omega'_4 = \Omega'_3 \cup V_3 = \{x_0, x_1, x_3\}$.

Iteration 4:

The state x_2 is the only state that is *1-step* directively attractable to Ω_4. Therefore $U_4 = \{x_2\}$, $V_4 = \{x_2\}$ and $\rho(x_2) = \lambda(x_2) = 6$. We have $\Omega_5 = \Omega_4 \cup V_4 = \{x_0, x_1, x_2, x_3, x_4\}$ and $\Omega'_5 = \Omega'_4 \cup V_4 = \{x_0, x_1, x_2, x_3\}$.

Iteration 5:

The state x_4 is the only state that is *1-step* directively attractable to Ω_5. Therefore $U_5 = \{x_4\}$, $V_5 = \{x_4\}$ and $\rho(x_4) = 11$. We have $\Omega_6 = \Omega_5 \cup V_5 = \{x_0, x_1, x_2, x_3, x_4\}$ and $\Omega'_6 = \Omega'_5 \cup V_5 = \{x_0, x_1, x_2, x_3, x_4\}$.

Termination:

Since $U_6 = \emptyset$, the algorithm terminates. Note that $\rho(x_5) = \rho(x_6) = \rho(x_7) = \infty$. The directed plant is shown in Figure 6.2(h) with the control cost of 11. Note that the director computed by Algorithm 8 may not be optimal. The optimally directed plant for this example should be the one shown in Figure 6.2(i) with the control cost of 9.

The function $\rho : X \to \mathcal{R}$ computed by Algorithm 8, where $\mathcal{R}$ denotes non-negative reals including infinity, is called the distance function of $D^\diamond$. The value of $\rho(x)$ for each state $x \in X^{D^\diamond}$ represents the distance of x under the control of $D^\diamond$, i.e., we have following theorem.

Theorem 11 Given a plant $G := (X, E, X_m)$ and consider the notation of Algorithm 8. It holds that $\forall x \in X^{D^\diamond} : \rho(x) = d_{G^{D^\diamond}}(x)$.

Proof: We prove by induction on the iteration ordinal k of Algorithm 8. Since $\Omega'_1 = X_{tm}$ and $\forall x \in \Omega'_1 : D^\diamond(x) = \emptyset$ and $\rho(x) = 0$, the base step trivially holds.

Assume for induction that the theorem holds for the states of Ω'_k, we need to show the theorem holds for the states of $\Omega'_{k+1} = \Omega'_k \cup V_k$.

Consider a state $x \in V_k$. Then from the definition of 1-step directive attractability, for every edge $e = (x, \sigma, x') \in E^{D^\diamond}$, we have $x' \in \Omega_k$. Note that $\Omega_k \supseteq \Omega'_k$ and $\Omega_k - \Omega'_k \subseteq X_m$. So either $x' \in \Omega'_k - X_m$ or $x' \in X_m$.

If $x' \in \Omega'_k - X_m$, then it follows from the induction hypothesis that $\rho(x') = d_{G^{D^\diamond}}(x')$. Furthermore, from the construction of the algorithm, we have $\lambda(x') = \rho(x') = d_{G^{D^\diamond}}(x')$. If $x' \in X_m$, then we have $\lambda(x') = 0$. Using these values of $\lambda(x')$, and applying definition of $\rho(x) = \rho^\diamond_k(x)$, we obtain:

$$\begin{aligned}\rho(x) &= \max\{\lambda(x') + c(e) \mid e = (x, \sigma, x') \in E^{D^\diamond}\} \\ &= \max\{d_1, d_2\}\end{aligned}$$

where

$$d_1 := \max\{d_{G^{D^\diamond}}(x') + c(e) \mid e = (x, \sigma, x') \in E^{D^\diamond}, x' \notin X_m\}$$

$$d_2 := \max\{0 + c(e) \mid e = (x, \sigma, x') \in E^{D^\diamond}, x' \in X_m\}$$

Then the result follows from Equation (4.1). ∎

We claim that for each state $x \in X^{D^\diamond}$, the value of $\rho(x)$ computed by Algorithm 8 is the *minimum* distance of x under directed control. We need the following lemma to establish this claim.

Lemma 6 Consider the notation of Algorithm 8, assume Algorithm 8 terminates after the nth iteration for some $n > 0$ and let $x_k \in V_k$ be any state of V_k, then $0 < \rho(x_1) < \rho(x_2) < \ldots < \rho(x_k) < \ldots < \rho(x_n)$.

Proof: Note that for any two states $\hat{x} \in V_k$ and $\bar{x} \in V_k$, we have $\rho(\hat{x}) = \rho(\bar{x}) = \min_{x \in U_k} \rho^\diamond_k(x)$.

We prove the lemma by induction on the iteration ordinal k of Algorithm 8. We first show that the base step holds, i.e., $0 < \rho(x_1)$. Since only terminating marked states have the distance of 0 and $x_1 \notin \Omega'_k = X_{tm}$, it follows from Theorem 11 that $\rho(x_1) = d_{G^{D^\diamond}}(x_1) > 0$. Assume for induction that the lemma holds after the kth iteration, i.e., $0 < \rho(x_1) < \rho(x_2) < \ldots < \rho(x_k)$. We need to show $\rho(x_k) < \rho(x_{k+1})$.

Clearly, $x_{k+1} \in U_{k+1}$. So either $x_{k+1} \in U_{k+1} \cap U_k$ or $x_{k+1} \in U_{k+1} - U_k$.

For the latter case, x_{k+1} is 1-step directively attractable to Ω_{k+1} but not 1-step directively attractable to Ω_k. So there must exist an edge $e = (x, \sigma, x') \in E^{D^\diamond}(x_{k+1})$ such that $x_{k+1} = x$ and $x' \in V_k - X_m$. Since $x' \notin X_m$, it follows that $\rho(x_{k+1}) > \lambda(x') = \rho(x') = \rho(x_k)$.

For the former case, x_{k+1} is 1-step directively attractable to Ω_k. However, $x_{k+1} \notin V_k$ and $\rho(x_k) = \min_{x \in U_k} \rho^\diamond_k(x)$, so it must be $\rho^\diamond_k(x_{k+1}) > \rho^\diamond_k(x_k) = \rho(x_k)$. The computation of $\rho^\diamond_{k+1}(x_{k+1})$ differs from that of $\rho^\diamond_k(x_{k+1})$ if there is at least one edge $e = (x, \sigma, x') \in E^{D^\diamond}(x_{k+1})$ such that $x_{k+1} = x$ and $x' \in V_k - X_m$. If such an edge exists, then $\rho(x_{k+1}) = \rho^\diamond_{k+1}(x_{k+1}) \geq \max((\lambda(x') + c(e)), \rho^\diamond_k(x_{k+1})) > \rho(x_k)$. If such an edge doesn't exist, then we have $\rho(x_{k+1}) = \rho^\diamond_{k+1}(x_{k+1}) = \rho^\diamond_k(x_{k+1}) > \rho(x_k)$. ■

With Lemma 6 in hand, we are ready to prove our claim in the following theorem.

Theorem 12 Given a plant $G := (X, E, X_m)$ and consider the notation of Algorithm 8. It holds that $\forall x \in X^{D^\diamond}$: $\rho(x) = \min_D d_{G^D}(x)$.

Proof: Suppose $\hat{x}$ is the state that belongs to the argument of $\min_{x \in U_k} \rho^\diamond_k(x)$ in the kth iteration of Algorithm 8, then we have $\rho(\hat{x}) = \rho^\diamond_k(\hat{x}) = \max(\rho^c_k(\hat{x}), \rho^u_k(\hat{x}))$. To show $\rho(\hat{x})$ is the minimum distance of $\hat{x}$, we only need to show $\rho^c_k(\hat{x})$ is not greater than $\min_{e \in E_c(\hat{x})}\{\lambda(x') + c(e) \mid e = (\hat{x}, \sigma, x')\}$ since $\rho^u_k(\hat{x})$ is fixed for any iteration of Algorithm 8. Note that $\rho^c_k(\hat{x})$ is not greater than $\min_{e \in E_{c_k}(\hat{x})}\{\lambda(x') + c(e) \mid e = (\hat{x}, \sigma, x')\}$ such that $x' \in \Omega_k$ due to the definition of $\rho^c_k(\hat{x})$. So we just need to show $\rho^c_k(\hat{x})$ is not greater than $\lambda(\bar{x}) + c((\hat{x}, \sigma, \bar{x}))$ for any state $\bar{x} \notin \Omega_k$.

It is clear that $\bar{x} \notin X_m$ because $X_m \subseteq \Omega_k$ for any k. It follows that $\lambda(\bar{x}) = \rho(\bar{x})$. Since $\bar{x} \notin \Omega_k$, so $\bar{x} \notin \Omega'_k$, then $\rho(\bar{x})$ is computed in some ith iteration of Algorithm 8 such that $i > k$ or $\rho(\bar{x}) = \infty$. Then it follows from Lemma 6 that $\rho(\bar{x}) > \rho(\hat{x})$, so $\rho^c_k(\hat{x}) \leq \rho(\hat{x}) < \rho(\bar{x}) = \lambda(\bar{x}) < \lambda(\bar{x}) + c((\hat{x}, \sigma, \bar{x}))$. ■

Next we present an algorithm that computes an optimal director by applying Algorithm 8 iteratively on different input plants starting from G. During the kth iteration, the input plant G_k is processed to derive the next input plant G_{k+1}, which is a sub-plant of G_k. This is conceptually illustrated by Figure 6.3. (Note that all the uncontrollable events in Figure 6.3 are disturbance inputs.)

Algorithm 9 Input a plant $G := (X, E, X_m)$, the following steps compute an optimal director D^*.

Initiation:

1. Set $k = 1$ and $G_k = G$.

2. Apply Algorithm 8 on G_k. We denote the resulting director and its distance function as D_k and ρ_k, respectively.

3. $D^* = D_k$.

Iteration:

1. $I_k := I(G_k^{D_k})$.

2. Let $V_k \subseteq I_k$ be the set of states that belong to the argument of $\max_{x \in I_k} \rho_k(x)$.

3. If $x_0 \in V_k$, then skip the remaining iteration steps.

4. $G_{k+1} := G_k \setminus V_k$, where $G_k \setminus V_k$ represents the operation to reduce G_k to a trim plant in which the states of V_k are removed. This can be accomplished by setting the cost of all incoming edges to V_k to be ∞ (so that the states of V_k are seen as

"illegal" states) and then by applying Algorithm 5 (presented in the Chapter 4) on the resulting modified G_k.

5. If $G_{k+1} = \emptyset$, then skip the remaining iteration steps.

6. Apply Algorithm 8 on G_{k+1}. We denote the resulting director and its distance function as D_{k+1} and ρ_{k+1}, respectively.

7. If $P(D^*) \geq P(D_{k+1})$, then set $D^* = D_{k+1}$.

Termination:

- If $x_0 \in V_k$ or $G_{k+1} = \emptyset$, then stop. If $P(D^*) = \infty$, then there exists no optimal director, otherwise the optimal director is D^*.

- If $x_0 \notin V_k$ and $G_{k+1} \neq \emptyset$, then continue the iteration with $k = k + 1$.

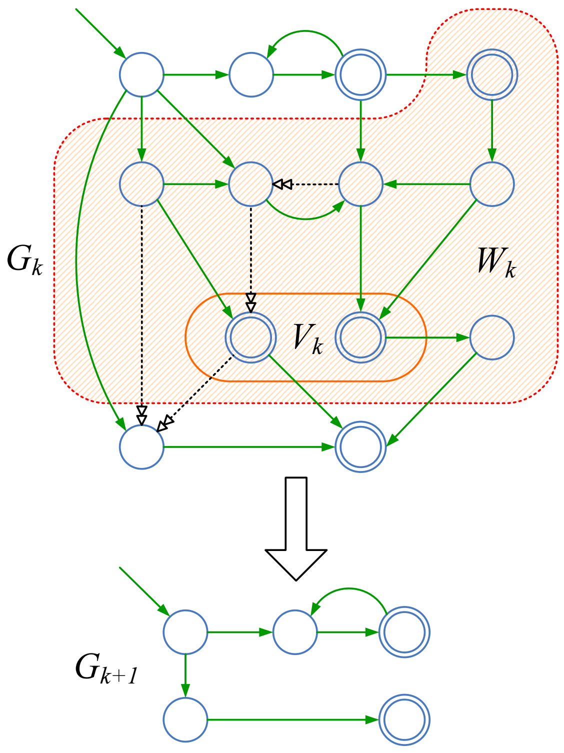


Figure 6.3 Illustration of input plants for Algorithm 9

Remark 19 Algorithm 9 iteratively applies Algorithm 8 on different input plants and compares the results. It also uses Algorithm 5 to derive the input plant for the next iteration. It is clear that there is a maximum $|X_m - X_t| + 1$ (resp. $|X_m - X_t|$) number of executions of Algorithm 8 (resp. Algorithm 5) inside Algorithm 9. Note that the complexity of Algorithm 8 and that of Algorithm 5 are both $O(|X| \times |\Sigma|)$. So the overall complexity of Algorithm 9 is $O(|X| \times |\Sigma| \times (|X_m - X_t| + 1))$, which is linear in the number of states, events and non-terminating mark states of G.

We present an example to aid the understanding of Algorithm 9.

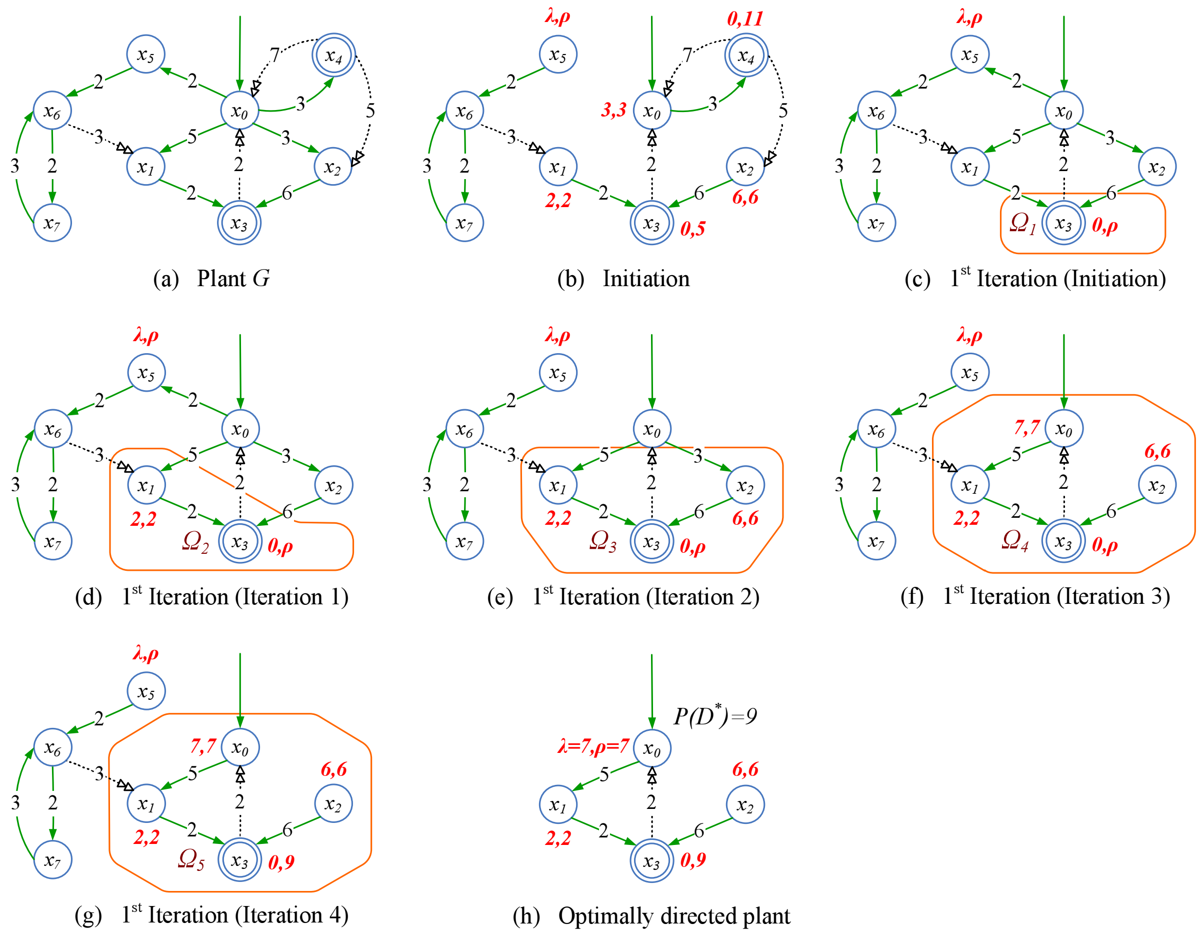

(a) Plant G
(b) Initiation
(c) 1st Iteration (Initiation)
(d) 1st Iteration (Iteration 1)
(e) 1st Iteration (Iteration 2)
(f) 1st Iteration (Iteration 3)
(g) 1st Iteration (Iteration 4)
(h) Optimally directed plant
λ,ρ
0,11
3,3
2,2
6,6
0,5
0,ρ
7,7
0,9
λ=7,ρ=7
P(D*)=9
Ω1
Ω2
Ω3
Ω4
Ω5

Figure 6.4 Example for Algorithm 9

Example 11 The plant G is the same as in Example 10 and repeats in Figure 6.4(a). In the initiation, Algorithm 9 applies Algorithm 8 on $G_1 = G$. The resulting plant is repeated in Figure 6.4(b) with cost of control being $P(D_1) = 11$. From Figure 6.4(b), we can see $\rho_1(x_4) > \rho_1(x_3) > \rho_1(x_0)$, so in the 1st iteration of Algorithm 9, $\{x_4\}$ is removed and the resulting (trim) plant is denoted as G_2 and shown in Figure 6.4(c). Next we apply Algorithm 8 on G_2, whose progress is illustrated in Figure 6.4(c)-6.4(g), with the text inside parentheses in the captions tracking the progress. Then we have G^{D_2} as shown in Figure 6.4(h) with cost of control being $P(D_2) = 9$.

Since $\rho_2(x_3) > \rho_2(x_0)$, Algorithm 9 then removes $\{x_3\}$. The resulting (trim) plant G_3 is empty, so Algorithm 9 terminates with optimal director as $D^* = D_2$.

Remark 20 We use $W_k := X(G_k) - X(G_{k+1})$ to represent the set of states removed (by Algorithm 5) during the kth iteration of Algorithm 9 (if $x_0 \notin V_k$), as illustrated by Figure 6.3. Note that W_k contains not only the states of V_k but also the states that have at least one path visiting V_k in *all* directed plants and the states that should be trimmed away if the above states are absent. Also note that both G_k and G_{k+1} are trim. So for *any* director D, it holds that $W_k \cap X^D \neq \emptyset$ if and only if $V_k \cap X^D \neq \emptyset$.

Next we prove the correctness of Algorithm 9. We need two lemmas, which assume the following notation: Let m be the last iteration ordinal of Algorithm 9, i.e., Algorithm 9 terminates after the mth iteration; let n be the *largest* iteration ordinal such that $P(D_n) = P(D^*)$, i.e., $n = \max_{0<k\leq m}\{k \mid P(D_k) = P(D^*)\}$.

The first lemma shows that the states removed before the nth iteration should *not* exist in some trim optimally directed plant.

Lemma 7 Given a plant G, consider the notation of Algorithm 9 and that for n stated above. If there exists an optimal director for G, then there exists an optimal director D for G such that $(X(G_1) - X(G_n)) \cap X^D = \emptyset$.

Proof: We prove by induction on the iteration ordinal k of Algorithm 9 for any $0 < k \leq n$. It is clear that the base step trivially holds, i.e., if there exists an optimal director for G, then there exists an optimal director D for G such that $(X(G_1) - X(G_1)) \cap X^D = \emptyset$.

For induction hypothesis, assume that there exists an optimal director D such that $(X(G_1) - X(G_k)) \cap X^D = \emptyset$ for any $0 < k < n$. We need to show there exists an optimal director D' such that $(X(G_1) - X(G_{k+1})) \cap X^{D'} = \emptyset$.

Assume for contradiction that there exists a state $x_k \in (X(G_1) - X(G_{k+1})) \cap X^D$ for *any* optimal D such that $((X(G_1) - X(G_k)) \cap X^D = \emptyset$ and $0 < k < n$. Then $x_k \in W_k \cap X^D$. So it follows (from Remark 20) that there exists a state $x'_k \in V_k \cap X^D$.

Note that x'_k belongs to the argument of $\max_{x \in I_k} \rho_k(x)$. Also note that $P(D_n) \leq P(D_k)$ holds for any $0 < k < n$, so $\rho_k(x'_k) = P(D_k) \geq P(D_n)$.

Since G_k is a sub-plant of $G_1 = G$ and $((X(G_1) - X(G_k)) \cap X^D = \emptyset$, we have $\forall x \in X^D : d_{G^D}(x) = d_{G_k^D}(x)$. Then it follows from Theorem 12 that $d_{G^D}(x'_k) = d_{G_k^D}(x'_k) \geq \rho_k(x'_k)$.

Therefore $P(D) \geq d_{G^D}(x'_k) \geq \rho_k(x'_k) = P(D_k) \geq P(D_n)$. Since D is optimal, $P(D) \leq P(D_n)$ also holds, so it follows that $P(D) = P(D_n)$, which implies D_n is optimal. Since $X(G_1) \supseteq X(G_k) \supseteq X(G_n)$ holds for any $0 < k < n$, we have $(X(G_1) - X(G_{k+1})) \cap X(G_n) = \emptyset$. It is clear that $X^{D_n} \subseteq X(G_n)$, then it follows that $(X(G_1) - X(G_{k+1})) \cap X^{D_n} = \emptyset$, which is a contradiction.

Hence if there exists an optimal director for G, then there exists an optimal director D for G such that $(X(G_1) - X(G_n)) \cap X^D = \emptyset$. ■

The next lemma shows that certain states removed during and after the nth iteration should exist in some trim optimally directed plant.

Lemma 8 Given a plant G, consider the notation of Algorithm 9 and those for m and n stated above. If there exists an optimal director D for G such that $(X(G_1) - X(G_n)) \cap X^D = \emptyset$ and $m \neq n$, then $W_n \cap X^D \neq \emptyset$.

Proof: We first prove by induction on the iteration ordinal k of Algorithm 9 that $(X(G_1) - X(G_k)) \cap X^D \neq \emptyset$ for $n < k \leq m$.

We show the base step holds, i.e., $(X(G_1) - X(G_m)) \cap X^D \neq \emptyset$. Assume for contradiction that $(X(G_1) - X(G_m)) \cap X^D = \emptyset$.

If $x_0 \in V_m$, then x_0 belongs to the argument of $\max_{x \in I_m} \rho_m(x)$. Also note that $P(D_n) < P(D_k)$ holds for any $n < k \leq m$. Then it follows from Theorem 12 that $P(D) \geq d_{G^D}(x_0) = d_{G_m^D}(x_0) \geq \rho_m(x_0) = P(D_m) > P(D_n)$, which is a contradiction to the optimality of D. If $x_0 \notin V_m$, then $G_{m+1} = \emptyset$ and we have $W_m = X(G_m) - X(G_{m+1}) = X(G_m)$. Since $x_0 \in X(G_m) \cap X^D$, we have $X(G_m) \cap X^D = W_m \cap X^D \neq \emptyset$. Then it follows (from Remark 20) that $V_m \cap X^D \neq \emptyset$. Pick any state $x_m \in V_m \cap X^D$, then x_m belongs to the argument of $\max_{x \in I_m} \rho_m(x)$. It follows that $P(D) \geq d_{G^D}(x_m) = d_{G_m^D}(x_m) \geq \rho_m(x_m) = P(D_m) > P(D_n)$, which is also a contradiction to the optimality of D. Therefore $(X(G_1) - X(G_m)) \cap X^D \neq \emptyset$.

For induction hypothesis, assume that $(X(G_1) - X(G_k)) \cap X^D \neq \emptyset$ for any $n + 1 < k \leq m$. We need to show $(X(G_1) - X(G_{k-1})) \cap X^D \neq \emptyset$.

Assume for contradiction that $(X(G_1) - X(G_{k-1})) \cap X^D = \emptyset$. So $\forall x \in X^D : d_{G^D}(x) = d_{G_{k-1}^D}(x)$. Since $(X(G_1) - X(G_k)) \cap X^D \neq \emptyset$, there exists a state $x_{k-1} \in (X(G_{k-1}) - X(G_k)) \cap X^D = W_{k-1} \cap X^D$. Then it follows (from Remark 20) that there exists a state $x'_{k-1} \in V_{k-1} \cap X^D$. Note that x'_{k-1} belongs to the argument of $\max_{x \in I_{k-1}} \rho_{k-1}(x)$. Then it follows from Theorem 12 that $P(D) \geq d_{G^D}(x'_{k-1}) = d_{G_{k-1}^D}(x'_{k-1}) \geq \rho_{k-1}(x'_{k-1}) = P(D_{k-1}) > P(D_n)$, which is a contradiction to the optimality of D. Therefore $(X(G_1) - X(G_{k-1})) \cap X^D \neq \emptyset$.

So we have proved that $(X(G_1) - X(G_k)) \cap X^D \neq \emptyset$ for any $n < k \leq m$. Note that $(X(G_1) - X(G_n)) \cap X^D = \emptyset$. So we have $(X(G_n) - X(G_{n+1})) \cap X^D \neq \emptyset$. Hence $W_n \cap X^D \neq \emptyset$. ■

With Lemma 7 and 8 in hand, the correctness of Algorithm 9 is established as follows.

Theorem 13 Given a plant G, consider the notation of Algorithm 9 and those for m and n stated above. If there exists an optimal director for G, then the director D^* computed by Algorithm 9 is optimal.

Proof: If $m = n$ and there exists an optimal director for G, then it follows from Lemma 7 that there exists an optimal director D for G such that $(X(G_1) - X(G_m)) \cap X^D = \emptyset$. So $\forall x \in X^D : d_{G^D}(x) = d_{G_m^D}(x)$.

If $x_0 \in V_m$, then x_0 belongs to the argument of $\max_{x \in I_m} \rho_m(x)$. Then it follows from Theorem 12 that $P(D) \geq d_{G^D}(x_0) = d_{G_m^D}(x_0) \geq \rho_m(x_0) = P(D_m) = P(D_n) = P(D^*)$. Since D is optimal, $P(D) \leq P(D^*)$ also holds, so it follows that $P(D) = P(D^*)$, which implies D^* is optimal. If $x_0 \notin V_m$, then $G_{m+1} = \emptyset$ and we have $W_m = X(G_m) - X(G_{m+1}) = X(G_m)$. Since $x_0 \in X(G_m) \cap X^D$, we have $X(G_m) \cap X^D = W_m \cap X^D \neq \emptyset$. Then it follows (from Remark 20) that $V_m \cap X^D \neq \emptyset$. Pick any state $x_m \in V_m \cap X^D$, then x_m belongs to the argument of $\max_{x \in I_m} \rho_m(x)$. It follows that $P(D) \geq d_{G^D}(x_m) = d_{G_m^D}(x_m) \geq \rho_m(x_m) = P(D_m) = P(D_n) = P(D^*)$. Since D is optimal, $P(D) \leq P(D^*)$ also holds, so it follows that $P(D) = P(D^*)$, which implies D^* is optimal.

If $m \neq n$ and there exists an optimal director for G, then it follows from Lemma 7 and 8 that there exists an optimal director D for G such that $(X(G_1) - X(G_n)) \cap X^D = \emptyset$ and $W_n \cap X^D \neq \emptyset$. So $\forall x \in X^D : d_{G^D}(x) = d_{G_n^D}(x)$. Also it follows (from Remark 20) that there exists a state $x_n \in V_n \cap X^D$. Note that x_n belongs to the argument of $\max_{x \in I_n} \rho_n(x)$. Then it follows from Theorem 12 that $P(D) \geq d_{G^D}(x_n) = d_{G_n^D}(x_n) \geq \rho_n(x_n) = P(D_n) = P(D^*)$. Again since D is optimal, $P(D) \leq P(D^*)$ also holds, so it follows that $P(D) = P(D^*)$, which implies D^* is optimal. ■

It is clear that Algorithm 9 can be used to check the existence of an optimal director. So the existence and the synthesis of an optimal director for general plants are both polynomially solvable.

6.2 Application example

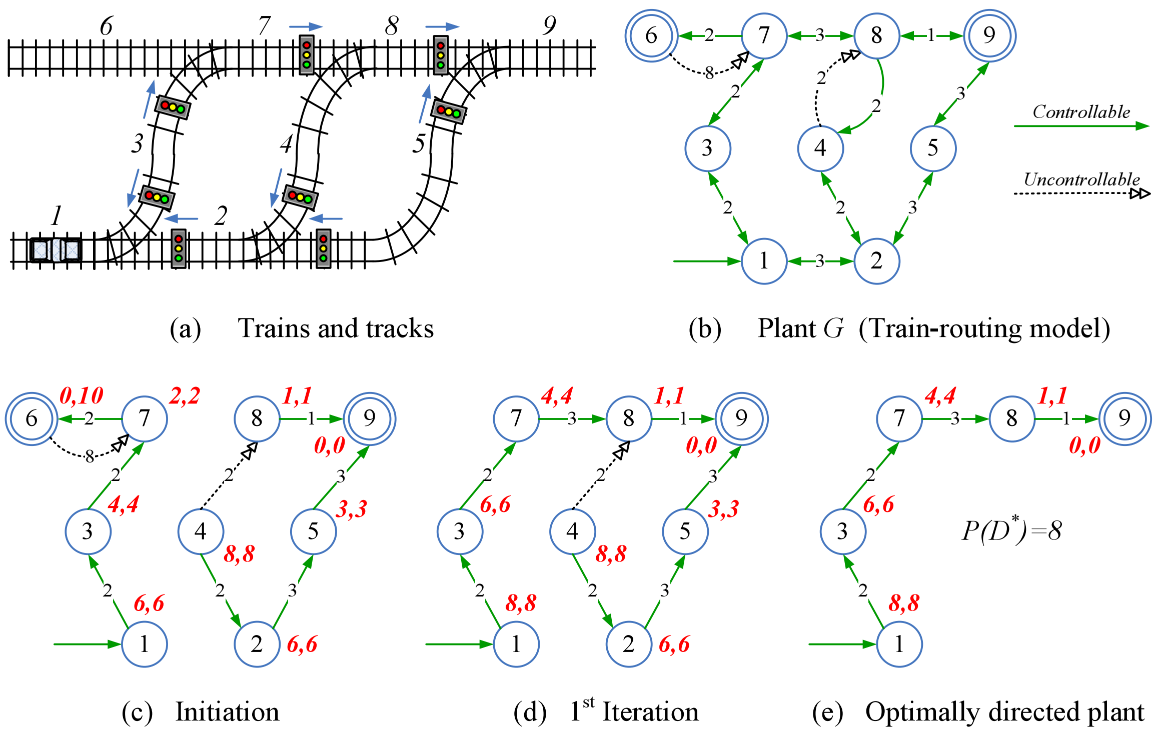


Figure 6.5 Synthesis of an optimal director for the application example

We provide an application example to demonstrate our result. The application is of train-traffic control over a set of track sections. As shown in Figure 6.5(a), nine sections of tracks labeled from 1 to 9 are separated by some traffic lights and switches. The traffic lights can be used to stop the traffic in the directions as indicated by the arrows above the lights, but have no effect for the traffic in the opposite directions. Suppose initially a train is in the section 1. We are required to synthesize an optimal director to control the traffic lights and switches to ensure the train eventually reach the section 6 or 9.

We model the movement of the train by a plant G, shown in the Figure 6.5(b), with the state "1" as the initial state, and the states "6" and "9" as marked states. The cost associated with each edge of G is the amount of time taken by the train to go from the source section to the destination section of the edge. Also note that the way traffic lights and switches are placed, the events corresponding to transitions from the section 4 to 8 and from the section 6 to 7 are (uncontrollable) disturbance inputs, whereas all other transitions are controllable.

We apply Algorithm 9 on G. After initiation, we obtain the directed plant with control cost of 10, shown in Figure 6.5(c). For each state x in the plant, we show the values $\lambda(x), \rho(x)$ as a pair of numbers beside the state x. We then execute the 1st iteration of Algorithm 9 and the algorithm terminates thereafter, which yields the optimally directed plant with the control cost of 8, shown in Figure 6.5(d). The trim optimally directed plant is shown in Figure 6.5(e).

The control strategy implemented by this optimal director is summarized as follows. We first guide the train from the section 1 to the section 7 via the section 3. Then we guide the train from the section 7 to the section 9 via the section 8.

CHAPTER 7. CONCLUSION

7.1 Summary

For the logic control of discrete event systems, prevailing work deals with the notion of supervisory control. A supervisory controller computes a maximal allowable set of controllable events at each state, but leaving undecided exactly which one is to be enabled. In this work, we have introduced the notion of directed control, which refines that of supervisory control.

A directed controller selects at most one controllable event to be enabled at any state while keeping all uncontrollable event enabled. This mechanism is in fact how a discrete event control is implemented and it becomes more relevant when the plant under control is an executor of controllable events rather than a generator of those. In addition, we divide the uncontrollable events into the disturbance inputs and the sensor outputs, and enrich the notion of non-terminating marked states to come up with a more meaningful and useful control strategy.

To understand how a plant under directed control works, we have modeled and analyzed the system using predominant automaton formalism. With modeling techniques at our disposal, we have designed control so as to achieve the desired controlled system behaviors. Our first goal is logical correctness as specified using safety and nonblocking. Subsequently we have addressed the best performance issue by providing an optimization based framework. The optimization task is to direct a system in such a way that regardless of the history of evolution, it accomplishes a pending task in a minimal cost.

Without loss in generality, we have formulated and studied the existence and synthesis problems in a state-based setting. We have showed that a safe and nonblocking directed controller exists if and only if a safe and nonblocking supervisory controller exists. The complexity of existence and synthesis of a safe and nonblocking directed controller is then shown to be polynomial in the size of plant, and is the same as that of a safe and nonblocking supervisory controller.

Next, we have obtained a test for verifying the existence of an optimal directed controller. An algorithm of polynomial complexity has been presented for solving the optimal directed control problem for systems that are cycle-free. The solution is proved to be dynamic-programming optimal.

Finally we have developed a novel approach with polynomial complexity for the synthesis (and existence) of an optimal directed controller for general plants. This is accomplished in two steps, the first of which finds a “locally optimal” directed controller using a “greedy” search, and the second step iteratively refines the search space for the greedy search. The final result provides a complete solution to all the existence and synthesis problems in discussion.

7.2 Future research

7.2.1 Directed control under partial observation

7.2.1.1 Notion of directed control under partial observation

It is assumed in the previous work that a director is capable of observing the occurrence of all events that plants execute. In many situations, it is difficult, if not impossible, for a director to observe all events due to limitations of the sensors or the distributed nature of some systems where events at some locations are not seen at other locations [15][8].

Thus, a particular event may be completely unobservable to a director, or two different events may be indistinguishable to a director. The presence of such partial observation can be captured by defining an observation *projection* (or *mask*) from the set of events to the set of "observed events" $P : \Sigma \rightarrow \Delta \cup \{\epsilon\}$. If an event is mapped to $\{\epsilon\}$, then the corresponding event is unobservable to a director. Similarly, if two events are mapped to the same observed event, then the corresponding events are indistinguishable to a director [21][15][8].

The special case in which a projection P simply erases some of the events in Σ occurs frequently, for example, due to absence of sensors. This is called *natural projection* [21][15][8]. In this case, the event set Σ is partitioned into two disjoint subsets:

$$\Sigma = \Sigma_o \cup \Sigma_{uo}$$

Σ_o is the set of observable events while Σ_{uo} is the set of unobservable events. Thus, a natural projection $P : \Sigma^* \rightarrow \Sigma_o^*$ is defined as

$$P(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_o \\ \epsilon & \text{otherwise} \end{cases}$$

The control action of a projection P is extended to traces by

$$\forall s \in \Sigma^*, \sigma \in \Sigma : P(s\sigma) = P(s)P(\sigma)$$

Since a director takes its control actions based on the observed sequence of events, it must take identical control action following all traces that have identical projections. In order to capture this fact, we define a partial-observation director, called P-director, as a map $D_P : P[L(G)] \rightarrow 2^{\Sigma_c}$ such that

$$\forall s \in L(G) : |D_P[P(s)]| \leq 1 \text{ and } \forall s \in L_e(G) : |D_P[P(s)]| = 1.$$

This means that control action can change only after the occurrence of an observable event, i.e., when $P(s)$ changes.

The languages generated and marked by the directed plant under partial observation are denoted by $L(G^{D_P})$ and $L_m(G^{D_P})$ respectively, which are defined as follows:

$\epsilon \in L(G^{D_P})$;

$[s \in L(G^{D_P}), \sigma \in D_P[P(s)] \cup \Sigma_u, s\sigma \in L(G)] \Leftrightarrow [s\sigma \in L(G^{D_P})]$;

$L_m(G^{D_P}) := L(G^{D_P}) \cap L_m(G)$.

7.2.1.2 Nonblocking directed control under partial observation

Under partial observation, it is expected that the existence and synthesis of directors will be different from those under complete observation, as the case for nonblocking supervisor.

Given a nonempty specification language $K \subseteq L_m(G)$, it is known [21][15][8] that there exists a nonblocking supervisor S under partial observation if and only if K is controllable, i.e., $pr(K)\Sigma_u \cap L(G) \subseteq pr(K)$, relative-closed, i.e., $pr(K) \cap L_m(G) = K$, and observable, i.e.,

$$\forall s \in pr(K), \sigma \in \Sigma : (s\sigma) \notin pr(K) \text{ and } (s\sigma) \in L(G) \Rightarrow P^{-1}[P(s)]\{\sigma\} \cap pr(K) = \emptyset.$$

It is expected that under the same conditions, there exist a nonblocking director under partial observation.

It is well known, however, that most discrete decision and control problems with partial information are computationally difficult [22], which is also the case for nonblocking supervisory control under partial observation [26]. The complexities for the existence and synthesis of nonblocking directors under partial observation are subjects for future research.

7.2.1.3 Optimal directed control under partial observation

Similarly, we plan to investigate the solvabilities and computational complexities for the existence and synthesis of optimal directors under partial observation. Further, depending on the application, different notions of optimality may be defined and new algorithms for obtaining new types of optimal director may be developed.

7.2.2 Directed control of timed systems

The above discussion for directed control do not admit an explicit modeling of time. For analysis of time-dependent systems, timed automata [1] and other timed DES models [5][18][4] were introduced. For example, the definition of timed automata provides a simple way to annotate state-transition graphs with timing constraints using finitely many real-valued clock variables.

Under the framework of directed control, we plan to investigate the properties of timed discrete event systems. Reachability, nonblocking and optimality under complete or partial observation are of interest for future research.

APPENDIX

Algorithm for computing $\mathcal{SLIN}$

Algorithm 10 Given a component $(\hat{X}, \hat{\alpha})$, the following steps compute $\mathcal{SLIN}(\hat{X}, \hat{\alpha})$.

1. $(X_0, \alpha_0) = (\hat{X} - X_i, \hat{\alpha}|_{\hat{X}-X_i})$; $k = 0$.
2. $X_{k+1} = X_k - V(X_k, \alpha_k)$; $\alpha_{k+1} = \alpha_k|_{X_{k+1}}$.
3. Repeat Step 2 with $k = k + 1$ until $X_{k+1} = X_k$.
4. Compute $\mathcal{S}(X_k, \alpha_k)$.
5. For each sub-component $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{S}(X_k, \alpha_k)$, if it is
   (a) invariant and nonblocking, then $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{SLIN}(\hat{X}, \hat{\alpha})$;
   (b) variant and nonblocking, then go to Step 1 with $(X_0, \alpha_0) = (\widetilde{X}, \widetilde{\alpha})$;
   (c) blocking, then $(\widetilde{X}, \widetilde{\alpha}) \notin \mathcal{SLIN}(\hat{X}, \hat{\alpha})$.

Algorithm for computing $\mathcal{SLA}$

Algorithm 11 Given a reference state set $X_r \subseteq X$ and a component $(\hat{X}, \hat{\alpha})$, the following steps compute $\mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$.

1. $(X_0, \alpha_0) = (\hat{X} - X_i, \hat{\alpha}|_{\hat{X}-X_i})$; $k = 0$.

2. $X_{k+1} = X_k - U((X_k, \alpha_k), X_r)$; $\alpha_{k+1} = \alpha_k|_{X_{k+1}}$.

3. Repeat Step 2 with $k = k + 1$ until $X_{k+1} = X_k$.

4. Compute $\mathcal{S}(X_k, \alpha_k)$.

5. For each sub-component $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{S}(X_k, \alpha_k)$, if it is singularly X_r-attractable, then $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$; otherwise go to Step 1 with $(X_0, \alpha_0) = (\widetilde{X}, \widetilde{\alpha})$.

BIBLIOGRAPHY

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

[2] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, July 1993.

[3] M. Barbeau, M. Frappier, F. Kabanza, and R. St.-Denis. A supervisory control synthesis case study: The antenna control system. In *Proceedings of the 35th Annual Allerton Conference on Communication, Control, and Computing*, pages 533–542, Urbana-Champaign, IL, 1997.

[4] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345, 2004.

[5] B. A. Brandin and W. M. Wonham. Supervisory control of timed discrete event systems. *IEEE Transactions on Automatic Control*, 39(2):329–342, February 1994.

[6] Y. Brave and M. Heymann. On stabilization of discrete event processes. *International Journal of Control*, 51(5):1101–1117, 1990.

[7] Y. Brave and M. Heymann. On optimal attraction in discrete-event processes. *Information Sciences: an International Journal*, 67(3):245–276, 1993.

[8] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, New York, NY, second edition, 2007.

[9] V. Chandra, Z. Huang, and R. Kumar. Automated control synthesis for an assembly line using discrete event system control theory. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 33(2):284–289, 2003.

[10] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1–2):35–84, 2003.

[11] M. Fabian and A. Hellgren. PLC-based implementation of supervisory control for discrete event systems. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 3, pages 3305–3310, Tampa, FL, December 1998.

[12] F. Giunchiglia and P. Traverso. Planning as model checking. In *Recent Advances in AI Planning*, volume 1809 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2000.

[13] C. H. Golaszewski and P. J. Ramadge. Supervisory control of discrete event processes with arbitrary controls. In P. Varaiya and A. B. Kurzhanski, editors, *Discrete Event Systems: Models and Applications*, pages 459–469. Springer-Verlag, 1987.

[14] M. Heymann. Concurrency and discrete event control. *IEEE Control Systems Magazine*, 10(4):103–112, June 1990.

[15] R. Kumar and V. K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, MA, 1995.

[16] R. Kumar and V. K. Garg. Optimal supervisory control of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 33(2):419–439, 1995.

[17] R. Kumar, S. Takai, M. Fabian, and T. Ushio. Maximally permissive mutually & globally nonblocking supervision with application to switching control. *Automatica*, 41(8):1299–1312, August 2005.

[18] F. Lin and W. M. Wonham. Supervisory control of timed discrete event systems under partial observation. *IEEE Transactions on Automatic Control*, 40(3):558–562, 1995.

[19] H. Marchand, O. Boivineau, and S. Lafortune. Optimal control of discrete event systems under partial observation. In *Proceedings of IEEE Conference on Decision and Control*, pages 2335–2340, Orlando, FL, 2001.

[20] K. M. Passino and P. J. Antsaklis. On the optimal control of discrete event systems. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 2713–2718, Tampa, FL, December 1989.

[21] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[22] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, January 1989.

[23] R. Sengupta and S. Lafortune. A graph-theoretic optimal control problem for terminating discrete event processes. *Discrete Event Dynamic Systems: Theory and Applications*, 2(2):139–172, 1992.

[24] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM Journal of Control and Optimization*, 36(2):488–541, 1998.

[25] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[26] J. N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control, Signals, and Systems*, 2(2):95–107, June 1989.