A hierarchical control structure for a class of timed discrete event systems

Danjing Li
Magdeburg

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktorin der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Manfred Opper
Berichter:	Prof. Dr.-Ing. Jörg Raisch
Berichter:	Prof. Dr.-Ing. Thomas Moor

Tag der wissenschaftlichen Aussprache: 28. November 2008

Berlin 2008
D 83

Contents

Zusammenfassung in deutscher Sprache

Ein ereignisdiskretes System ("discrete event system", DES) ist ein dynamisches System, dessen Verhalten durch eine Ereignisreihenfolge beschrieben werden kann. Ereignisse entsprechen dem Beginn oder Ende von Aktivitäten. Die Studie solcher Systeme hat in den letzten Jahren immer mehr an Bedeutung gewonnen. In der heutigen, von digitalen Systemen geprägten Welt finden sich viele Beispiele für ereignisdiskrete Systeme, beispielsweise in Bezug auf Analyse, Optimierung und Steuerung von flexiblen Fertigungssystemen, Verkehrssystemen, Datenbankverwaltungssystemen, Kommunikationsnetzen, Logistiksystemen, chemischen Prozessen und weiteren Anwendungen.

Viele ereignisdiskrete Systeme haben eine vorbestimmte Reihenfolge der Aktivitäten. In der Realität treten häufig Störungen auf, so dass die vorbestimmte Reihenfolge ihre Optimalität verliert. Die derzeitigen Herausforderungen bei der Steuerung und Regelung komplexer ereignisdiskreter Systeme umfassen die Koordination der Teilsysteme ("subplants", z. B. Züge in einem Eisenbahn-Netz, Produkte in flexiblen Fertigungssystemen) und die sachgemäße Behandlung von Störungen wie beispielsweise Blockierung oder mechanischer Ausfall. Unter solchen Umständen ist es wichtig, dass sich die Koordination und die Implementierungsstrategie dem dynamischen Umfeld anpassen. Angeregt durch die Einfachheit und die Wirksamkeit der Methode der Max-Plus-Algebra für ereignisdiskrete Systeme, wird in dieser Arbeit ein neues hierarchisches Steuerungs- und Regelungsverfahren für ereignisdiskrete Systeme mit einem generischen Ansatz vorgeschlagen.

Für die Regelung von Max-Plus-Systemen sind hauptsächlich zwei Ansätze von Bedeutung: Der erste Ansatz ist die Methode "model predictive control" (z. B. de Schutter und van den Boom [87]). Der Hauptvorteil dieses Ansatzes ist die Möglichkeit, eine breite Klasse von Zielfunktionen vorgeben zu können. Allerdings ist der erforderliche Rechenaufwand sehr hoch. Der zweite Ansatz ist die optimale Regelung auf Basis der "residua-

tion theory" (z. B. Cottenceau et al. [24, 25]). Dieser Ansatz findet sich in der Literatur auch unter der Bezeichnung "Just-in-Time control" (z. B. Menguy et al. [71, 72]). Mit der besonderen "Just-in-Time"-Anforderung als Zielfunktion kann die optimale Steuerung durch einfache algebraische Berechnungen ermittelt werden. Allerdings ist "Just-in-Time" für viele Anwendungen als Zielsetzung nicht zweckmäßig. In dieser Arbeit wird eine alternative, *energieoptimale* Zielfunktion für die Regelung spezifiziert. Im Kontext der in dieser Arbeit betrachteten Anwendung ist dieser Ansatz günstiger. Er erlaubt es dennoch, die Vorteile der Max-Plus-Algebra (kompakte Darstellung, schnelle Berechnung) zu nutzen.

Die meisten in der Literatur betrachteten deterministischen Max-Plus-Algebra-Systeme sind *zeitinvariant*. Üblicherweise treten dabei nur Systemmatrizen mit positiver Ordnung auf. Dies bedeutet, dass der Zeitpunkt eines Ereignisses nur von den Ereignissen desselben Zyklus oder von Ereignissen früherer Zyklen abhängen kann. Es gibt jedoch auch Anwendungen, bei denen Ereigniszeitpunkte von Ereignissen *späterer* Zyklen abhängen. Diese Situation entspricht Systemmatrizen *negativer Ordnung* im Max-Plus-Algebra-Modell. Bisher gibt es keine Studien zur optimalen Steuerung zeitvarianter Systeme mit Systemmatrizen negativer Ordnung. Dieses Thema wird deshalb hier diskutiert.

In dieser Arbeit wird eine neue hierarchische Reglerarchitektur für eine Klasse ereignisdiskreter Systeme ohne und mit zyklischen Eigenschaften vorgeschlagen. Die obere hierarchische Ebene basiert auf einem Max-Plus-Algebra-Modell und aktualisiert anhand von Information über den Systemzustand in Echtzeit die optimale Reihenfolge der Aktivitäten. Die aktualisierte Reihenfolgeinformation wird in der unteren Ebene in ein Referenzsignal übersetzt, anhand dessen das physikalische System geregelt werden kann. Durch die Ausnutzung der auf der unteren Ebene verbleibenden Freiheitsgrade wird der Gesamtenergieverbrauch verringert. Die eingeführte Methode nutzt die zur Max-Plus-Algebra duale Min-Plus-Algebra. Ein neuer, intuitiver Algorithmus beschleunigt die Berechnung und erzeugt die global optimalen Referenzsignale. Diese Arbeit konzentriert sich auf die Anwendung auf Schienenverkehrssysteme. Der vorgeschlagene Regelungsentwurf kann jedoch ebenso auf andere ereignisdiskrete Systeme, wie zum Beispiel "High-Throughput-Systeme", flexible Fertigungssysteme, oder chemische Batchprozesse angewendet werden.

Nach einer Einführung in Kapitel 1 beschreibt Kapitel 2 die vorgeschlagene Reglerstruktur im Allgemeinen und zeigt auf, wie sie für *nicht-zyklische* DES (d. h. für ereignisdiskrete Systeme mit nur einem Zyklus) bei Schienenverkehrssystemen funktioniert. Kapitel 3 erweitert die in Kapitel 2 eingeführte hierarchische Reglerarchitektur auf eine Klasse

ereignisdiskreter Systeme mit *zyklischer* Wiederholung. Kapitel 4 befasst sich mit dem Regelungsproblem für zyklische ereignisdiskrete Systeme mit weniger einschränkenden Bedingungen als im vorhergehenden Kapitel: Ereignisse in ein späteren Zyklus dürfen vor Ereignissen in den früheren Zyklen stattfinden. Diese Eigenschaft wird im Max-Plus-Modell durch Abhängigkeiten (Steuerkanten) "negativer Ordnung" repräsentiert. Zusätzlich wird in Kapitel 5 die vorgeschlagene Reglerarchitektur auf ein modifiziertes Schedulingproblem aus dem Bereich des "High-Throughput-Screening" [69] angewandt.

Chapter 1

Introduction

1.1 Discrete event systems

A Discrete Event System (DES, for more information on DES, see e. g. [12]) is a dynamic system whose behaviour can be described by a sequence of events. Events correspond to starting or ending of some activities. For example, an event may represent a batch entering a resource, a product leaving a machine, or a disturbance occurring. The study of such systems has become increasingly important and popular in recent years. Many Examples of DES can be found in today's digital world, including the industries related to analysis, optimisation and control: flexible manufacturing systems, traffic systems, database management systems, communication protocols, logistic service systems, and chemical processes as well.

Many discrete event systems have predefined orders of activities. The activities either repeat themselves periodically or operate without any regular pattern. As a result, there are two corresponding DES operation styles in general. The former operation style benefits from the simplicity of the structure and the steady progress. Therefore it is widely used in many application fields including manufacturing systems, chemical batch plants and transportation systems. On the other hand, the latter includes more degrees of freedom and may therefore, at least in principle, make better use of resources. In particular, Max-plus algebra is a powerful modelling and analysis tool for discrete event systems in which the order of competing activities on shared resources have been predetermined.

In reality, disturbances often happen in a way that the predetermined order loses its superiority. Based on online information, the control system should then update the optimal

order in real time. The updated order information is passed to the lower level where it is translated into a reference signal for the physical system to be controlled.

Fundamental difficulties in designing optimal control structures for both cyclic and non-cyclic DESs motivate us to look for new research avenues in max-plus-based DES optimisation techniques. In this thesis, we particularly focus on the application of railway transportation systems. It should be noted that the proposed control scheme can also be applied to other discrete event systems such as high throughput systems, flexible manufacturing systems and chemical batch processing.

To effectively extend max-plus algebra based DES control into more generic settings, it is beneficial to first review the general concept of max-plus algebra and its current role in the area of discrete event system control.

1.2 Literature review

Max-plus algebra is a convenient tool to model and analyse timed DES. For technical details, the reader is referred to the corresponding text books [3, 26, 50].

The initial motivation of max-plus algebra arises in modelling and analysis of the time behaviour of DESs. The DES timing often involves maximum and addition operations. The equations describing the behaviour of the system are nonlinear in the conventional algebra, but are linear in the sense of max-plus algebra. As Gaubert and the Max Plus working group point out in [39], max-plus algebra originated in the late fifties or even earlier, and developed rapidly in the sixties, with contributions by Cuninghame-Green, Vorobyev, Romanovskiĭ, and more generally within the Operations Research community on path algebra. Contributions to max-plus algebra can also be found under the headings as "path-algebra", "minimax algebra" and "dioid"[20]. For years, a large number of studies on max-plus algebra have been published.

The most studied and most important class of problems relates to the eigenstructure in the sense of max-plus algebra. For example, the *(max,+) Perron-Frobenius Theorem*, dealing with the eigenvalue determination for an irreducible matrix was proved by, for instance, Romanovskiĭ [83], Cuninghame-Green [26], Zimmermann [105], Gondran and Minous [43], Baccelli et al. [2], Bapat et al. [7, 8] and Akian et al. [1]. The common way to compute the eigenvalue relies on *Karp's Algorithm* [56]. Many algorithms based on Karp's algorithm have been proposed, for example, by Megiddo [70], by Karp and Orlin

[57], by Young et al. [103], by Hartmann and Orlin [48], by Ito and Parhi [54], and by Dasdan and Gupta [29]. The max-plus version of the *Howard Algorithm*, which is well known in stochastic control (see e. g. [31]), leads to a very different eigenvalue computation method, see Cochet-Terrasson et al. [17]. The *Power Algorithm*, proposed by Braker and Olsder [11], Elsner and van den Driesche [33], and Subiono and van der Woude [92], computes the eigenvalue through computing the max power of the matrix. Furthermore, based on Karp's formula, Elsner and van den Driessche [34] modify the Power Method. Cuninghame-Green and Yixun [28] and Olsder et al. [78] transform the graph theoretic approach of the Karp algorithm into a linear program to calculate eigenvalue and eigenvectors. Besides the above algorithms, Van der Woude [97] provides an approach from a different point of view. It resembles the well known simplex method in linear programming. As an extension, Lahaye et al. [60] analyse the spectral problem of max plus systems with periodically varying coefficients. Van der Woude and Olsder give a complete proof of the general formulation of the spectral theorem in [98]. For the eigenvalue problem of a reducible matrix, related results can be found in Wende et al. [101], Bapat et al. [7, 8]. Finally, Dasdan and Gupta [29] compare the efficiency of Karp's algorithm and some other algorithms; Soto y Koelemeijer [102] studies the relationships between the Power Algorithm and the Howard Algorithm.

Another important aspect is the residuation theory in the max-plus algebra context. Discussions from the system theoretic point of view include the work by the Max Plus working group [79], Cohen et al. [21], Gaubert et al. [39]. Short reviews of the interesting problems within max-plus algebra can be found in [22, 27, 39, 81]. Olsder and colleagues [75, 77] provide certain stochastic extensions in the context of max-plus algebra. Work in this area has also been reported by Resing et al. [82], by Baccelli [2, 6], by Mairesse [66], and by Heidergott et al. [51].

In a number of discrete event systems arising in operations research, control theory, computer science, etc., the state evolution involves not only the maximum operation and addition, but also the minimum operation. The concept of min-max functions is introduced in [20, 26, 47, 76] to describe the dynamic behaviour of such systems. Furthermore, Gunawardena [46], Gaubert and Gunawardena [38], Cochet-Terrasson et al. [18, 19], van der Woude [96], van der Woude and Subiono [99], Gavalec [40] analyse the dynamic behaviour of min-max-plus systems indicated by systems' eigenvalue or cycle time. An algorithm which is similar to the Howard Algorithm in [17] is proposed by Cheng and Zheng [15]. The Power Algorithm for min-max-plus systems is proposed in Suiono and van der Woude [92]. Jean-Marie and Olsder [55] extend the concept of min-max-plus sys-

tems to stochastic min-max-plus systems. Recently, Cheng and Zheng [16] have provide an algorithm to solve min-max inequalities which represent timing verification problems.

Application areas of max-plus algebra include computer communication systems [5], telecommunication networks, parallel processors [13], robotic systems [4, 58, 100], manufacturing systems (Chen et al. [14] apply max-plus algebra to a hot-metal rolling serial production line, other applications include Di Febbraro et al. [35, 36], T.-E. Lee [61], J.-W. Seo and T.-E. Lee [90], Zhu et al. [104], Elmahi et al. [32]), chemical batch plants [84] and transportation systems, see e. g. Car/Bus traffic [64, 73], train traffic [30, 62, 63, 68, 89]. Goverde and Odijk [45] propose an analytical tool called PETEP (Performance Evaluation of Timed Events in Railways) to assess rail network performance indicators in a deterministic setting.

Cohen et al. [20] discuss the relation between systems theory and max algebra. As in many applications, a lot of work has been focused on performance analysis and, in particular, the performance of periodic systems. Consequently, the notions of eigenvalue and eigenvectors in max-plus algebra sense play an important role. As Gaubert et al. claim in [39]: "one might argue that 90% of current applications of (max,+) algebra are based on a complete understanding of the spectral problem." The concepts of stability, controllability, reachability and observability have been discussed for example by Cohen et al. [23], by Spacek et al. [91], by Gazarik and Kamen [41]. Starting from the late 1990s, more attention was paid to control problems by Prou and Wagneur [80], Cottenceau et al. [25], and, in the context of transportation networks (which will also be the main focus of application in this thesis), for example, by de Vries et al. [30], Heidergott and de Vries [49], de Schutter et al. [89].

There appear to be two main approaches for solving control problems for max plus systems. The first approach is model predictive control, studied in de Schutter and van den Boom [87], van den Boom et al. [95], van den Boom and de Schutter [93, 94]. This control approach has also been extended to min-max-plus systems [85, 86, 88]. The main advantage of this approach is that it allows to specify a wide class of performance objectives. However, the required computational load is heavy. The second approach is residuation theory based optimal control as discussed in Cottenceau et al. [24, 25], Lahaye et al. [59]. This approach also appears under the names of "Just-in-Time control"(see e. g. Menguy et al. [71, 72], Maia et al. [65]), "Internal model control" (see e. g. Boimond and Ferrier [10]) and "Greatest subsolution method" (see Masuda et al. [67], Goto et al. [44]). With the specific "Just-in-Time" performance requirement as the control objective, the optimal control can be derived from simple algebraic calculations. Apparently, "Just-in-Time" is

not always the appropriate performance target for many applications. In this thesis, an alternative, *Energy Optimal Control* requirement will be specified as an objective. This makes more sense in the context of the application studied in this thesis while still taking advantage of the simplicity of max plus algebraic calculations.

The most studied deterministic max-plus algebra systems are *time invariant*. *Time varying* systems, i. e. systems whose parameters may change as functions of time while the system structure remains unchanged have been investigated in e. g. Lahaye et al. [59, 60], Masuda et al. [67] and van den Boom et al. [94]. A common feature for most investigated cyclic systems is that only system matrices of *Non-negative Order* appear. This implies that the timing of events only depends on events in the same or previous cycles (see Section 3.2). Nevertheless, in many applications, certain events in earlier cycles are desired to depend on events in later cycles, which corresponds to *Negative Order* system matrices. To the best of our knowledge, the only study dealing with negative order system matrices for time invariant systems is by Geyer [42]. Up to now, there are no studies of time varying systems involving negative order system matrices and their optimal control. This topic will therefore be discussed in this thesis.

1.3 Max-plus algebra

The basic mathematical theory of max-plus algebra used in this thesis is briefly presented in this section. For further details on max-plus algebra and its properties, the reader is referred to the textbooks [3, 26].

Definition 1 (Max-plus algebra) *Max-plus algebra [3, 26] is the set* $\mathbb{R}_{max} = \mathbb{R} \cup \{-\infty\}$, *endowed with two operations:* addition $\oplus$ *and* multiplication $\otimes$. *Addition* $\oplus$ *is defined to be the maximum of two elements in conventional algebra, while multiplication* $\otimes$ *is defined to be conventional addition, i. e.* $\forall a, b \in \mathbb{R}_{max}$ *:*

$$a \oplus b := \max(a, b), \tag{1.1}$$

$$a \otimes b := a + b. \tag{1.2}$$

As in conventional algebra, the multiplication symbol is sometimes omitted and $a \otimes b$ is written as ab. The additive identity (neutral element of addition) in max-plus algebra is $-\infty$, also denoted as ϵ. The multiplicative identity (neutral element of multiplication) in

max-plus algebra is 0, also denoted as e:

$$\forall a \in \mathbb{R}_{max}, a \oplus \epsilon = \epsilon \oplus a = a.$$

$$\forall a \in \mathbb{R}_{max}, a \otimes e = e \otimes a = a.$$

Similar to conventional algebra, some important properties also hold in max-plus algebra:
Associativity of addition:

$$\forall a, b, c \in \mathbb{R}_{max}, (a \oplus b) \oplus c = a \oplus (b \oplus c).$$

Commutativity of addition:

$$\forall a, b \in \mathbb{R}_{max}, a \oplus b = b \oplus a.$$

Associativity of multiplication:

$$\forall a, b, c \in \mathbb{R}_{max}, (a \otimes b) \otimes c = a \otimes (b \otimes c).$$

Distributivity of multiplication over addition:

$$\forall a, b, c \in \mathbb{R}_{max}, (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c),$$

$$\forall a, b, c \in \mathbb{R}_{max}, c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b).$$

Matrix addition and multiplication in max-plus algebra are defined respectively as:

$$C_{ij} = (A \oplus B)_{ij} = A_{ij} \oplus B_{ij}, A, B, C \in \mathbb{R}_{max}^{p \times q},$$

$$C_{ij} = (A \otimes B)_{ij} = \bigoplus_{k=1}^{m} A_{ik} \otimes B_{kj} = \max_{k}(A_{ik} + B_{kj}), A \in \mathbb{R}_{max}^{l \times m}, B \in \mathbb{R}_{max}^{m \times n}, C \in \mathbb{R}_{max}^{l \times n}.$$

The power of a square matrix A is:

$$A^k = \underbrace{A \otimes A \otimes \cdots \otimes A}_{k \text{ times}}.$$

The *identity matrix* in max-plus algebra has the form of:

$$I = \begin{bmatrix} e & & \epsilon \\ & \ddots & \\ \epsilon & & e \end{bmatrix}.$$

The *null matrix* in max-plus algebra is N, $N_{ij} = -\infty$.

A very typical and important max-plus algebra problem is the spectral problem:

$$A \otimes v_e = \lambda \otimes v_e.$$

Similar to conventional algebra, λ and v_e are called *eigenvalue* and eigenvector of the square matrix $A \in \mathbb{R}_{max}^{n\times n}$, respectively.

In the following, we need the notion of an irreducible matrix. A *directed graph* G [3] can be defined as a pair (V, E), where V is a set of elements called *vertices* and where E is a set the elements of which are ordered pairs of vertices, called *edges*. According to graph theory, for any matrix $A \in \mathbb{R}_{max}^{n\times n}$, there is a corresponding graph $G(A)$ whose *weighted incidence matrix* is A: the weight of the edge from vertex j to vertex i takes the numerical value of A_{ij}. If $A_{ij} = \epsilon$, the corresponding edge does not exist.

Definition 2 (Irreducible matrix) *[101] Let $G(A)$ be a graph whose weighted incidence matrix is A. If $G(A)$ is strongly connected, i. e. from any node to any other node there is a path, then A is called irreducible. If $G(A)$ is not strongly connected, A is called reducible.*

Theorem 1 ((max,+) Perron-Frobenius Theorem) *[39] An irreducible matrix $A \in \mathbb{R}_{max}^{n\times n}$ has a unique eigenvalue,*

$$\lambda = \bigoplus_{j=1}^{n} (tr(A^j))^{\frac{1}{j}},$$

where $tr(A^j)$ is the trace $\oplus_{i=1}^{n}(A^j)_{ii}$. In conventional algebra, this can be written as

$$\lambda = \max_{1\le j\le n} \max_{i_1,\cdots,i_j} \frac{A_{i_1 i_2} + \cdots + A_{i_j i_1}}{j}$$

and represents the maximal cycle mean of A.

Definition 3 (Min-plus algebra) *Min-plus algebra is the set $\mathbb{R}_{min} = \mathbb{R} \cup \{+\infty\}$, endowed with two operations:* addition $\oplus'$ *and* multiplication $\otimes'$. *Addition $\oplus'$ is defined to be the minimum of two elements in conventional algebra, while multiplication $\otimes'$ is defined to be conventional addition, i. e. $\forall a, b \in \mathbb{R}_{min}$:*

$$a \oplus' b := \min(a, b), \tag{1.3}$$

$$a \otimes' b := a + b. \tag{1.4}$$

For matrices $A \in \mathbb{R}_{max}^{l\times m}$, $B \in \mathbb{R}_{max}^{m\times n}$, multiplication is defined by

$$(A \otimes' B)_{ij} = \bigoplus_{k=1}^{m}{}' A_{ik} \otimes' B_{kj} = \min_k(A_{ik} + B_{kj}). \tag{1.5}$$

The min-max-plus algebra consists of the set $\mathbb{R}_{mm} = \mathbb{R} \cup \{+\infty, -\infty\}$, endowed with operations $\oplus$, $\otimes$, $\oplus'$, $\otimes'$ and the additional rules:

$$(-\infty) \otimes (+\infty) = -\infty, (-\infty) \otimes' (+\infty) = +\infty. \tag{1.6}$$

The following properties and inequalities hold if the involved matrices and vectors have consistent dimensions ("$-$" is the conventional minus):

$$\text{If } x \geq y, \text{ then } A \otimes x \geq A \otimes y, A \otimes' x \geq A \otimes' y, \tag{1.7}$$

$$A \otimes (B \otimes' C) \leq (A \otimes B) \otimes' C, \tag{1.8}$$

$$A \otimes' (B \otimes C) \geq (A \otimes' B) \otimes C, \tag{1.9}$$

$$A \otimes ((-A^T) \otimes' B) \leq B \leq A \otimes' ((-A^T) \otimes B), \tag{1.10}$$

$$(A \otimes (-A^T)) \otimes' x \leq x \leq ((-A^T) \otimes' A) \otimes x, \tag{1.11}$$

$$x \otimes' (A \otimes (-A^T)) \leq x \leq x \otimes ((-A^T) \otimes' A). \tag{1.12}$$

Definition 4 (Greatest subsolution) *For $A \in \mathbb{R}_{max}^{m\times n}, x \in \mathbb{R}_{max}^{n}, b \in \mathbb{R}_{max}^{m}$, the greatest subsolution $\overline{x}$ of equation $Ax = b$ is defined as*

$$\overline{x} = \max\{x | Ax \leq b\}, \tag{1.13}$$

where $\max(x, y)$ *is a vector with elements* $\max(x_i, y_i)$ *and "$\leq$" is taken elementwise.*

Lemma 1 *The greatest subsolution $\overline{x}$ of $Ax = b$ is*

$$\overline{x} = (-A^T) \otimes' b. \tag{1.14}$$

Synchronisation and concurrency are two basic phenomena of DES dynamics. Synchronisation requires the fulfilment of several conditions. Fig. 1.1 is a simple synchronisation example. The happening of event c depends on both happenings of event a and event b. A discrete event system exhibiting only synchronisation properties can be easily modelled as a linear system using max-plus algebra. For the system shown in Fig. 1.1, the happening time of event c is:

$$t_c = 2 \otimes t_a \oplus 2 \otimes t_b.$$

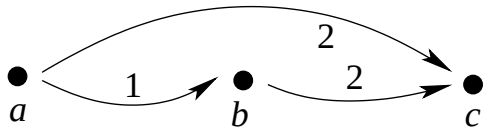

Figure 1.1: Synchronisation

Concurrency corresponds to for example, the competition for shared resources or confliction. For systems also exhibiting the concurrency aspect, an additional high level scheduling module may be included.

Chapter 2

Hierarchical control structure for noncyclic DES

For complex discrete event systems, challenging control issues include coordination of sub-plants (for example, trains in rail networks, products in flexible manufacturing systems) and appropriate handling of unexpected or uncertain events such as blockings, mechanical failure. Under such circumstances, it is important that the coordination plan and its implementation strategy should adapt to the dynamic environment. Inspired by the simplicity and the effectiveness of max-plus algebra models for discrete event systems, a new hierarchical planning and control scheme for DES control in a generic setting is proposed in this research.

This chapter describes the proposed control structure in general and explains how it works for *noncyclic* DES in railway transportation systems, i. e. for a DES with the feature of "running only one cycle". A general introduction of the proposed scheme is given in Section 2.1. Sections 2.2-2.4 describe each part of the scheme in detail. Simulation results for a noncyclic train-track system example are shown in Section 2.5. Finally, Section 2.6 provides the chapter conclusion.

2.1 General control structure

The proposed hierarchical control structure for discrete event systems is shown in Fig. 2.1. This architecture can be used to solve DES control problems with or without cyclic features. In general, it is composed of a two-level structure along with an independent C/D

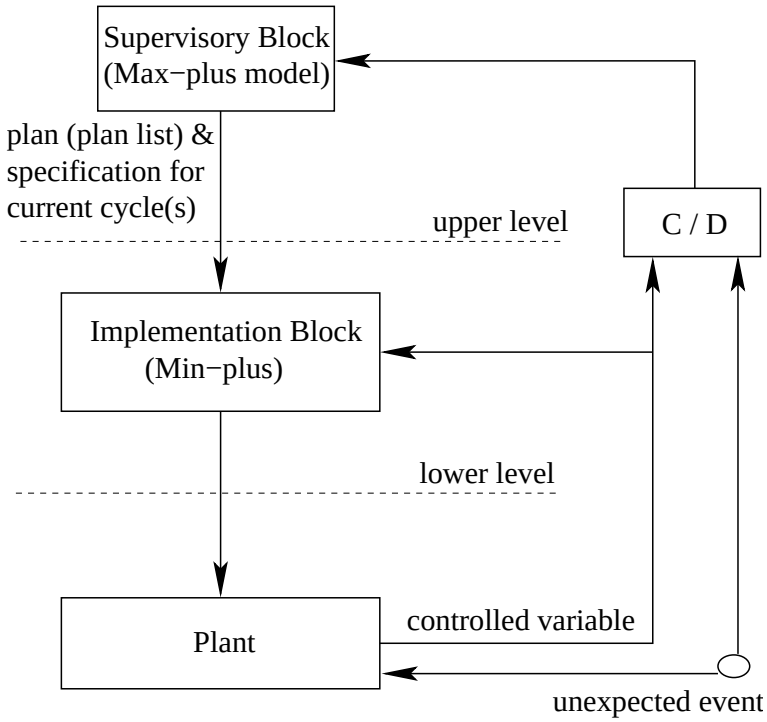

Figure 2.1: Control structure

(continuous/discrete) module which transforms information from the continuous plant to the timed DES framework. On the upper level, a max-plus algebra model is introduced to determine the sequence of events (i. e. the time optimal plan for noncyclic systems, or, the time optimal plan list for cyclic systems) which optimises the user-designated objective function. It also provides the event time specifications needed by the lower level. On the lower level, min-plus algebra not only coordinates the behaviour of sub-plants but also minimises energy consumption. In brief, the control system has a hierarchical structure: the upper level is a supervisory block, which produces the optimal plan or plan list for the lower level. The lower level is an implementation block which operates on the basis of the specifications generated by the upper level.

In most parts of this thesis, focus is on rail traffic DES applications. For such systems, events correspond to specific trains crossing specific locations within the rail system. As a result, a plan generated by the upper control level specifies the sequence of trains and track segments where trains pass each other. The lower level generates velocity reference signals for the trains to implement the given optimal plan. For other DES applications, the terms “train” and “track (segment)” have of course to be replaced by different ones. For example, in flexible manufacturing systems, “product” and “machine” can be used

instead. In a general context, “train” stands for “user” while “track (segment)” stands for “resource”:

General context	user	resource
Train-track system	train	track segment
Flexible manufacturing system	product	machine
Chemical batch plant	batch	device / unit
⋮		

2.2 Upper level — supervisory block

Max-plus algebra is a convenient modelling and analysis tool not only for cyclic systems but also for systems with noncyclic behaviour. In this chapter, we will focus on the latter. Before discussing the max-plus model on the supervisory level, appropriate terminology for train-track systems is provided.

2.2.1 Terminology

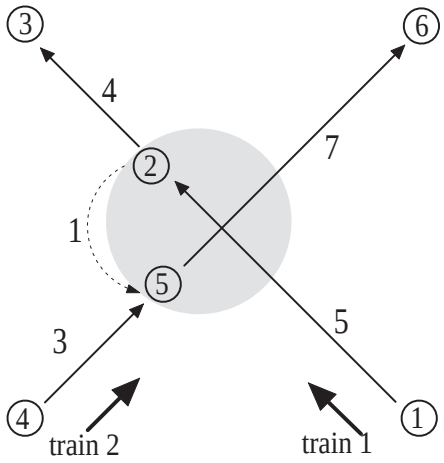

Figure 2.2: A simple train-track network

Consider the simple train-track network illustrated by Fig. 2.2, where two trains are travelling along their tracks. This is pictured as a graph with nodes and arcs. *Nodes* signify *events* where specific trains cross specific locations within the train-track system. For example, node 1 and node 4 represent the starting events of train 1 and train 2 respectively.

Node 3 and node 6 represent the events of train 1 and train 2 arriving at their terminal stations. The *arcs* between the nodes represent the minimum time needed. For example, the event represented by node 2 may not happen earlier than 5 time units after the event time of node 1. The arc 21 (from node 1 to node 2) is a *travelling arc* representing minimum train travelling times. The light grey part of Fig. 2.2 represents the crossing area, which only one train can occupy at any instant of time in order to ensure safe crossing. Node 5 represents the event of train 2 entering the crossing area. Node 2 represents the event of train 1 leaving the crossing area. A "control arc" (dashed arc from node 2 to node 5 in Fig. 2.2) is also introduced. This *control arc* reflects a safety requirement: the earliest time at which train 2 is allowed to enter the crossing area is 1 time unit after train 1 has left it.

The time associated with an arc is called *weight* of the arc. The weights of the arcs can be represented by matrices A_{01} and A_{02}. The elements of A_{01} correspond to weights of travelling arcs while the elements of A_{02} correspond to weights of control arcs. For example, the element $(A_{01})_{ij}$ is the weight of travelling arc ij, i. e. the minimum travelling time needed from node j to node i. If such an arc does not exist, the corresponding matrix element is $\epsilon = -\infty$ which means that there is no time constraint from node j to node i. Thus, in this thesis, the elements $(A_{01})_{ij}, (A_{02})_{ij} \in \mathbb{R}^+ \cup \{-\infty, 0\}$. The matrices A_{01} and A_{02} for the example shown in Fig. 2.2 can be represented as:

$$A_{01} = \begin{bmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ 5 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 4 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 3 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 7 & \epsilon \end{bmatrix}, \quad A_{02} = \begin{bmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}.$$

A *path* is a sequence of arcs connecting a sequence of nodes. When the initial and the final node coincide, it is called a *circuit* [3]. The *length* of a path or a circuit is equal to the number of arcs of which it is composed. The *weight* of a path or a circuit is the sum of the weights of all arcs contained in the path or the circuit. For physically meaningful systems, the graph does not contain circuits.

For systems with noncyclic behaviour, the supervisory block on the upper level will have the goal of determining the optimal plan. A plan in general is a sequence of events. Specifically, in train-track systems, a *plan* shows the sequence of trains and the track segments where trains pass each other.

While determining the optimal plan is an online task, generation of all feasible plans can be done offline. In this thesis, we assume that the network is not too large, so that the complete set of feasible plans can be established. Once all feasible plans are given, it is possible to select the optimal plan online and pass it to the implementation block.

2.2.2 Max-plus algebra models — implicit and explicit

For the train-track network shown in Fig. 2.2, let x_i denote the event time for node i. Then, $\underline{x}_i$ is the earliest possible event time for node i, i. e. the lower bound of x_i. Train 1 and train 2 start their trips at time 0. The input to the system is a vector $u \in \mathbb{R}^2_{max}$ representing the earliest starting times: $u = [0 \quad 0]^T$. The output of the system is defined by the earliest possible arrival times of the trains, $Y = [\underline{x}_3 \quad \underline{x}_6]^T$. It can be conveniently computed by

$$\underline{X} = \begin{bmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ 5 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 4 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 3 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 7 & \epsilon \end{bmatrix} \otimes \underline{X} \oplus \begin{bmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix} \otimes \underline{X} \oplus \begin{bmatrix} e & \epsilon \\ \epsilon & \epsilon \\ \epsilon & \epsilon \\ \epsilon & e \\ \epsilon & \epsilon \\ \epsilon & \epsilon \end{bmatrix} \otimes u, \quad (2.1)$$

$$Y = \begin{bmatrix} \epsilon & \epsilon & e & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & e \end{bmatrix} \otimes \underline{X}, \quad (2.2)$$

where $\underline{X} = [\underline{x}_1\, \underline{x}_2\, \underline{x}_3\, \underline{x}_4\, \underline{x}_5\, \underline{x}_6]^T$ is the earliest possible event time vector.

In general, (2.1) and (2.2) can be written as:

$$\underline{X} = A_{01}\underline{X} \oplus A_{02}\underline{X} \oplus Bu, \quad (2.3)$$

$$Y = C\underline{X}, \quad (2.4)$$

where the elements of matrix A_{01} represent minimal travelling times, the elements of matrix A_{02} represent minimal times associated with control arcs and B, C are appropriate input and output matrices.

Of course, we can combine the two matrices A_{01} and A_{02} into one matrix:

$$A_0 = A_{01} \oplus A_{02}. \quad (2.5)$$

For noncyclic systems, the general max-plus model therefore has the following form:

$$\begin{aligned} \underline{X} &= A_0 \otimes \underline{X} \oplus B \otimes u, && (2.6)\\ Y &= C \otimes \underline{X}. && (2.7) \end{aligned}$$

The appearance of $\underline{X}$ in the right hand side of equation (2.6) implies that the max-plus model (2.6)-(2.7) is an *implicit model*. Repeated insertion of (2.6) into itself provides:

$$\begin{aligned} \underline{X} &= A_0 \otimes (A_0 \otimes \underline{X} \oplus B \otimes u) \oplus B \otimes u \\ &= A_0^2 \otimes \underline{X} \oplus [A_0 \oplus I] \otimes Bu \\ &= A_0^2 \otimes (A_0 \otimes \underline{X} \oplus B \otimes u) \oplus [A_0 \oplus I] \otimes Bu \\ &= A_0^3 \otimes \underline{X} \oplus [A_0^2 \oplus A_0 \oplus I] \otimes Bu \\ &\vdots \\ &= A_0^n \underline{X} \oplus [A_0^{n-1} \oplus \cdots \oplus A_0 \oplus I] \otimes Bu, \qquad (2.8) \end{aligned}$$

where n is the dimension of the square matrix A_0. It is also the number of nodes of the network represented by A_0. Recalling the definition of "$\otimes$" for matrices, we have

$$(A_0^2)_{ij} = (A_0 \otimes A_0)_{ij} = \bigoplus_k ((A_0)_{ik} + (A_0)_{kj}), \qquad (2.9)$$

i. e. the element $(A_0^2)_{ij}$ is the largest weight of all paths with the following two properties:

1. The path is from node j to node i,
2. The length of the path is 2.

Similarly, the element $(A_0^n)_{ij}$ is the largest weight of all paths from node j to node i with length n. Furthermore, for a n-node network, if a path is of length n, then at least one node appears twice in the path. Thus, for a n-node network, there must be a circuit in a path of length n if the path exists.

For a physically meaningful system, the graph associated with A_0 does not contain circuits, i. e. the largest weight of all paths of length n is ϵ, i. e. A_0^n will only contain ϵ elements. In fact, we have

$$A_0^k = N, \forall k \geq n. \qquad (2.10)$$

Therefore, in the absence of circuits, the last identity in (2.8) represents an *explicit max-plus algebra model*:

$$\begin{aligned} \underline{X} &= A_0^* \otimes B \otimes u, && (2.11)\\ Y &= C \otimes \underline{X}, && (2.12) \end{aligned}$$

where

$$A_0^* = [A_0^{n-1} \oplus \cdots \oplus A_0 \oplus I]. \tag{2.13}$$

Assume that the system input u consists of initial values of lower bounds for the state vector X, i. e. $u = \underline{X}\_in$ (see Section 2.3 for further information), then (2.11) and (2.12) can be rewritten as:

$$\underline{X} = A_0^* \otimes B \otimes \underline{X}\_in, \tag{2.14}$$

$$Y = C \otimes A_0^* \otimes B \otimes \underline{X}\_in. \tag{2.15}$$

2.2.3 Feasible plans and feasible plan set

A complex train-track network usually contains track segments which are used by several trains. For safety reasons only one train is allowed to occupy such a resource at any instant of time. For each resource (track segment), there are several possible train sequences, usually causing different combinatoric possibilities. Correspondingly, for the whole system consisting of a number of trains, there are several feasible plans. For example, while the control arc in Fig. 2.2 realises a special feasible plan, another feasible plan is to make train 2 pass the crossing area before train 1 enters it. Of course, to model this plan there is the need of adding nodes to the graph in Fig. 2.2. It is straightforward to find the time optimal plan within the set of feasible plans by simulating the max-plus model for each feasible plan.

In the following, it is shown by use of an example, how the set of feasible plans and their max-plus models can be determined.

Matrix A_{01} contains minimum travelling times for segments of the network. Clearly, these times can be easily determined from the length of tracks and the achievable maximal speed of trains, if acceleration and braking can be neglected, i. e. if it were possible for trains to always move with maximal velocity.

On double-line track segments, passing trains will not interfere with each other. This is different for single-line track segments, which can only be occupied by one train at a time. So control arcs are necessary to ensure safe travelling of trains on these track segments. The structure of the control arcs carries the information about the sequence of trains. The weight of a control arc determines a possible safety time that should pass between the trains' movements. If two trains compete for the travel on one single-line track, there are two possibilities: either train 1 or train 2 can go first. If a train-track

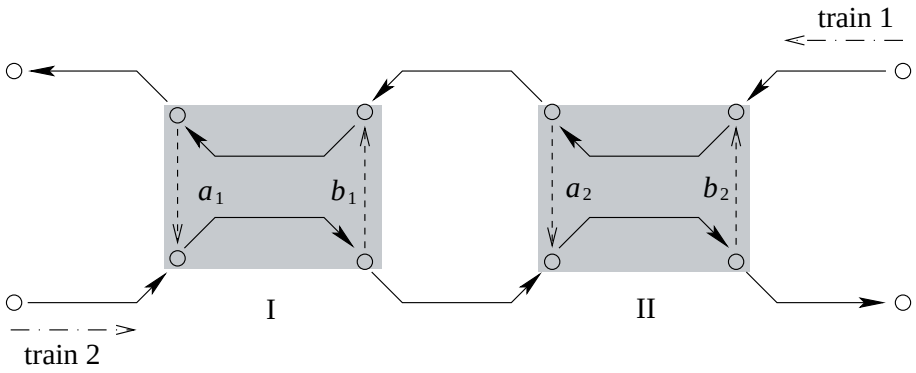

Figure 2.3: A network with all possible control arcs

network includes several single-line track segments, the different combinations of those possibilities generate different plans (with different A_{02}s). Of course, some of them could be infeasible, namely cause blocking.

Fig. 2.3 demonstrates the directed graph of a simple network with two single-line track segments and two trains, where train 1 moves from right to left and train 2 moves in opposite direction. For single-line segment I, the control arc labelled a_1, with $a_1 \geq 0$ represents the fact that train 2 may occupy this segment at the earliest a_1 time units after train 1 has left it. The control arc labelled b_1 states the reverse sequence. Accordingly, only one of those two arcs can exist in any one plan. Instead of erasing a non-existent arc from a graph, we can also label it by ϵ. Hence, by definition, non-existing arcs have weight $-\infty$. Similarly, in segment II, the arcs labelled a_2 and b_2 represent the two possibilities. Table 2.1 shows all four combinations of the selection of arcs. Among these combinations, the case $a_1 \geq 0$, $b_2 \geq 0$ causes a conflict (blocking), thus this is an infeasible plan. This situation is illustrated in Fig. 2.4.

For large train-track network systems, we need a simple criterion to determine whether a plan is infeasible or not. Fig. 2.4 shows that the conflict of combing $a_1 \geq 0$ and $b_2 \geq 0$ comes with a circuit in the directed graph. A circuit with a positive weight causes a contradiction, because the events within the circuit would have to "happen some time before they actually happen". So the problem of determining infeasible plans can be solved by checking for the existence of circuits.

Within the A_0 matrix, the diagonal entry $A_0(i,i)$ represents an arc from node i to node i itself. If $A_0(i,i)$ is not ϵ, there is a circuit of length 1 containing node i. Consider now the set of all circuits of length 2 containing node i. From the matrix product definition under the max-plus scheme, $A_0^2 = A_0 \otimes A_0$, $A_0^2(i,i)$ represents the largest weight of any

Table 2.1: Combinations of control arcs

combination	**segment where train 1 and train 2 pass**	**plan**
$a_1, a_2 \geq 0$ $b_1 = b_2 = \epsilon$	double-line track segment on the left of area I	plan 1
$b_1, a_2 \geq 0$ $a_1 = b_2 = \epsilon$	double-line track segment between area I and area II	plan 2
$a_1, b_2 \geq 0$ $b_1 = a_2 = \epsilon$	not feasible, see Fig. 2.4	/
$b_1, b_2 \geq 0$ $a_1 = a_2 = \epsilon$	double-line track segment on the right of area II	plan 3

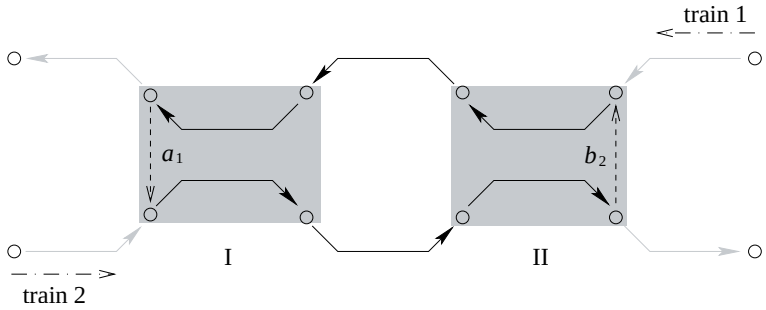

Figure 2.4: Combination of $a_1 \geq 0$ and $b_2 \geq 0$ causes a circuit

circuit of length 2 containing node i. Hence, if $A_0^2(i,i)$ is not ϵ, there is at least one circuit of length 2 containing node i. Similarly, if $A_0^k(i,i)$ is not ϵ, there is at least one circuit of length k from node i to node i. On the other hand, if there is a circuit, the possible maximum length is n. In order to determine whether a plan is infeasible, we just need to calculate A_0^k, $k \leq n$, in the max-plus sense. Then we have:

$$\exists k \leq n, A_0^k(i,i) > \epsilon \;\Leftrightarrow\; A_0 \text{ is infeasible.} \tag{2.16}$$

Correspondingly, for a feasible plan, considering (2.10), the following properties hold:

$$\forall k, i \in \mathbb{Z}^+, k < n, i \leq n, \qquad A_0^k(i,i) = \epsilon, \tag{2.17}$$

$$\forall k \in \mathbb{Z}^+, k \geq n, \qquad A_0^k = N. \tag{2.18}$$

Based on the set of max-plus models for all feasible plans, the supervisory block finds the optimal plan by simulation. Note that this seemingly offline task usually should also be resolved online so that the supervisory level can handle unexpected events promptly. The

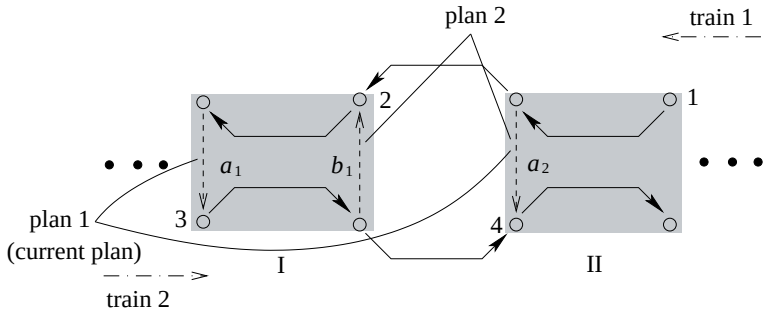

Figure 2.5: Control arcs for plan 1 and plan 2

number of feasible plans reduces as the trains progress through the network. As described above, building the set of feasible plans is based on the different possible combinations of control arcs at the single-line track segments. During runtime, once a train enters a single-line track segment, there is no need to consider the competing control arcs on this track segment because the choice has already been made. The plans which correspond to the competing control arcs may be eliminated from the set of feasible plans. The described online update of the set of feasible plan can be performed by observing the occurrence of events corresponding to in-nodes.

Definition 5 (In-node) *Consider a control arc of a plan preventing collision of two trains on a single-line track segment. The node to which the control arc points is called an in-node of the plan with respect to the considered trains and the considered track segment.*

Going back to the example from Fig. 2.3, Fig. 2.5 shows the control arcs of plan 1 and plan 2. The in-nodes of plan 1 and plan 2 with respect to train 1, train 2 and track segment II are the same (node 4), which shows that both plans include the same control arc for track II. When the system runs under either of the two plans, it is always possible to switch to the other plan, at least in the circumstance where track II is the only single-line track segment in the system. However, there is another single-line segment. The in-node of plan 1 with respect to train 1, train 2 and track segment I is node 3. The corresponding in-node of plan 2 is node 2. For the system already running with one plan, it is impossible to switch to the other plan once the event corresponding to node 3 or node 2 has occurred. Specifically, in the case shown in Fig. 2.5, suppose the current plan is plan 1. It is then impossible to switch to plan 2 if train 1 passed the location corresponding to event 2.

If the system is operated under a certain plan, the set of feasible plans can be updated using the following procedure:

Step 1 List the sets of in-nodes, $Ins1$ and $Ins2$, for the operated plan and for the plan to be checked, respectively. The in-nodes in both sets should be listed in the same order with respect to the track segments and pairs of trains in considered. For instance, in Fig. 2.5, $Ins1$ and $Ins2$ are:

operated plan:plan 1	$Ins1$={3, 4}
plan to be checked:plan 2	$Ins2$= {2, 4}

Step 2 For the plan to be checked, find the key in-node set (denoted by $kins$), whose elements are not in the intersection of corresponding 1-entry-subsets of $Ins1$ and $Ins2$:

$$kins = \bigcup_{i}(\ Ins1_i \text{ SETXOR } Ins2_i \ ). \tag{2.19}$$

where $Ins1_i$, $Ins2_i$ are the sets consisting of the ith element of $Ins1$, $Ins2$ respectively and

$$Ins1_i \quad \text{SETXOR} \quad Ins2_i := (Ins1_i \cup Ins2_i) \backslash (Ins1_i \cap Ins2_i).$$

Again in Fig. 2.5, the $kins$ for plan 2 can be found as follows:

$$\begin{aligned}
(Ins1)_1 \quad \text{SETXOR} \quad (Ins2)_1 &= \{3\} \quad \text{SETXOR} \quad \{2\} = \{3, 2\}, \\
(Ins1)_2 \quad \text{SETXOR} \quad (Ins2)_2 &= \{4\} \quad \text{SETXOR} \quad \{4\} = \emptyset, \\
kins = \bigcup_{i=1}^{2}(\ Ins1_i \text{ SETXOR } Ins2_i \ ) &= \{3, 2\} \cup \emptyset = \{3, 2\}.
\end{aligned}$$

Since the elements of Ins associate to pairs of competing trains on corresponding shared single-line track segments, instead of from 1-entry-subsets of Ins, $kins$ can be easily obtained from subsets corresponding to pairs of competing trains. Denote by $Ins1_{(i,j)}$ the subset of $Ins1$ related to train i and train j on the considered segment. Then:

$$kins = \bigcup_{\substack{i,j \\ j>i}} Ins1_{(i,j)} \quad \text{SETXOR} \quad Ins2_{(i,j)}. \tag{2.20}$$

In Fig. 2.5, there is only one pair of trains competing, thus

$$kins = Ins1_{(1,2)} \text{ SETXOR } Ins2_{(1,2)} = \{3, 4\} \text{ SETXOR } \{2, 4\} = \{3, 2\}.$$

Similarly, for plan 3 which is not shown in Fig. 2.5, its $kins$ is $\{1, 2, 3, 4\}$ in which event 1 and event 2 relate to the movement of train 1, event 3 and event 4 for train 2 as well. There is a further simplification: if a $kins$ set contains several nodes relating to the same train, we only need to retain the node corresponding to the earliest event for this train. The resulting set is called "$Kins$". In this way, the maximum number of elements for a $Kins$ is the number of the trains in the network. For example, for the $kins$ set of plan 3, nodes 2 and 1 relate to the same train, with the event corresponding to 1 occurring earlier. Hence node 2 can be dropped. With the same argument, node 4 can be omitted. Thus, the resulting $Kins$ sets for plan 2 and plan 3 are:

$$Kins_{\text{plan 2}} = \{2, 3\}, \quad Kins_{\text{plan 3}} = \{1, 3\}.$$

We can store $Kins$ information in a KN matrix:

$$KN = \begin{bmatrix} +\infty & +\infty \\ 2 & 3 \\ 1 & 3 \end{bmatrix},$$

where the elements $+\infty$ in the first row mean that no events will deactivate plan 1. $(KN)_{ij}$ shows the key in-node of train j for the plan $i (i \neq 1)$ which needs to be checked. Note that for each plan, a different KN matrix is computed off-line and stored.

Step 3 Update the feasible plan set online. If a certain plan is in operation, i. e. the corresponding KN matrix is valid, we only need to check whether an event occurs that is listed in one or more of the rows of KN. If this happens, the plans corresponding to these rows become infeasible.

In practice, $(KN)_{ij}$ can be simplified further if there is a fixed temporal order for the events corresponding to nodes in one row. E. g., if plan 1 is in operation, node 3 in row 2 and 3 of the KN matrix can be neglected. The resulting KN is

$$KN = \begin{bmatrix} +\infty & +\infty \\ 2 & +\infty \\ 1 & +\infty \end{bmatrix}.$$

Note that Step 1 and Step 2 are implemented offline. Therefore, for each plan, there is a corresponding KN matrix which can be used online to check the possibility of switching to other plans. The shrinking of the feasible plan set while trains progress through the network also helps to speed up the online optimisation.

2.2.4 Online plan optimisation

During runtime, the progress of the trains is observed by the supervisory block. At regular time instants (e. g. in an equidistant discrete time scheme), simulation is performed for all feasible plans to compare the values of the designated objective function. The objective function is evaluated from the output vector Y. An example for the objective function would be the last train's arrival time, i. e. we want to minimise $\max_i(Y_i)$.

2.3 C/D block

Based on the set of max-plus models for all feasible plans, the optimal plan with respect to a specific objective function can be easily determined by simulation. As indicated in the system control structure diagram (Fig. 2.1), to generate the optimal plan for the sequence and timing of the trains, the supervisory block not only needs information on the track topology but also real time information on the current status of the trains. Max-plus simulation of each plan is based on the assumption that there is no unexpected event and that each train either waits for a synchronisation condition to be met or otherwise may move at its maximum velocity. If any unexpected incident happens, it may affect some trains' travelling times either directly by blocking them or indirectly via synchronisation with other trains. If it does happen, the runtime event times will not match the event times calculated by a-priori model simulation any more. In order to achieve online plan optimisation, rescheduling has to be performed at runtime. For this, the state vector needs to be reinitialised based on the current status of the trains at time t_k, where t_k is a time instant in a regular sampling grid. The reinitialised state vector is denoted by $\underline{X}\_in$. In general, it contains three different kinds of elements:

1. For an event j which already happened, the reinitialised state value is the actual event time, i. e.

$$\underline{x}\_in_j(t_k) = x_j. \tag{2.21}$$

2. For the next event of any train, the reinitialised state has to be calculated based on the current position of the respective train.

Suppose at a time instant t_k, the distance a train has to move before it reaches the location associated with the next event j is $d_j(t_k)$. As $\underline{x}\_in_j$ is the earliest possible event time for

event j, the reinitialised value will be:

$$\underline{x}\_in_j(t_k) = \begin{cases} t_k + \frac{d_j(t_k)}{v_{max}} & \text{unblocked trains} \\ t_{rel} + \frac{d_j(t_k)}{v_{max}} & \text{blocked trains,} \end{cases} \tag{2.22}$$

where t_{rel} is the estimated release time for the blocked train. If this time cannot be estimated beforehand, recalculation of the plan is based on the assumption of immediate release, i. e.

$$t_{rel} = t_k. \tag{2.23}$$

3. For other events j in the future, the reinitialised state variables still remain undefined, i. e.

$$\underline{x}\_in_j(t_k) = \epsilon. \tag{2.24}$$

Therefore, the reinitialised state variable at time t_k is:

$$\underline{x}\_in_j(t_k) = \begin{cases} x_j & j \text{ is a past event} \\ \text{from (2.22)} & j \text{ is a next event for any train} \\ \epsilon & \text{otherwise.} \end{cases} \tag{2.25}$$

Then, the updated vector of event times can be obtained for all feasible plans by simulation, i. e. by evaluating (2.11) with $u = \underline{X}\_in(t_k)$ and $B = I$ ($B_{ii} = e, B_{ij} = \epsilon$ for $i \neq j$):

$$\underline{X}(t_k) = A_0^* \otimes \underline{X}\_in(t_k), \tag{2.26}$$

$$Y(t_k) = C \otimes A_0^* \otimes \underline{X}\_in(t_k). \tag{2.27}$$

2.4 Lower level — implementation block

The upper level of the proposed DES control structure determines the sequence of events optimising the given objective function as well as the detailed time specifications for events. These results are implemented by the lower level. With the help of min-plus algebra, the lower level exploits the remaining degrees of freedom and provides an energy saving implementation while maintaining event times determined by the upper level.

2.4.1 EPET and LNET

The output of the supervisory block generates the time optimal plan, i. e. the time specifications $\underline{X}$ for the events. $\underline{X}$ provides the earliest possible time for each event to occur,

and is therefore referred to as EPET (earliest possible event time) specification. If EPET were implemented, trains would always move at maximum speed, or otherwise stop to wait until the synchronisation conditions are met. Clearly, this is undesirable from an energy saving point of view. However, an overall optimisation approach, minimising energy consumption for all trains in common, yields a large nonlinear problem and therefore is not realizable for large systems. Instead, a suboptimal more conservative approach is used here in determining a velocity signal separately for each train. This is done as follows: first, as an upper bound for the event times, LNET (Latest Necessary Event Time) is derived for each train. This is done in such a way that the train's arrival time at the final destination according to EPET will be met. It is also ensured that LNET does not violate the EPET schedule of the other trains. In order to generate the LNET specifications, min-plus algebra, the dual system of max-plus algebra, is used.

Let QS be the index set for all events corresponding to the arrival of trains at their final destinations and PS_m the index set for all events related to train m, then the LNET specifications $\overline{X}_m$ for train m can be calculated as

$$\overline{X}_m(t_k) = (-(A_0^T))^{*'} \otimes' \underline{XR}_m(t_k), \tag{2.28}$$

where

$$(\underline{XR}_m)_i(t_k) = \begin{cases} \underline{x}_i(t_k), & i \in QS \text{ or } i \notin PS_m \\ +\infty, & \text{otherwise.} \end{cases} \tag{2.29}$$

The min-plus algebra equation (2.28) represents "backward simulation". Note that the following relation holds for the $*$ and $*'$ operations of max-plus algebra and min-plus algebra:

$$(-A^T)^{*'} = -(A^*)^T. \tag{2.30}$$

Therefore, (2.28) can be rewritten as

$$\overline{X}_m(t_k) = (-(A_0^*)^T) \otimes' \underline{XR}_m(t_k). \tag{2.31}$$

In fact, we have the following corollary about LNET:

Corollary 1 (LNET) *For system without cyclic behaviour, the LNET specification $\overline{X}_m$ for train m which ensures the earliest final event time of trains as well as the earliest event times of other trains can be calculated by (2.29) and (2.31).*

Proof Since the greatest subsolution of $Ax = b$ is $\overline{x} = (-A^T) \otimes' b$ (see Lemma 1), (2.31) represents the greatest subsolution of $A_0^* \otimes X_m = \underline{XR}_m$. From the definition of

greatest subsolutions (Definition 4), it is immediately clear that

$$A_0^* \otimes \overline{X}_m \leq \underline{XR}_m, \tag{2.32}$$

and

$$\forall X_m \in \{X_m | A_0^* \otimes X_m \leq \underline{XR}_m\}, X_m \leq \overline{X}_m. \tag{2.33}$$

(2.32) means that $\overline{X}_m$ calculated using (2.31) ensures the event times specified by $\underline{XR}_m$. Furthermore, considering (2.33), $\overline{X}_m$ represents the largest and therefore the latest necessary event times LNET.

□

$\underline{X}$ represents the earliest possible event times for all trains. Therefore, now, for each train m, its EPET and its LNET $\overline{X}_m$ are available. Within this corridor, the velocity signal can be optimised locally for each train.

For the problem of driving a train from one station to the next, an energy optimal policy consists of four subsequent parts: maximum acceleration, speedholding, coasting and maximum breaking e. g. [37, 53]. For long distances, the speedholding phase becomes dominant. Then, assuming the journey must be completed within a given time, the holding speed (i. e. the energy optimal velocity) is approximately the total distance divided by the total time between adjacent nodes [52]. In the following, we neglect acceleration and braking effects, and an energy optimal trajectory results from minimising

$$J_m = \int v_m^2 dt, \tag{2.34}$$

where integration is between starting and arrival time of train m. Then,

$$J_m = \sum_{i=1}^{n} \frac{(S_i - S_{i-1})^2}{(t_i - t_{i-1})} \tag{2.35}$$

s.t.

$$t_0 = 0 \tag{2.36}$$

$$t_i = t_{i-1} + \frac{S_i - S_{i-1}}{(v_m)_i} \tag{2.37}$$

$$t_{1E} \leq t_1 \leq t_{1L} \tag{2.38}$$

$$t_{2E} \leq t_2 \leq t_{2L} \tag{2.39}$$

$$\vdots$$

$$t_{(n-1)E} \leq t_{n-1} \leq t_{(n-1)L} \tag{2.40}$$

$$t_{nE} = t_n = t_{nL}, \tag{2.41}$$

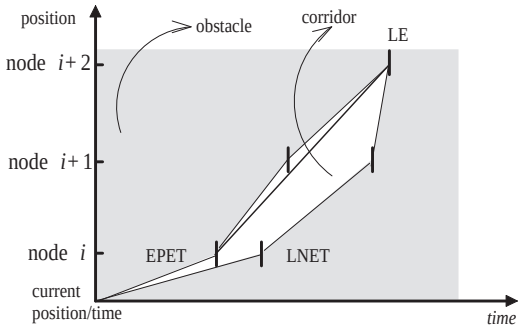

Figure 2.6: Energy optimal trajectory(bold line)

where $S_0 = 0$, S_i $(i = 1, 2, \cdots, n)$ is the distance from the current position of train m to the place where event i happens. t_{nE} and t_{nL} are EPET and LNET of event n, respectively. Minimisation of (2.35) with respect to $t_i, i = 1, \cdots, n-1$ gives the energy optimal trajectory. An example for an energy optimal trajectory is given in Fig. 2.6.

It may take a long time to solve the above nonlinear optimisation problem when the number of events is large. Also, sometimes instead of the global optimal, the numerical solution is likely to be only a local optimal. To ensure globally optimal solutions, two computational geometry algorithms can be used here. One involves the famous Dijkstra's algorithm and may also be quite slow to get the solution. The other is a proposed intuitive algorithm which speeds up the computational procedure. Both algorithms are described in the next subsection.

2.4.2 Dijkstra's algorithm and a new intuitive algorithm

Before discussing the detailed algorithms, we first show the solutions for our minimal energy problem and the shortest path problem coincide.

In Fig. 2.7, O and F are two points with fixed coordinates. The s-coordinate of an intermediate point M is also fixed. x is the directed horizontal distance from M to the straight line OF. Fig. 2.7 (a) and (b) correspond to the situations when M is located on the right and on the left of OF respectively. In Fig. 2.7 (a), $x > 0$ and M is restricted within the gray area corresponding to b_1, i. e. $x \in [b_1, \frac{b}{a}-b)$, where $0 < b_1 < (\frac{b}{a}-b)$. In Fig. 2.7 (b), $x < 0$ and M is restricted within the gray area corresponding to b_2, i. e. $x \in (-b, -b_2]$, where $0 < b_2 < b$. The first problem is to find the value of x so that the path OMF is the shortest path.

The second problem is to find the minimum energy trajectory. It can also be depicted

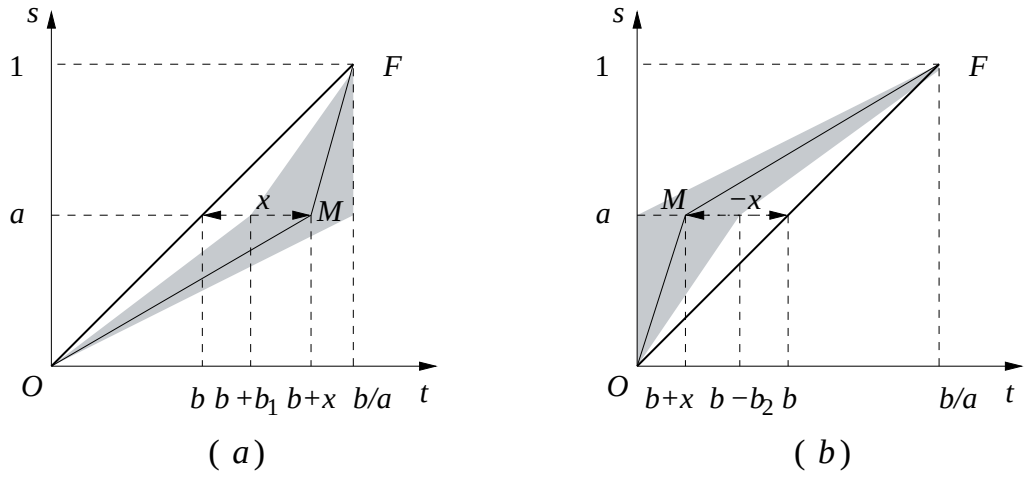

Figure 2.7: Minimum energy / shortest path

graphically as Fig. 2.7. The coordinate s represents location of events, and t represents the corresponding time specification. For example, in Fig. 2.7 (a), for the intermediate event M which is located on the right of OF, its s-coordinate is fixed. According to the EPET and the LNET specifications, event M is expected to happen during time $[b+b_1, \frac{b}{a})$, i. e. $x \in [b_1, \frac{b}{a} - b)$. Similarly, in Fig. 2.7 (b), $x \in (-b, -b_2]$ and event M can happen during time $(0, b - b_2]$. The solutions to these two problems coincide (see Appendix A.1 for the proofs):

Solution 1: Shortest path problem In Fig. 2.7, it is clear that the shortest path between point O and point F without any constraints is the straight line OF. But for an OMF path, the shortest one is the path with $x = b_1$ if M is located on the right of OF. When M is located on the left of OF, the shortest path is OMF with $x = -b_2$. In a word, the smallest absolute value $|x|$ gives the shortest path.

Solution 2: Minimum-energy trajectory problem In Fig. 2.7, the minimum-energy trajectory between event O and event F without any constraints is the straight line OF. If there is an intermediate event M with time constraint, when M is located on the right of OF, the minimum-energy trajectory is OMF with $x = b_1$. When M is located on the left of OF, the minimum-energy trajectory is OMF with $x = -b_2$. In a word, the smallest $|x|$ gives the minimum-energy trajectory.

If an algorithm based on the idea of Solution 1 can be used to solve more complicated shortest path problems, which have more intermediate-point constraints, the algorithm

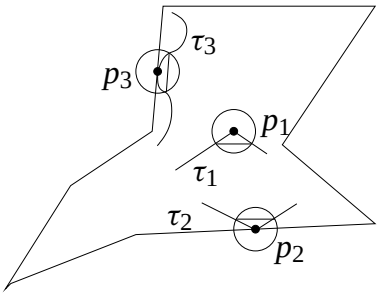

Figure 2.8: Shorten τ

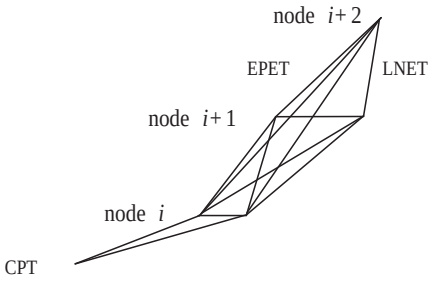

Figure 2.9: Visibility graph of Fig. 2.6

should also work for similar minimum-energy trajectory problems, which have more intermediate-event constraints. These two problems are equal in this regard.

In the field of computational geometry (see e. g. [9]), typical problems concern motion planning. Specifically, the task is to find the shortest path from the start position to the target position without colliding with any of a given set of obstacles. From Fig. 2.6, it is immediately clear that the minimum-energy trajectory problem can be interpreted as a special motion planning problem: the complement of the EPET-LNET corridor (which is composed by a simple polygon[1]) is interpreted as the obstacle, the current position (CPT) as starting position, and LE (the EPET of the last event) as the target position, respectively. Note that unlike in general motion planing problems, in the minimum-energy trajectory problem, the "obstacle" has a very specific structure: For example, the EPET and the LNET of event i are always at the same horizontal level (i. e. they are a pair of same-level vertices). And, EPET_{i+1} locates always above EPET_i.

The intuitive algorithm proposed in this section specifically applies to the velocity determination problem, it might be extended to more general motion planning cases.

A commonly used algorithm for motion planning problems is to construct a visibility graph and then apply Dijkstra's single-source shortest path algorithm to the graph. The idea is introduced in the following, for more detailed information, the reader is referred to e. g. [9].

Let P be a corridor of our minimum-energy trajectory problem and it is a simple polygon having n vertices, and let CPT be a given *source vertex* of P. For each vertex v of P, the shortest path from CPT to v is denoted by $\pi(\text{CPT}, v)$. Then, $\pi(\text{CPT}, v)$ is a polygonal path whose corners are vertices of P. This can be proved by considering a shortest path τ which violates the above mentioned features, for detailed information, the reader is

[1] In geometry, a simple polygon is a polygon which does not intersect itself anywhere.

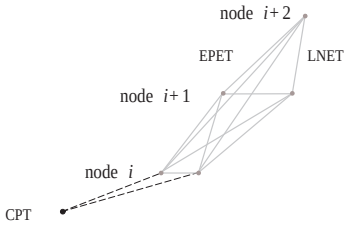

Figure 2.10: Step 1

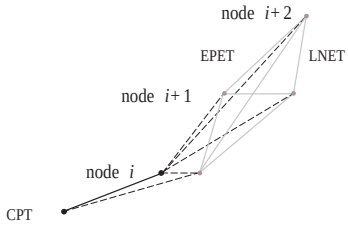

Figure 2.11: Step 2

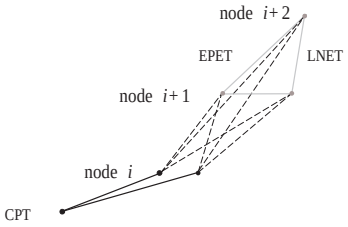

Figure 2.12: Step 3

referred to [9]. For example (Fig. 2.8), let a corner of τ_1 be a point p_1 inside P. Within P, it is always possible to find a small circle centred at p_1, thus the part of path τ_1, i. e. the part which is inside the circle, can always be shortened by a straight line segment connecting two intersecting points of the circle and τ_1. If the corner p_2 is a non-vertex point on an edge of P, it is always possible to find a small circle centred at p_2, and half of the circle is inside P. Again the part of τ_2 which is inside this half circle can be shortened by a straight line segment. Similarly, suppose that part of τ_3 is not a line segment. Since τ_3 is inside P, it is always possible to find a small circle centred at any point p_3 on that part, so that τ_3 can be shortened by the straight line segment connecting two intersecting points. All these shortenings contradict the "shortest" property of τ. Therefore, the shortest path is a polygonal path whose corners are vertices of P.

The visibility graph contains one vertex for each obstacle vertex and one edge for each pair of obstacle vertices that are mutually visible. Two vertices are called mutually visible if the edge connecting these two vertices does not intersect the interior of any obstacle. The visibility graph for Fig. 2.6 is shown in Fig. 2.9.

Now, we briefly review the basic idea behind Dijkstra's algorithm. The first step of Dijkstra's algorithm is to start from the single-source which is CPT as in Fig. 2.9 and store the lengths[2] of edges in which one of the vertices is the single-source, i. e. the edge "CPT-EPET$_i$" and the edge "CPT-LNET$_i$" in the visibility graph, see the dashed lines in Fig. 2.10. At this moment, the set of all vertices is partitioned into two sets: Set 1 contains only the single-source (shown in black) while Set 2 consists of all the other vertices (shown in grey) since they haven't been considered yet.

In the second step, find the minimum length of all stored lengths (in Fig. 2.10, it is the length of the edge "CPT-EPET$_i$", which is shown as a black solid line in Fig. 2.11). This means the shortest path π(CPT, EPET$_i$) is the edge "CPT-EPET$_i$". Move the end vertex (i. e. EPET$_i$) of the shortest path from Set 2 to Set 1. Thus, in Fig. 2.11, the vertex EPET$_i$

[2]The term "length" in this subsection has its normal meaning instead of the definition in Section 2.2.1.

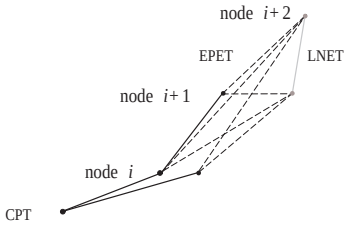

Figure 2.13: Step 4

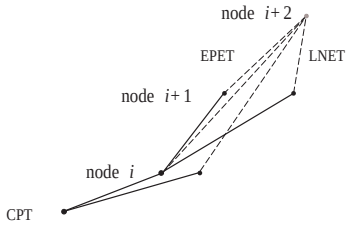

Figure 2.14: Step 5

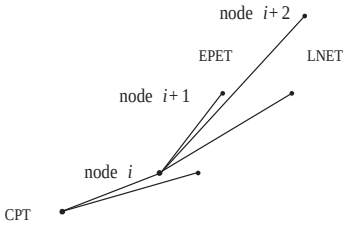

Figure 2.15: Step 6

is the second vertex to be considered, i. e. we now investigate all edges not considered so far if one of their vertices is EPET_i. In our example, these are the edges: "EPET_i-LNET_i", "EPET_i-EPET_{i+1}", "EPET_i-LNET_{i+1}" and "EPET_i-EPET_{i+2}". Connected to the shortest path $\pi(\mathrm{CPT}, \mathrm{EPET}_i)$, each of these edges provides a path starting from the source via the currently considered vertex (EPET_i) to the end point of the respective edges. Calculate the lengths of these paths, store them together with all the lengths stored in the previous step(s) except those of the paths whose two end vertices are the source and a vertex in Set 1, respectively. In our example, the lengths of the paths (black) related to the dashed lines in Fig. 2.11 are all stored.

The second step is now repeated: we select the shortest of the stored paths. In our example, this is the edge "CPT-LNET_i". Hence, the end vertex of this edge (LNET_i) is moved from Set 2 to Set 1, and the shortest path $\pi(\mathrm{CPT}, \mathrm{LNET}_i)$ has been determined. We proceed in an analogous fashion until Set 2 is empty, i. e. until the shortest paths from the single-source CPT to all the other vertices including EPET_{i+2} are found (Fig. 2.12-Fig. 2.15). For systems consisting of a large number of vertices, although the algorithm is straightforward, it still takes quite a long time for both visibility graph construction and searching. For a graph with n vertices, the worst-case running time of Dijkstra's algorithm is $O(n^2)$.

However, we are not concerned with the general problem of finding the shortest path between two vertices in an arbitrary directed graph. As our problem is much more specific, it can be solved by a computationally less demanding, intuitive algorithm. It operates directly on P, the polygon representing the corridor provided by EPET and LNET. By noticing that the shortest path is a polygonal path with some vertices of P as its corners, the proposed intuitive algorithm finds the shortest path by searching these key vertices (kv) using the idea of Solution 1 (thus the algorithm also applies to the energy optimal trajectory problem). Following is the detailed description for this proposed method. Note CPT and LE are always key vertices. All other key vertices are called intermediate key

vertices.

Step 1 Initialisation. Set new key vertex nkv=CPT, last key vertex lkv=LE. Store nkv in the set sp which contains all key vertices found so far.

Step 2 Starting from nkv, check the straight line segment "nkv-lkv".

1. If the straight line segment is contained in P, store the last key vertex lkv in sp to complete the shortest path and finish searching.
2. Otherwise, along the straight line segment "nkv-lkv", find a temporal key vertex tkv:
   $\rightarrow$ Along "nkv-lkv", search from lkv to nkv, identify the point α where this line first leaves the polygon P.
   $\rightarrow$ Below α, identify the pair of same-level vertices v_α which is the closest pair to α.
   $\rightarrow$ For two v_α vertices, set the one which is closer to line 'nkv-lkv" as tkv.

Step 3 Update tkv. Check the straight line segment "nkv-tkv". If the straight line segment leaves P, similar to Step 2.2, search from tkv to nkv along the straight line segment "nkv-tkv" and replace tkv with the new one. Repeat Step 3 until finding a tkv so that the straight line segment "nkv-tkv" does not leave P. This tkv is a key vertex kv . Finally replace nkv with the latest tkv (i. e. kv) and store it into sp.

Step 4 Repeat Step 2-3.

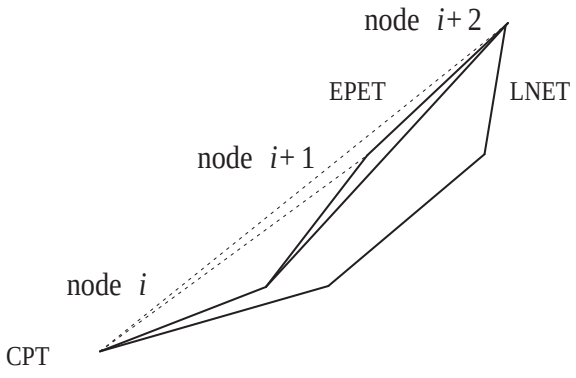

Figure 2.16: Search energy optimal trajectory using the intuitive algorithm

Fig. 2.16 shows the procedure of finding the energy optimal trajectory using the intuitive algorithm for the polygon depicted in Fig. 2.6. From lkv (which is EPET_{i+2}) to nkv

(which is CPT, the first key vertex kv_1), Step 2 gives EPET_{i+1} as tkv. It is then replaced by EPET_i in Step 3. And since new "nkv-tkv" does not leave P, this latest tkv (i. e. EPET_i) is then the second key vertex kv_2 and therefore also the new nkv for the next step. In the next step, it directly gives lkv (i. e. EPET_{i+2}) as kv_3. Thus the optimal trajectory is "CPT-EPET_i-EPET_{i+2}".

We now pick any key vertex (except the source vertex CPT and the target vertex LE) and call it "kv_i". The key vertex above it is called kv_{i+1} while the key vertex below it is called kv_{i-1}. We now consider the subpolygon P_i of P: In subpolygon P_i, the lowest vertex kv_{i-1} is the source vertex of P_i, the highest vertex kv_{i+1} is the target vertex of P_i, all the vertices and corresponding edges located above kv_{i-1} but below kv_{i+1} in P are also vertices and edges of P_i. Then we have the following Lemma.

Lemma 2 *For all key vertices kv_i (except the source vertex CPT and the target vertex LE) given by the intuitive algorithm, the path "kv_{i-1}-kv_i-kv_{i+1}" is the shortest path of the subpolygon P_i of P.*

The proof of the lemma is given in Appendix A.2. To facilitate the proof, we introduce the concept of auxiliary temporary key vertices $rtkv_i$ for every key vertex kv_i with the exception of CPT and lkv. It is the last element in the list of temporal key vertices in the construction of kv_i with the exception of kv_i itself. In the example on page 31, there is only one auxiliary temporary key vertex, namely $rtkv_1 = \text{EPET}_{i+1}$. In the following, we prove that the intuitive algorithm provides the optimal path solution.

Lemma 3 *From CPT to LE, the path sequentially connecting all key vertices is the shortest path in the simple polygon P.*

Proof If the shortest path is a single straight line inside P, the algorithm gives it at the first step: a straight line directly connecting CPT and LE.

A path in P is called *convex inward-P* if every point on the line connecting any two points on the path is in P. A path in P is called *concave inward-P* if inside P, we cannot find a straight line connecting two points on the path. If the shortest path is not a straight line, it cannot be convex inward-P, because a convex part of the path can always be shortened by a straight line segment inside P. Also according to Solution 1, if kv_i is an LNET (or an EPET), then within P_i, "kv_{i-1}-kv_i-kv_{i+1}", the shortest path from kv_{i-1} to kv_{i+1}, is concave inward-P_i and kv_{i+1} is at the right (or left) of straight line "kv_{i-1}-kv_i".

Except CPT and LE, if there is only one intermediate kv, according to Lemma 2, the

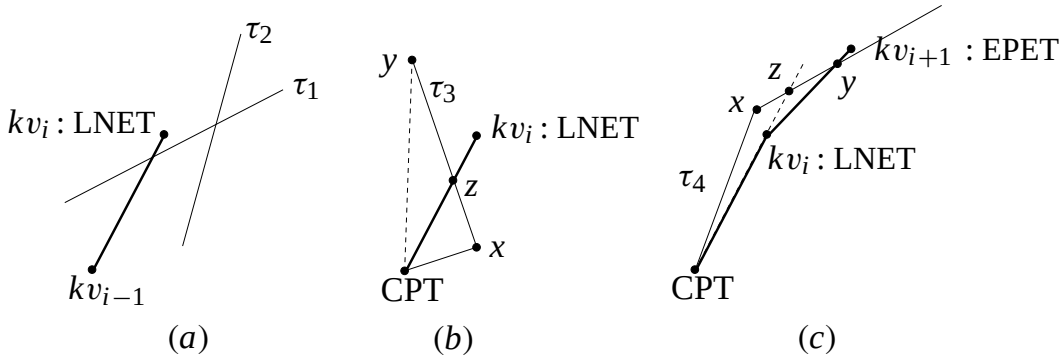

Figure 2.17: Different paths τ which do not contain any intermediate key vertex

optimal path is "CPT-kv-LE". So we only need to consider the cases when there are two or more intermediate kvs located between CPT and LE. In the following, we prove that CPT-kv_1-kv_2-LE is the optimal path for two intermediate kvs case. For the cases of more than two kvs, it can be proved similarly.

First, for a P with two intermediate kvs, if one kv, e. g. kv_1, is on the shortest path, then the other one, e. g. kv_2, must also be on it and the path "CPT-kv_1-kv_2-LE" is the shortest path. Otherwise according to Lemma 2, the subpath, e. g. from kv_1 to LE, can always be shortened by "kv_1-kv_2-LE", which is the shortest path of P_2, i. e. the upper half subpolygon of P.

Second, suppose the overall shortest path is τ which does not contain any intermediate kv, i. e. the corners of τ are not intermediate kvs. Note that a kv is either L (LNET) or E (EPET) type, the possible combinations of two sequential intermediate kvs could be L-L, E-E, L-E or E-L. For an intermediate kv_i, τ does not pass it at its right side when kv_i is an LNET or at its left side when kv_i is an EPET. This is clearly shown in Fig. 2.17 when kv_i is an LNET (for an EPET kv_i, the situations are similar): τ_1 and τ_2 are impossible to be the optimal path in P since kv_i is the most right vertex of P at this level. For the two-intermediate kv case, let CPT $= kv_{i-1}$, it is for sure that a path τ_3 (see (b) in Fig. 2.17) which crosses the line segment "CPT-kv_i" and then passes kv_i at its left side, if it exists, cannot be the optimal path since the part "CPT-x-z-y" at least can be shortened by "CPT-z-y" (even by "CPT-y" if it is inside P). Here x and y are non-kv vertices of P, z is an intersecting point of τ_3 and "CPT-kv_i". Thus if kv_i is an LNET, to be a shortest path, τ needs to pass the line segment "CPT-kv_i" at its left side and does not cross it. An example is τ_4 shown in (c) of Fig. 2.17. Here, x is the lowest "above-kv_i" corner of τ_4. Under kv_i, for τ_4, there could be many corners, but the best possible case is that the path from CPT to the vertex x is a straight line segment.

If the next key vertex kv_{i+1} is also an LNET, τ is always at the left of path "CPT-kv_i-kv_{i+1}-LE" which is a concave inward path. Therefore, τ cannot be the shortest path. It can always be shortened by the concave inward path. If kv_{i+1} is an EPET, since it is impossible to pass an EPET at its left side, the best possible path looks like τ_4, z and y are the intersecting points of τ_4 and the line passing through CPT and kv_i, τ_4 and the line segment "kv_i-kv_{i+1}" respectively. Clearly, such a path τ cannot be the shortest one. The part "CPT-x-z-y" can be shortened by the path "CPT-kv_i-z-y" and furthermore by "CPT-kv_i-y". This contradicts an overall shortest path assumption of τ. Therefore, the overall shortest path does contain intermediate kv. Considering the first result (i. e. if one of the intermediate kv is on the shortest path, the other kv is also on it), it can be concluded that for P with two intermediate kvs, the intuitive algorithm gives the overall shortest path.

Now consider P with more than two intermediate kvs. According to the algorithm, each kv_i is derived from a similar procedure, except that the source vertex of each search procedure is not always CPT, the source of P, but kv_{i-1}. Let LE $= kv_{i+2}$, $(i > 1)$, directly applying the above shortest path solution (for a simple polygon with two intermediate key vertices) leads to the result that the shortest path π_{i-1} from kv_{i-1} to LE is "kv_{i-1}-kv_i-kv_{i+1}-LE". This shortest path result for a polygon with two intermediate key vertices can be also applied to other subpolygons of P. For example, with kv_{i-2} and kv_{i+1} as the source vertex and the target vertex, respectively, the shortest path from kv_{i-2} to kv_{i+1} is "kv_{i-2}-kv_{i-1}-kv_i-kv_{i+1}". Now, consider the shortest path π_{i-2} from kv_{i-2} to LE. We know that if kv_{i-1} is on π_{i-2}, then π_{i-1} is a part of π_{i-2}. Suppose π_{i-2} is τ which does not contain kv_{i-1}, thus τ does not contain kv_i either, since kv_{i-1} is the corner of the shortest path from kv_{i-2} to kv_i. Similarly, kv_{i+1} is not on τ either. In a word, if τ does not contain the first intermediate key vertex kv_{i-1}, it does not contain any intermediate key vertices.

Repeat the analysis as shown in Fig. 2.17. Suppose the shortest path is τ whose corners are not key vertices, note that this time there are three sequential intermediate kvs (i. e. kv_{i-1}, kv_i, kv_{i+1}), the possible combinations are L-L-L, L-E-L, L-E-E, L-L-E, E-E-E, E-L-E, E-L-L, E-E-L (Here we again only discuss the cases where kv_{i-1} is an LNET. EPET cases can be easily discussed using the same idea). In the cases of L-E-E and L-E-L, since there is only one LNET key vertex under the first EPET key vertex, the situations are the same as (c) of Fig. 2.17. The case of L-L-L is also the same as the L-L case which has been discussed before. The new combination case is L-L-E, or generally speaking, there is more than one LNET key vertex under the first EPET key vertex. Fig. 2.18 shows the case of an L-L-E combination. Actually, from the source vertex, no matter how many

LNET type intermediate key vertices there are under the first EPET key vertex, the best possible path looks like τ_5. Its first corner is below the first EPET key vertex but above the LNET key vertex which is located right below that EPET. As shown in Fig. 2.18, τ_5 can be shortened by "kv_{i-2}-kv_{i-1}-kvi-y". This contradicts the shortest assumption of τ. So the shortest path π_{i-2} contains kv_{i-1} and therefore it is "kv_{i-2}-kv_{i-1}-kv_i-kv_{i+1}-LE".

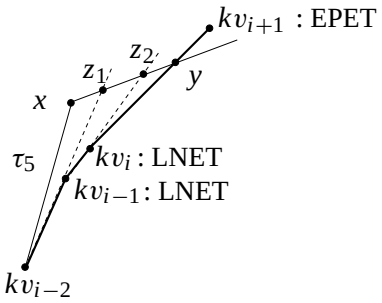

Figure 2.18: L-L-E combination of key vertices and a τ with non-kv corners

If CPT is still below the current source vertex kv_{i-2}, continuously repeat the above steps with a new source vertex which is the nearest key vertex below the current one until the new source vertex $kv_{i-j}(j > 2)$ is CPT. The whole procedure shows that in the simple polygon P the path sequentially connecting all key vertices is the shortest path from CPT to LE. □

For applying the intuitive algorithm to a simple polygon P composed by an EPET-LNET corridor, the worst cases are either its right or left border is concave inward. For such a polygon P with n vertices, there are all together $1 + (n/2)$ events. Then for these worst cases, the running time of the algorithm is

$$(1 + \frac{n}{2} - 1) + (1 + \frac{n}{2} - 2) + \cdots + 2 + 1 = \frac{n^2}{8} + \frac{n}{4}.$$

Now we consider possible improvements. As stated in the proof of Lemma 3, it is clear that if the right border of P is concave inward, the optimal path from CPT to LE is the exact right border. There is then no need to repeat Step 2 and 3 to find intermediate kv_is. All LNETs located above CPT but below LE are the corresponding kv_is. Similarly, the left border is the optimal path if it is concave inward. Furthermore, the search direction of key vertices does not influence the solution, i. e. , we can search from CPT to LE, or we can search from LE to CPT as long as there is the corresponding change for search direction for tkv. Therefore, to reduce the search space during the search procedure, both search directions can be included in the algorithm, i. e. not only nkv but also lkv keeps being updated during the procedure.

The final intuitive algorithm is the following:

Step 1 Initialisation. Set new key vertex nkv=CPT, last key vertex lkv=LE. Store nkv in the set sp which contains all key vertices found so far (direction: forward). Similarly, store lkv in the set $sp1$ (direction: backward).

Step 2 Starting from nkv, check the straight line segment "nkv-lkv".

1. If the straight line segment is contained in P, store all vertices of $sp1$ into sp to complete the shortest path and finish searching.
2. Otherwise, if the straight line segment "nkv-lkv" is completely outside P (except nkv and lkv) and for the subpolygon P_{nl} [3], if its border, which is the one closer to "nkv-lkv", is concave inward-P_{nl}, store sequentially all vertices (except nkv and lkv) on the border and all vertices of $sp1$ into sp to complete the shortest path, finish searching.
3. Otherwise, along the straight line segment "nkv-lkv", find a temporal key vertex tkv:
   $\rightarrow$ Along "nkv-lkv", search from lkv to nkv, identify the point α where this line first leaves the polygon P.
   $\rightarrow$ **Below** α, identify the pair of same-level vertices v_α which is the closest pair to α.
   $\rightarrow$ For two v_α vertices, set the one which is closer to line 'nkv-lkv" as tkv.

Step 3 Update tkv. Check the straight line segment **"nkv-tkv"**. If the straight line segment leaves P, similar to Step 2.3, search from tkv to nkv along the straight line segment "nkv-tkv" and replace tkv with the new one. Repeat Step 3 until finding a tkv so that the straight line segment "nkv-tkv" does not leave P. This tkv is a key vertex kv . Finally replace nkv with the latest tkv and store it into sp.

Step 4 Repeat Step 2 but in opposite direction: Starting from lkv, check the straight line segment "nkv-lkv".

1. Same as Step 2.1: If the straight line segment is contained in P, store all vertices of $sp1$ into sp to complete the shortest path and finish searching.

[3] In subpolygon P_{nl}, the lowest vertex nkv is the source vertex of P_{nl}, the highest vertex lkv is the target vertex of P_{nl}, all the vertices and corresponding edges located above nkv but below lkv are also vertices and edges of P_{nl}.

2. Same as Step 2.2: Otherwise, if the straight line segment "nkv-lkv" is completely outside P (except nkv and lkv) and for the subpolygon P_{nl} , if its border, which is the one closer to "nkv-lkv", is concave inward-P_{nl}, store sequentially all vertices (except nkv and lkv) on the border and all vertices of $sp1$ into sp to complete the shortest path, finish searching.

3. Otherwise, along the straight line segment "nkv-lkv", find a temporal key vertex tkv:
   → Along "nkv-lkv", search from nkv to lkv, identify the point α where this line first leaves the polygon P.
   → **Above** α, identify the pair of same-level vertices v_α which is the closest pair to α.
   → For two v_α vertices, set the one which is closer to line 'nkv-lkv" as tkv.

Step 5 Repeat Step 3 but in opposite direction: Update tkv. Check the straight line segment **"lkv-tkv"**. If the straight line segment leaves P, similar to Step 4.3, search from tkv to lkv along the straight line segment "lkv-tkv" and replace tkv with the new one. Repeat Step 5 until finding a tkv so that the straight line segment "lkv-tkv" does not leave P. This tkv is a key vertex kv . Finally replace lkv with the latest tkv and store it into $sp1$.

Step 6 Repeat Step 2-5.

By applying the proposed intuitive algorithm on the lower level, we can obtain the global optimum very quickly. More importantly, since for online operation, we only need the current velocity, i. e. it is not necessary to find all key vertices. The key vertex which is located right above CPT is sufficient to determine the velocity. In this case, the running time of the algorithm to determine the current velocity is $O(n)$.

2.5 Example

The effectiveness of the hierarchical control architecture proposed in the previous sections is illustrated by a small rail traffic example (Fig. 2.19) involving 3 trains and 3 tracks. Initially, train 1, 2 and 3 are located at the end points of the 3 tracks, i. e. in points A, B and C, respectively. The trains move along the tracks in the directions shown in Fig. 2.19. In the middle of both track AO and track CO, double-line track segments are available,

which makes it possible that two trains pass between points N_1 and M_1, and between points N_2 and M_2, respectively. As in any real world application, unexpected events (e. g. blocking of the track, mechanic failure) may occur and have to be handled by the controller.

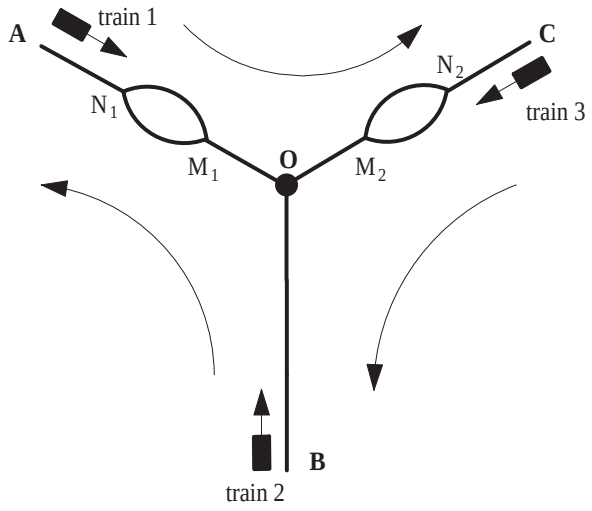

Figure 2.19: A simple rail traffic network

2.5.1 Feasible plans

First, a set of feasible plans is computed offline. Single-line track segment AN_1 can either be occupied by train 1 or train 2. Nevertheless, the order is predetermined, as the track is already occupied by train 1 in the beginning. The same situation occurs for segments BO and CN_2. Therefore, although there are five single-line track segments, the order can only be chosen for segments OM_1 and OM_2. The proposed approach yields a total set of 3 feasible plans pictured in Fig. 2.20, Fig. 2.21 and Fig. 2.22. In plan 1, train 1 occupies track segment OM_1 just after train 2 leaves it. Similarly, train 1 enters track segment OM_2 after train 3 leaves it. In plan 2 the sequence of trains is reversed in both segments. Plan 3 is similar to plan 1 except of the trains' order on OM_2. Fig. 2.23 shows the directed graph for plan 1 including the control arcs needed to enforce this plan.

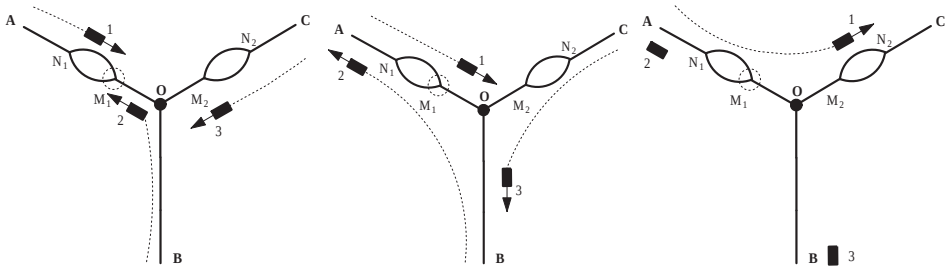

Figure 2.20: Plan 1 (OM_1: train2 train1, OM_2: train3 train1)

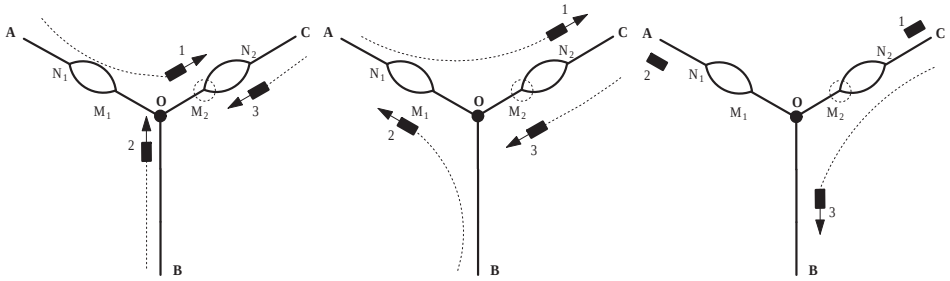

Figure 2.21: Plan 2 (OM_1: train1 train2, OM_2: train1 train3)

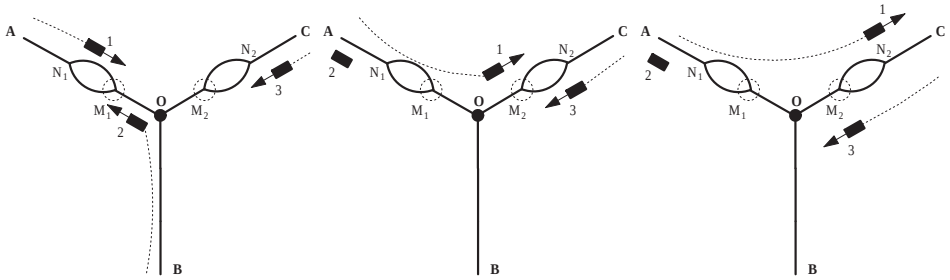

Figure 2.22: Plan 3 (OM_1: train2 train1, OM_2: train1 train3)

2.5.2 Closed loop simulation results

Max-plus simulation of the system under plans 1, 2, and 3 reveals that plan 1 is the time optimal plan if there is no unexpected event. Fig. 2.24 shows the movements of the trains under the proposed hierarchical control scheme. In detail, train 3 moves a little bit slower than train 2 before train 2 empties the track segment BO. Train 1 moves much slower than the other two trains at the beginning since it is expected to pass train 2 on track segment N_1M_1. In case of a disturbance (here train 2 is blocked on track BO at time 2), the supervisory level checks if it is necessary to change the plan. Such a situation is illustrated in Fig. 2.25. After train 2 has been blocked for some time, it is optimal to change from plan 1 to plan 2. This is determined by the supervisory block on the upper

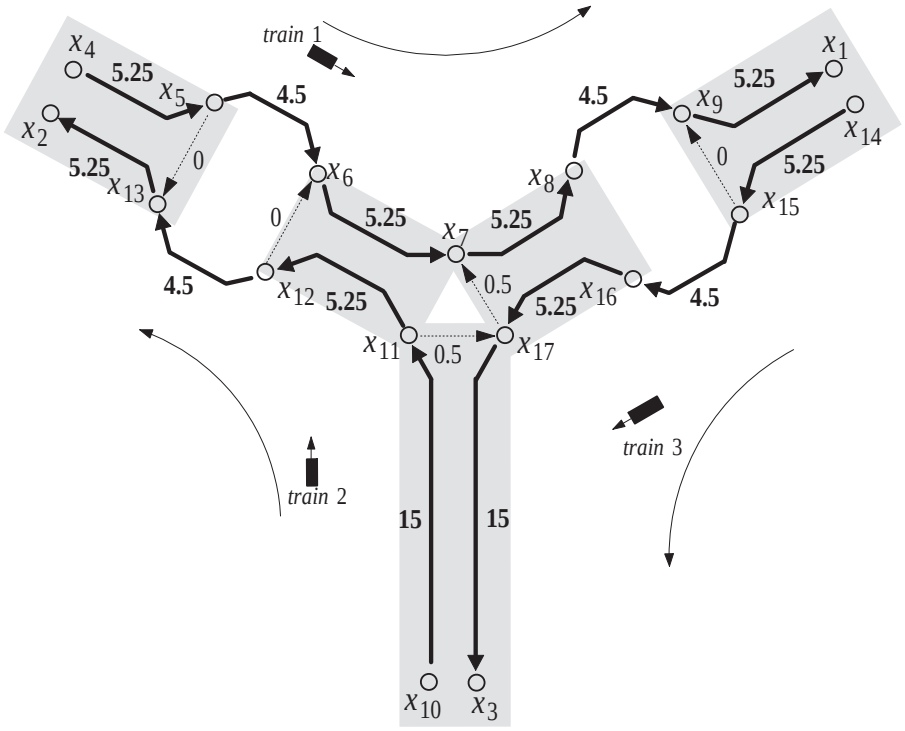

Figure 2.23: Directed graph for plan 1

level and implemented by the implementation block on the lower level. This changes the operation sequence of trains. Now, train 3 occupies track segment OM_2 after train 1 leaves it, train 2 enters track segment OM_1 after train 1 empties it. If the blocking is removed at time 12, this control policy will ensure that all trains arrive at final destinations at time 42.4. For comparison, Fig. 2.26 gives the result when there is no plan change. This implies that the original plan is maintained during the blocking and after it has been removed. Clearly it takes longer for the trains to finish their travel, as the last train arrives at 50.2. In Fig. 2.25 and Fig. 2.26, no matter if there is a plan change after blocking happens, the synchronisation of trains is always ensured by the max-plus model and the min-plus backward calculation. Note that in the case of blocking, the continuous adjustment of velocity signals leads to non-straight trajectories in Fig. 2.25 and Fig. 2.26.

2.6 Conclusions

A novel hierarchical control architecture for a class of discrete-event systems without cyclic features has been proposed in this chapter. It has been applied to a simple rail

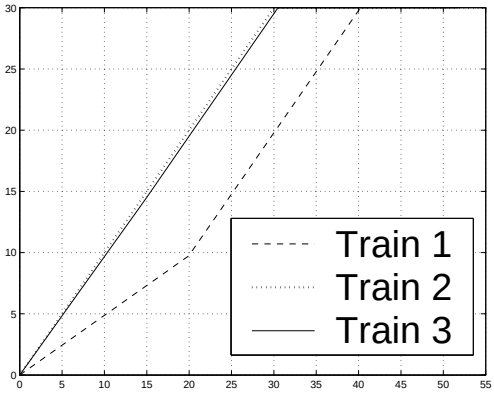

Figure 2.24: Simulation result under normal situation

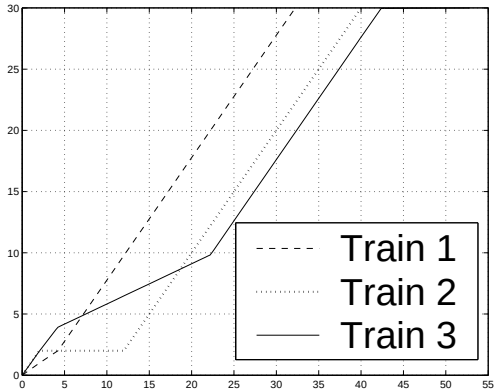

Figure 2.25: Change of plans due to unexpected events

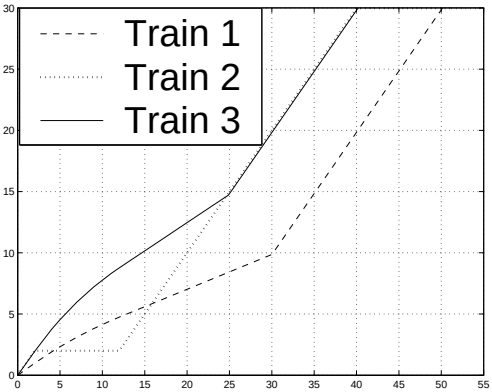

Figure 2.26: Comparison study: No change of plans

traffic network. Based on a max-plus algebra model, the sequence of resource allocations is chosen such that minimum overall time is achieved. An upper level supervisory block ensures the optimal sequence of train movements even under disruptive conditions. A lower level implementation block provides reference velocity signals for each train. By exploiting the remaining degrees of freedom, overall energy consumption is reduced. The implemented policy is generated by use of the dual min-plus algebra model. A new intuitive algorithm speeds up computations and gives the globally optimal reference velocity signals.

The method has been illustrated by means of a small rail traffic example. Simulation results show the effectiveness of the approach.

With respect to the extension of the method to large systems as well as for similar discrete-event systems in other fields, such as flexible manufacturing or chemical process control, two aspects have not been treated in this chapter. The first is to find methods to cope

with combinatoric explosion during the computation of the set of feasible plans for large systems; the second is the extension of the method to cyclically repeated processes where the advantages of the max-plus scheme can be further exploited. Both of them will be the focus of the next chapters.

Chapter 3

Hierarchical control structure for cyclic DES with a conservative condition

In Chapter 2, we proposed a general hierarchical two-level control architecture but discussed each level in detail only for a class of discrete event systems without cyclically repeated features (i. e. for systems "running only one cycle"). In practice, many DES applications in areas like transportation, flexible manufacturing and chemical batch processing exhibit cyclically repeated features. In such cases, cooperation and competition issues among different cycles have to be addressed, which is beyond the scope of Chapter 2. The corresponding control problem is harder than for the noncyclic case, especially under disruptive conditions.

Section 3.1 proposes a conservative assumption for the DES discussed, Section 3.2 describes the max-plus models for the upper level with cyclically repeated features. Section 3.3 discusses in detail the optimisation issues on the upper level. This section also explains how the optimal plan is then chosen efficiently in an online procedure. Section 3.4 elaborates how the plan generated at the upper level can be implemented on the lower level in the more general setting of cyclical behaviour. A case study is provided in Section 3.5 to illustrate the effectiveness of the proposed approach. Finally, a conclusion is given in Section 3.6.

3.1 Conservative condition

In Section 2.2.1, there are two kinds of arcs: travelling arcs and control arcs. The former represent an integral property of the system while the latter are related to shared resources and differ for different plans. For system with cyclicly repeated behaviour, several cycles may exist concurrently. Therefore, control conditions have to be extended such that, e. g., trains from different cycles will not occupy the same track segment simultaneously. Hence, additional control arcs have to be introduced. Also, additional travelling arcs are needed to represent the passage of trains from one cycle into the subsequent cycle.

In Chapter 2, the events connected by either travelling arcs or control arcs are the events from the same cycle. For cyclic DES, arcs may connect events from different cycles. An arc that connects events related to the same cycle is a "zero order arc", an arc that connects events related to two sequential cycles is a "first order arc". In general, there can also be "second order arcs" and so on. To keep our example reasonably simple, we introduce the following conservative condition.

Conservative condition: For each track segment used in several cycles, trains in a latter cycle may not occupy it unless all trains in the previous cycle have left it.

With such a condition, there is no need to introduce arcs with order higher than 1. Besides the zero order arcs, for cyclicly running DES, there are first order control arcs and first order travelling arcs. The former correspond to the conservative condition imposed above while the latter correspond to the transition of trains from one cycle into the subsequent one.

In Chapter 2, for the simple rail traffic network in Fig. 2.19, all trains immediately stopped after they arrived at their destinations (i. e. after one cycle). Now, however, each train will start a new cycle, e. g. a given number of time units after train 1 arrived at C, it will start the next route to go to B, which represents a new cycle. Fig. 3.1 shows the different kinds of control and travelling arcs for this network.

Note that "first order control arcs" ensure the safe resource sharing between cycles and "first order travelling arcs" ensure that each train starts its new journey (cycle $k + 1$, $k \geq 1$) a specified number of time units after its arrival at its destination in cycle k. Although there are optional control arcs of zero order, all first order control arcs are fixed according to the proposed conservative condition.

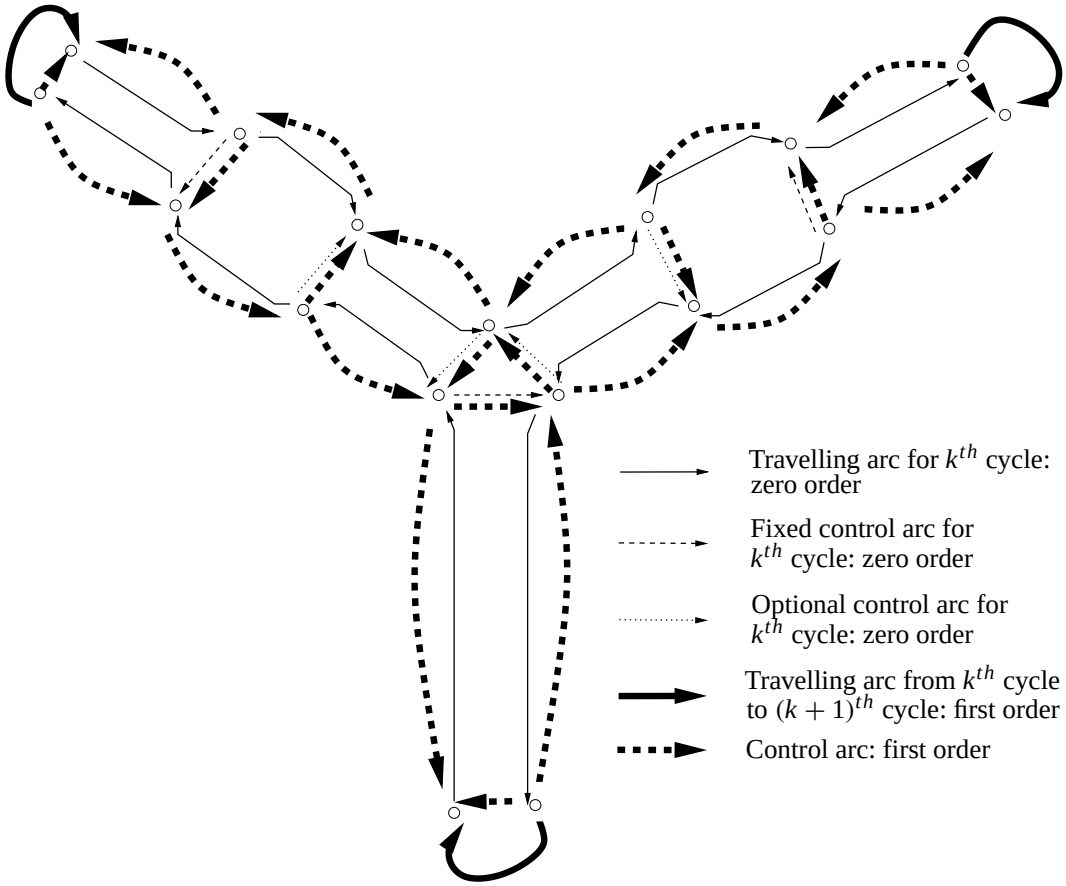

Figure 3.1: Control arcs and travelling arcs for Fig. 2.19.

3.2 Max-plus model

With our conservative assumption in place, for systems with cyclicly repeated behaviour, in addition to the zero order matrix A_0, we now have a first order matrix A_1 corresponding to first order arcs. For cycle k, the implicit max-plus algebra model is:

$$\underline{X}(k) = A_0(k)\underline{X}(k) \oplus A_1\underline{X}(k-1) \oplus Bu, \tag{3.1}$$

$$Y(k) = C \otimes \underline{X}(k). \tag{3.2}$$

As A_0, the matrix A_1 is the max-plus sum of two matrices A_{11} and A_{12}, i. e.

$$A_1 = A_{11} \oplus A_{12}, \tag{3.3}$$

where the elements of A_{11} correspond to "first order travelling arcs" representing the time needed for a train to start a new cycle after finishing the previous cycle. The elements of A_{12} correspond to the control arcs for trains belonging to two subsequent cycles. Unlike

A_{02}, A_{12} cannot represent different choices. Therefore, A_1 is a constant matrix while A_0 will depend on the specific plan to be implemented in the $k\text{-}th$ cycle and may be varying with k.

Repeated insertion of (3.1) into itself provides:

$$\begin{aligned}
\underline{X}(k) &= A_0(k) \otimes \underline{X}(k) \oplus A_1 \otimes \underline{X}(k-1) \oplus B \otimes u \\
&= A_0(k) \otimes [A_0(k) \otimes \underline{X}(k) \oplus A_1 \otimes \underline{X}(k-1) \\
&\quad \oplus B \otimes u] \oplus A_1 \otimes \underline{X}(k-1) \oplus B \otimes u \\
&= A_0^2(k) \otimes \underline{X}(k) \oplus [A_0(k) \oplus I] \otimes A_1 \\
&\quad \otimes \underline{X}(k-1) \oplus [A_0(k) \oplus I] \otimes B \otimes u \\
&\vdots \\
&= A_0^n(k) \otimes \underline{X}(k) \oplus [A_0^{n-1}(k) \oplus \cdots \oplus A_0(k) \\
&\quad \oplus I] \otimes A_1 \otimes \underline{X}(k-1) \oplus [A_0^{n-1}(k) \oplus \cdots \\
&\quad \oplus A_0(k) \oplus I] \otimes B \otimes u. \qquad (3.4)
\end{aligned}$$

As explained in Chapter 2, for physically meaningful systems, $A_0^n(k)$ will only contain ϵ elements. The last identity then represents an explicit max-plus algebra model:

$$\begin{aligned}
\underline{X}(k) &= A_0^*(k) \otimes A_1 \otimes \underline{X}(k-1) \oplus A_0^*(k) \otimes B \otimes u, \qquad (3.5) \\
Y(k) &= C \otimes \underline{X}(k), \qquad (3.6)
\end{aligned}$$

where

$$A_0^*(k) = [A_0^{n-1}(k) \oplus \cdots \oplus A_0(k) \oplus I]. \qquad (3.7)$$

As in Chapter 2, assume $B \otimes u = I \otimes \underline{X}\_in(k)$, (3.5) and (3.6) become

$$\begin{aligned}
\underline{X}(k) &= A(k) \otimes \underline{X}(k-1) \oplus A_0^*(k) \otimes \underline{X}_in(k), \qquad (3.8) \\
Y(k) &= C \otimes \underline{X}(k), \qquad (3.9)
\end{aligned}$$

where $A(k)$ depends on $A_0(k)$:

$$A(k) = A_0^*(k) \otimes A_1. \qquad (3.10)$$

For a given system with cyclicly repeated behaviour, the additional matrix A_1 is fixed. A_{02} differs corresponding to different plans, and, (2.16) still can be used to eliminate infeasible plans and the corresponding max-plus models.

3.3 Optimisation on the upper level

After all feasible plans have been generated, the supervisory level may simulate feasible max-plus-models (3.8), (3.9) over the required total number of cycles to determine the optimal sequence $A(k), k = 1, 2, \cdots$ corresponding to an optimal sequence of plans or a plan list, for short. This is initially done in an offline fashion, i. e. before the system is started. For rail track systems, a typical offline objective function is

$$J_{offline} = \min_{\substack{\text{all feasible} \\ \text{plan lists}}} \Big(\bigoplus_i Y_i(k_l) \Big), \tag{3.11}$$

where k_l is the required total number of cycles and $Y_i(k_l)$ is the arrival time of train i in the k_l-th cycle. With $\oplus$ representing max, the physical meaning is to choose the plan list which provides the minimum final arrival time, i. e. the minimal arrival time of the last train in the last cycle. When performing offline computation, $\underline{X}\_in(1)$ only carries the starting event time. Furthermore, $(\underline{X})_i(0)$ is set to ϵ. The resulting optimal plan list is then implemented via the lower control level.

Max-plus simulation in this step is obviously based on the assumption that there is no unexpected event. Thus if any unexpected incident happens, it may affect the travelling time of trains either directly by blocking them or indirectly via synchronisation with other trains. If this happens, the runtime event time will not match the event time calculated by a-priori model simulation any more. To address such difficulties, an online simulation is integrated to update the aforementioned optimal plan list.

At time t_k, for each of the concurrently existing cycles l (e. g. $l = k, k+1$), $[\underline{X}(l)](t_k)$, $[Y(l)](t_k)$ will be updated by using (3.8), (3.9) with $[\underline{X}\_in(l)](t_k)$ provided by the C/D block. Here, $\underline{X}(l)$ is $\underline{X}$ vector in cycle l, by $[\underline{X}(l)](t_k)$, it is the vector $\underline{X}(l)$ at time t_k.

The online objective may be the same as the offline objective (3.11), i. e.

$$J_{online1} = J_{offline}. \tag{3.12}$$

It may also be different from the offline objective, for example, it may state that after unexpected delays all trains have to catch up with the timetable corresponding to the original offline plan list as fast as possible. In this case,

$$J_{online2} = \min_{\substack{\text{all feasible} \\ \text{plan lists}}} K_{\triangle}, \tag{3.13}$$

where $K_\triangle$ is the index of the earliest cycle so that all delayed trains can start (or end) their cycle trips according to the timetable, i. e.

$$\begin{cases} \forall i, \triangle_i(l) = 0. & l \geq K_\triangle \\ \exists i, \triangle_i(l) > 0. & l < K_\triangle, \end{cases} \tag{3.14}$$

$$\triangle_i(l) = t_{start_i}(l) - timetable_i(l), \tag{3.15}$$

where $t_{start_i}(l)$ is the actual starting time of train i in cycle l and $timetable_i(l)$ is the starting time according to the timetable of train i in cycle l.

In real world applications, it is often preferable that a system exhibits a *strictly cyclic* pattern, i. e. in each cycle the same plan is implemented. Among all feasible plans, to maximise the throughput of the system, the plan which has the minimal eigenvalue λ is chosen. Note that this will in general not coincide with the optimal plan list for a predefined cycle number k_l.

With a strictly cyclic pattern, starting-times in each cycle tend to be regular, i. e. after enough cycles:

$$t_{start_i}(l+1) = t_{start_i}(l) + \lambda.$$

Consider the eigenspace problem in max-plus algebra, since

$$A \otimes v_e = \lambda \otimes v_e,$$

if the system starts with $\underline{X}(0) = v_e$, then $\underline{X}(1) = A \otimes \underline{X}(0) = \lambda \otimes v_e$, and in general $\underline{X}(k) = \lambda^k \otimes v_e$. Thus, for strictly cyclic systems, a timetable is determined by the corresponding eigenvector beforehand. Note that in practice a timetable will always be chosen to include a safety margin T_{sm}. Otherwise, an unexpected delay will never be recovered as changes of plans are not allowed within a strictly cyclic pattern.

Hence, for strictly cyclic systems, the cycle time T_{ct} is

$$T_{ct} = timetable_i(l+1) - timetable_i(l) = \lambda + T_{sm}. \tag{3.16}$$

If a delay lasts for t_{delay} time units and always the case for strictly cyclic system, the delay can be recovered in N_{re} cycles, where

$$N_{re} = \lceil \frac{t_{delay}}{T_{sm}} \rceil, \tag{3.17}$$

and $\lceil \rceil$ denotes the closest integer which is greater or equal to $\frac{t_{delay}}{T_{sm}}$.

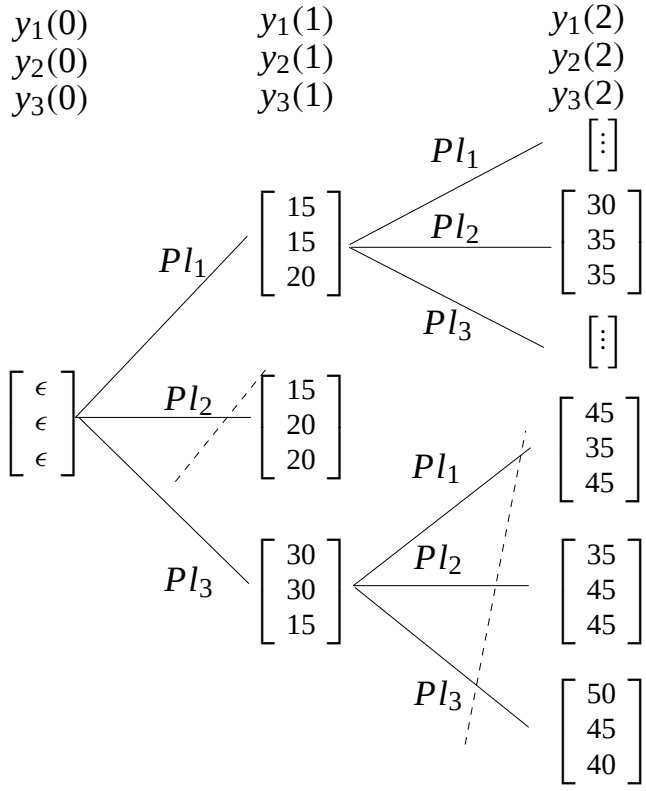

Figure 3.2: Reducing the search space

Online optimisation with objective (3.13) can speed up the recovery by temporarily changing to other suitable plans. After delay recovery, the system may go back to the original strictly cyclic pattern at cycle $K_{\triangle}$.

Besides the shrinking of the feasible plan set as described in Section 2.2.3, during online optimisation the search space can be reduced further by several other steps to enhance optimisation efficiency.

First, if the online objective is $J_{online2}$ and if $N_{re} < k_l - k$, instead of covering the predefined total number of cycles k_l, the search range only covers N_{re} cycles. For example, if the number of elements of the feasible plan set $F(j)$ in cycle j is $M(j)$, the search space is reduced from $\prod_{j=k}^{k_l} M(j)$ to $\prod_{j=k}^{N_{re}+k} M(j)$. Second, for the objective $J_{online2}$, if $\exists k^*$, $k \leq k^* < N_{re} + k$ (k is the index of the cycle at which the unexpected event happens), such that there exists a plan list which recovers the timetable in cycle k^*, the search space may be reduced to $\prod_{j=k}^{k^*} M(j)$. Third, for a designated objective $J_{online1}$ or $J_{online2}$, by comparing with $Y_i(j)$ for either every feasible plan (if $j = k$) or plan list (if $j > k$) after simulation for cycle j, the search space can be reduced further by deleting some plans or

plan lists according to the following rules:

$$Y_i^{(Pl_1)}(j) \geq Y_i^{(Pl_2)}(j), \forall i \implies \text{discard } Pl_1, \tag{3.18}$$

$$Y_i^{(Pl_1,Pl_f)}(j) \geq Y_i^{(Pl_2,Pl_h)}(j), \forall i, \text{for a specific } h \text{ and for any } f \in F(j)$$
$$\implies \text{discard } (Pl_1, \cdots). \tag{3.19}$$

Pl_i represents plan i. For example, as shown in Fig. 3.2, $\forall i$, $Y_i^{(Pl_2)}(1) \geq Y_i^{(Pl_1)}(1)$, hence we can discard Pl_2. Since $y_3^{(Pl_3)}(1) < y_3^{(Pl_1)}(1)$, it is still necessary to continue searching with Pl_3 as a possible option in the first cycle. In cycle 2, (3.19) is used to delete all plan lists starting with Pl_3.

In brief, the supervisory level involves both offline and online tasks. The offline process is used to find both the set of all feasible plans based on the system topology, and an offline optimal plan list which can be interpreted as a timetable. The online part updates both the feasible plan set and the optimal plan list. It also provides the time specification $\underline{X}(k)$ to the lower level.

3.4 Implementation level

It is clear that the output EPET specification from the upper level is not always necessary to follow from an energy saving point of view. With the remaining degrees of freedom, we need to find the LNET for each train as an upper bound for the event times.

Lemma 4 *Given V_1, $V_2 \in \mathbb{R}_{mm}^n$, $G_1, G_2, H_1, H_2 \in \mathbb{R}_{max}^{n\times n}$, and consider the following set of equation in Z_1 and Z_2*

$$V_1 = G_1 Z_1 \oplus H_1 Z_2, \tag{3.20}$$
$$V_2 = G_2 Z_1 \oplus H_2 Z_2. \tag{3.21}$$

The greatest subsolution for (3.20) and (3.21) is given by

$$\overline{Z}_1 = ((-G_1)^T) \otimes' V_1 \oplus' ((-G_2)^T) \otimes' V_2, \tag{3.22}$$
$$\overline{Z}_2 = ((-H_1)^T) \otimes' V_1 \oplus' ((-H_2)^T) \otimes' V_2. \tag{3.23}$$

Proof For all possible Z_1, Z_2 which satisfy (3.20), we have

$$G_1 \otimes Z_1 \leq V_1, \tag{3.24}$$
$$H_1 \otimes Z_2 \leq V_1. \tag{3.25}$$

That is, for (3.24),

$$\bigoplus_{k=1}^{n}((G_1)_{ik} \otimes (Z_1)_k) \leq (V_1)_i, \forall i = 1, 2, \ldots, n.$$

Thus,

$$\begin{aligned}
(G_1)_{ik} \otimes (Z_1)_k &\leq (V_1)_i, \quad \forall k, i = 1, 2, \ldots, n. \\
(Z_1)_k &\leq (V_1)_i - (G_1)_{ik}, \quad \forall k, i = 1, 2, \ldots, n \\
&= (-(G_1)_{ik}) + (V_1)_i, \quad \forall k, i = 1, 2, \ldots, n \\
&= (-(G_1)^T)_{ki} + (V_1)_i, \quad \forall k, i = 1, 2, \ldots, n \\
(Z_1)_k &\leq [(-(G_1)^T) \otimes' V_1]_k, \quad \forall k = 1, 2, \ldots, n.
\end{aligned}$$

The last inequality shows that for all Z_1 satisfying (3.20),

$$Z_1 \leq \overline{Z}_a = (-(G_1)^T) \otimes' V_1.$$

Similarly, for all Z_1 satisfying (3.21),

$$Z_1 \leq \overline{Z}_b = (-(G_2)^T) \otimes' V_2.$$

Thus, for all Z_1 satisfying (3.20) and (3.21),

$$Z_1 \leq \overline{Z}_1 = \overline{Z}_a \oplus' \overline{Z}_b = (-(G_1)^T) \otimes' V_1 \oplus' (-(G_2)^T) \otimes' V_2. \tag{3.26}$$

Similarly, for all Z_2 satisfying (3.20) and (3.21),

$$Z_2 \leq \overline{Z}_2 = (-(H_1)^T) \otimes' V_1 \oplus' (-(H_2)^T) \otimes' V_2. \tag{3.27}$$

On the other hand,

$$\begin{aligned}
[G_1 \otimes \overline{Z}_1]_i &= \bigoplus_{k=1}^{n}(G_1)_{ik} \otimes (\overline{Z}_1)_k, \quad \forall i = 1, 2, \ldots, n \\
&= \bigoplus_{k=1}^{n}(G_1)_{ik} \otimes (\overline{Z}_a \oplus' \overline{Z}_b)_k, \qquad \forall i = 1, 2, \ldots, n \\
&\leq \bigoplus_{k=1}^{n}(G_1)_{ik} \otimes (\overline{Z}_a)_k, \qquad \forall i = 1, 2, \ldots, n \\
&= \bigoplus_{k=1}^{n}(G_1)_{ik} \otimes \Big((-(G_1)^T) \otimes' V_1\Big)_k, \qquad \forall i = 1, 2, \ldots, n \\
&= \bigoplus_{k=1}^{n}(G_1)_{ik} \otimes \Big(\bigoplus_{j=1}^{n}{}'[(-(G_1)^T)_{kj} \otimes' (V_1)_j]\Big), \qquad \forall i = 1, 2, \ldots, n \\
&= \bigoplus_{k=1}^{n}\Big(\bigoplus_{j=1}^{n}{}'[(G_1)_{ik} \otimes ((-(G_1)^T)_{kj} \otimes' (V_1)_j)]\Big), \qquad \forall i = 1, 2, \ldots, n.
\end{aligned}$$

For $i = j$ and $(G_1)_{ik} = -\infty$, we have

$$[(G_1)_{ik} \otimes ((-(G_1)^T)_{kj} \otimes' (V_1)_j)] = -\infty.$$

For $i = j$ and $(G_1)_{ik} \in \mathbb{R}$, we have

$$[(G_1)_{ik} \otimes ((-(G_1)^T)_{kj} \otimes' (V_1)_j)] = (G_1)_{ik} + (-(G_1)^T)_{ki} + (V_1)_i = (V_1)_i.$$

Therefore,

$$\bigoplus_{j=1}^{n}{}'[(G_1)_{ik} + (-(G_1)^T)_{kj} + (V_1)_j] \leq (V_1)_i,$$

and

$$\begin{aligned} [G_1 \otimes \overline{Z}_1]_i &\leq \bigoplus_{k=1}^{n}(V_1)_i, \quad \forall i = 1, 2, \ldots, n \\ &= (V_1)_i, \quad \forall i = 1, 2, \ldots, n. \end{aligned}$$

This result can also be obtained by directly applying the first inequality of (1.10) to the right hand side of

$$G_1 \otimes \overline{Z}_1 \leq G_1 \otimes ((-(G_1)^T) \otimes' V_1).$$

Similarly,

$$\begin{aligned} [H_1 \otimes \overline{Z}_2]_i &\leq (V_1)_i, \quad \forall i = 1, 2, \ldots, n, \\ [G_2 \otimes \overline{Z}_1]_i &\leq (V_2)_i, \quad \forall i = 1, 2, \ldots, n, \\ [H_2 \otimes \overline{Z}_2]_i &\leq (V_2)_i, \quad \forall i = 1, 2, \ldots, n. \end{aligned}$$

This means that $\overline{Z}_1$ and $\overline{Z}_2$ calculated by using (3.22) and (3.23) form a subsolution of (3.20), (3.21). Furthermore, considering (3.26) and (3.27), $\overline{Z}_1$ and $\overline{Z}_2$ are the greatest subsoltion of (3.20), (3.21). □

Note that $\overline{Z}_1$, $\overline{Z}_2$ can be interpreted as vectors of latest necessary event times that satisfy event times specified by V_1 and V_2.

Based on Lemma 4, the following corollary holds:

Corollary 2 *Given V_i and G_{ij} ($V_i \in \mathbb{R}^n_{mm}$, $G_{ij} \in \mathbb{R}^{n\times n}_{max}$, $i = 1, 2, \ldots, l$, $j = 1, 2, \ldots, m$), for Z_j, ($j = 1, 2, \ldots, m$) which satisfy*

$$V_i = \bigoplus_{j=1}^{m} G_{ij} Z_j, \forall i = 1, 2, \ldots, l, \tag{3.28}$$

the latest necessary event times for Z_j ensuring all V_i ($i = 1, 2, \ldots, l$) are

$$\overline{Z}_j = \bigoplus_{i=1}^{l}{}'((-G_{ij})^T \otimes' V_i). \tag{3.29}$$

Using Lemma 4, the LNET of each train for cyclic systems can be easily determined. Recall that for systems with non-cyclicly repeated features, LNET is defined in such a way that the event time of the final event for each train according to EPET will be met. In addition, it is also ensured that LNET does not violate the EPET schedule of other trains. For systems with cyclicly repeated features, we have the additional requirement that LNET should not violate the EPET schedule of the sequential cycle.

Suppose $QS(j)$ is the event index set for all final events of trains in cycle j and $PS_m(j)$ is the index set for all events related to train m in cycle j. According to the max-plus model (3.8) and Lemma 4, at time t_k the LNET specifications $[\overline{X}_m(j)](t_k)$ for train m in cycle j can be calculated as

$$\begin{aligned} [\overline{X}_m(j)](t_k) &= (-(A_0(j)^*)^T) \otimes' [\underline{XR}_m(j)](t_k) \\ &\oplus' \ (-A^T(j+1)) \otimes' \underline{X}(j+1), \end{aligned} \tag{3.30}$$

where

$$[(\underline{XR}_m)_i(j)](t_k) = \begin{cases} [\underline{x}_i(j)](t_k), & i \in Q(j) \text{ or } i \notin PS_m(j) \\ +\infty, & \text{otherwise.} \end{cases} \tag{3.31}$$

Within the corridor specified by EPET $\underline{X}(j)$ and LNET $\overline{X}_m(j)$, the velocity signal can then be optimised locally for train m using the algorithms discussed in Chapter 2.

3.5 Simulation

We still use the small rail traffic example depicted in Fig. 2.19 to illustrate the effectiveness of our hierarchical control architecture for timed DES with cyclicly repeated behaviour.

First, a fixed time-optimal plan list is derived from offline simulations. The system is expected to run according to this list. If unexpected events happen and block some trains, the supervisory level will decide whether and how to change the plan list.

Let the total number of cycles be $k_l = 5$. Fig. 3.3 shows the movements of the trains. The optimal plan list is [1 2 2 1 2], i. e. in cycle 1, plan 1 is implemented, in cycle 2, plan 2, etc.

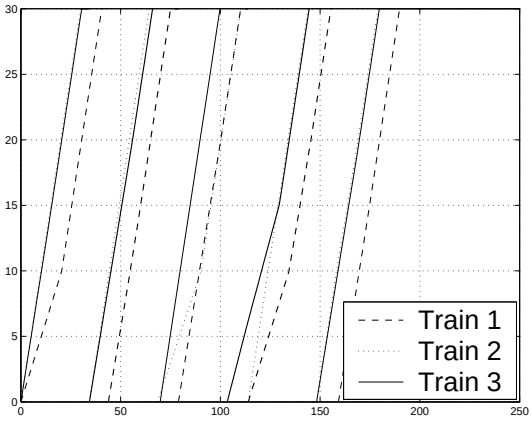

Figure 3.3: Simulation result without disturbances, plan list: [1 2 2 1 2]

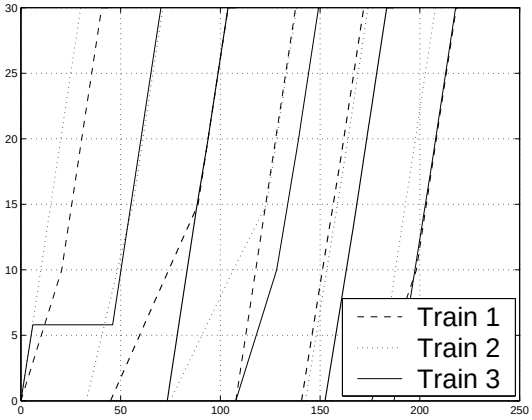

Figure 3.4: Blocking time is known beforehand, new plan list: [3 2 1 2 2]

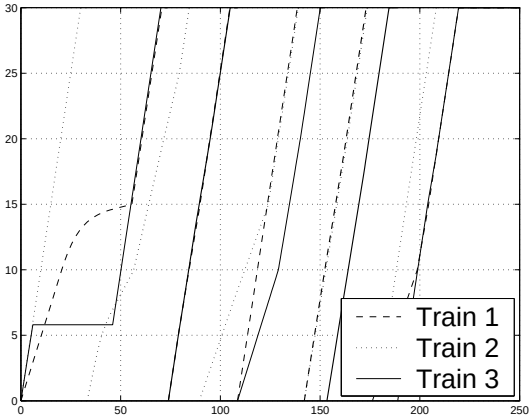

Figure 3.5: Blocking time is unknown beforehand, new plan list: [1 2 1 2 2]

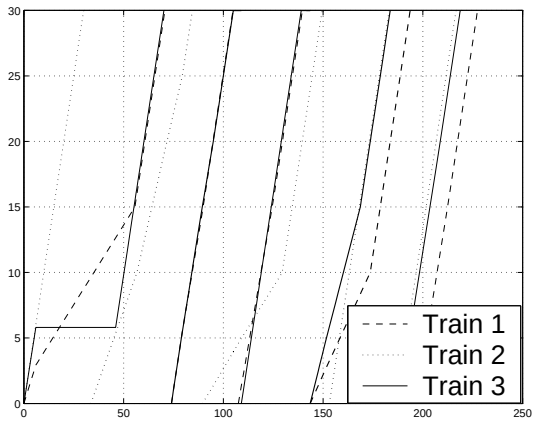

Figure 3.6: Blocking time is known beforehand, stick to original plan list: [1 2 2 1 2]

The last cycle is finished at time 190, if no disturbance occurs. In case of a disturbance (here, train 3 is blocked on track M_2N_2 during the time interval $[6, 46]$), the supervisory level checks if it is necessary to modify the plan list using the objective function $J_{online1}$. This scenario is presented in Fig. 3.4 and Fig. 3.5, respectively. In Fig. 3.4, the blocking time is known a-priori. With this information, the supervisory level changes the plan list immediately in order to reduce the delay time caused by the obstacle on the track. The required 5 cycles are finished in about 219 time units.

If the blocking time is unknown beforehand, the supervisory level keeps the original plan list until the status of the trains (including the blocked train) evolve to such a point where another plan list optimises the online objective $J_{online1}$. In this example, as shown in Fig. 3.5, the supervisor changes the plan at time 8 to [1 2 1 2 2]. In this case, the required 5 cycles still can be finished in 219 time units.

As a comparison, Fig. 3.6 illustrates what happens if the original plan list is kept. In this case, the required 5 cycles are finished in about 228 time units.

Under the same blocking situation, with different online objectives, the supervisory level may change to different plan lists. Suppose, for example, the required number of cycles is 7, and train 3 is blocked on track M_2N_2 during the time interval $[6, 26]$ which is unknown beforehand. If the online cost function $J_{online2}$ is used, the supervisory level changes the plan list from [1 2 2 1 2 2 2] to [1 2 1 2 2 2 2]. And the offline timetable is recovered after the $6th$ cycle. With $J_{online1}$, the plan list is changed to [1 2 1 2 2 1 2], which minimises the required time. However, for this policy, the timetable is not recovered.

3.6 Conclusion

This chapter extends the hierarchical control architecture proposed in Chapter 2 to a class of discrete-event systems with cyclicly repeated features. Based on a max-plus algebra model, an upper level supervisory block ensures the optimal sequence of train movements and the optimal sequence of cycle plans even under disruptive conditions. A lower level implementation block then provides reference velocity signals for each train. By exploiting the remaining degrees of freedom, the resulting control strategy reduces overall energy consumption. Note that the implementation policy is generated by use of the dual min-plus algebra model. Simulation results for a small rail traffic example illustrate the effectiveness of the presented approach. We expect the method to be also useful in other DES applications which exhibit cyclicly repeated features, such as those in flexible manufacturing systems and high throughput screening plants.

Chapter 4

Hierarchical control structure for general cyclic DES

In Chapter 3, we considered the control problem for cyclic systems under a conservative condition. With this condition, the users in a latter cycle may not occupy the shared resources before all the users in the previous cycle release them. Consequently, the resulting max-plus model is rather simple and involves a constant first order system matrix A_1. It is only the zero order system matrix A_0 which is time-variant and embodies different plans. In reality, there is a natural desire to relax this conservative condition in many application fields since it is rather rigid. For example, in a cyclic system, two users TA and TB both want to use the resource RC. If for some reason TB is not ready to occupy RC for a long period of time after TA has released it, it is better to let TA in the next cycle occupy RC first again. Consequently, performance will be improved at the expense of simplicity of the relevant max-plus model. This chapter focuses on control problems for cyclic DESs with less restrictive conditions than in the previous chapter.

This chapter is organised as follows: Section 4.1 introduces "negative order arcs" for more general systems and discusses possible combinations of optional control arcs on a shared resource. Section 4.2 enhances the method for determining feasible plans from Section 2.2.3. Section 4.3 briefly summarises a control method for strictly cyclic systems and discusses remaining problems. Section 4.4 discusses a new problem arising in the considered, more general, class of systems, that is, how to find the feasible plan lists for non-strictly cyclic systems. Section 4.5 solves the problem in terms of updating the feasible plan list set. Section 4.6 studies the upper level max-plus model for the more general system. Section 4.7 discusses the problem of plan list optimisation. Section 4.8

provides the lower level implementation. Section 4.9 shows the simulation results for a train-track system. Finally, a conclusion is given in Section 4.10.

4.1 Negative order arcs

4.1.1 Introduction of negative order arcs

As discussed before, we use travelling arcs to model the relation between event-times of the same physical train within a cycle or when transiting into its next cycle. Control arcs are used to model the relations between event-times of different trains. For systems with cyclicly repeated behaviour, a train starts a new cycle only after it finishes the previous one. Thus zero order and first order arcs are sufficient for the travelling arcs. Also, with the conservative condition imposed in Section 3.1, only zero and first order control arcs are needed.

Now we relax this condition so that trains on different routes competing for a shared resource do not have to wait for each other just because one of the trains is still in a previous cycle. Such a relaxation results in a new requirement for *negative order control arcs*. It represents a situation where a train in a previous cycle may not occupy a single-line track segment before a corresponding train in a later cycle leaves it. Note that identical events in different cycles are ordered by

$$x_i(k) < x_i(k+1). \tag{4.1}$$

With (4.1), on a certain route of a train, the number of the events, or correspondingly, the number of the zero order travelling arcs, limits the number of cycles existing concurrently on the route. In this chapter, we always consider the most complicated situation in which the event number is large enough to admit each train in a different cycle.

Thus for a system having N_t trains, the possible maximum number of cycles existing concurrently is $N_{mc} = N_t$, i.e. each train in a different cycle, successively, taking the same route. In this case, the maximum cycle difference is $N_{mc} - 1 = N_t - 1$. Depending on the specific application, N_{mc} may be a lower number.

To distinguish graphically negative order arcs and positive order arcs, we add small circles to negative order arcs and small slashes to positive order arcs. An arc without a circle or a slash is a zero order arc. Fig. 4.1 shows a single line track segment which is part of some train track system. The first order control arc labelled a_1 represents the fact that the

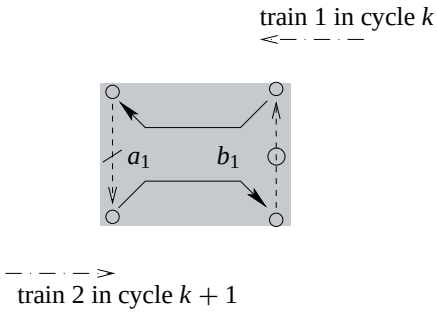

Figure 4.1: $+1$-order arc and -1-order arc

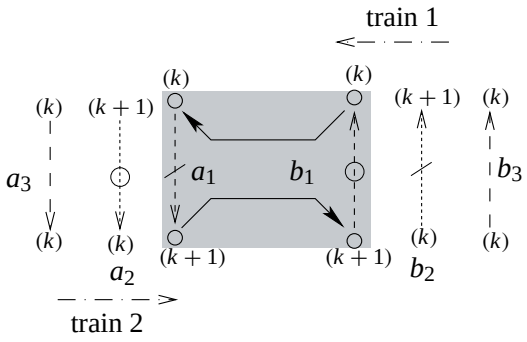

Figure 4.2: Three groups of control arcs

earliest time for train 2 in cycle $(k+1)$ to enter the track segment is a_1 time units after train 1 in cycle k has left it. The -1-order control arc labelled b_1 means the opposite: train 1 in cycle k may not occupy the track before b_1 time units after train 2 in cycle $(k+1)$ has left it. Obviously, only one of these two control arcs can exist in a feasible plan. A 2nd order arc is represented by two slashes etc.

4.1.2 Control arc combinations on a shared resource

Suppose the minimum and maximum order of control arcs for the system in Fig. 4.1 is -1 and 1 respectively. Then, there are three groups of optional control arcs relating train 1 and train 2 on this track segment, as shown in Fig. 4.2. The first group, (a_1, b_1), represents the event sequence between train 1 in a previous cycle k and train 2 in a latter cycle $(k+1)$, while the second group, (a_2, b_2), represents the event sequence between train 1 in a latter cycle $(k+1)$ and train 2 in a previous cycle k. The third group, (a_3, b_3), relates the event times of train 1 and train 2 in the same cycle k. To ensure the safe sharing, in each group, one of the control arc needs to be chosen. In any plan, there are three control arcs coming from these three groups accordingly. To simplify the discussion, *from now on we assume the weights of control arcs are identical.* From (4.1), it is clear that for any set of control arcs pointing from node i to node j, the arc with the lowest order automatically implies the arcs with higher order. For example, for three control arcs labelled a_i $(i = 1, 2, 3)$, the -1-order control arc labelled a_2 means that train 2 in cycle k can only enter the track segment a_2 time units after train 1 in cycle $(k+1)$ has left it. Thus train 2 in cycle k can of course enter the track segment at the earliest a_2 time units after train 1 in cycle k has left it. As the weights of arcs are assumed to be identical, this implies the zero order arc labelled a_3. The same argument applies for the first order arc labelled

a_1. Furthermore, in each group of optional control arcs ($a_i, b_i, i \leq 3$), choosing one arc (say a_i) implies not choosing the other one (say b_i). So in order to describe a possible combination ($a_i, b_j, i, j \leq 3$) of control arcs on a single-line track, only its lowest order arc labelled a_i needs to be considered.

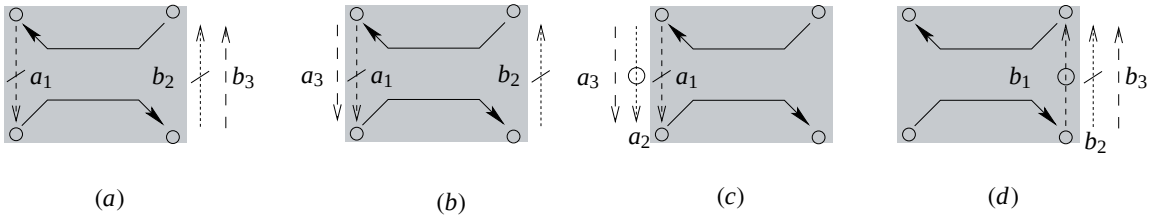

Figure 4.3: Combinations of optional control arcs

For example, in Fig. 4.3 (a), we illustrate the case where the lowest order arc labelled a_i is the first order arc a_1, which means that there are no control arcs a_2 and a_3. Instead, there are control arcs b_2 and b_3. In Fig. 4.3 (b), the lowest order arc labelled a_i is the zero order arc a_3, so this combination contains control arcs a_3, a_1 and b_2. If we choose the -1-order arc a_2 as the lowest order arc labelled a_i (Fig. 4.3 (c)), we need to consider a_1, a_2, a_3. Fig. 4.3 (d) shows the case where we have neither of the arcs labelled a_1, a_2, a_3.

In general, for a system with $N_{mc} = N_t$, let N_o denote the absolute value of the lowest possible order of all control arcs related to any two trains of the system. Considering the group of zero order control arcs, there are $2 \times N_o + 1$ groups of optional control arcs. Therefore, the total number of possible combinations of control arcs is:

$$N_{com} = 2 \times N_o + 1 + 1 = 2 \times (N_o + 1). \tag{4.2}$$

Since higher order control arcs are implied by the corresponding lower order control arcs, the combinations shown in Fig. 4.3 can be as shown in Fig. 4.4. Note that in Fig. 4.4 (c), the absence of a -2nd order arc with an a-label implies the presence of a 2nd order arc with a b-label. This is the lowest order arc with a b-label. Similarly, in Fig. 4.4 (d), the absence of a -2nd control arc with label b implies the presence of a 2nd order arc with label a. It is clear that for a single-line track segment, the differences of plans only lie in the different orders of its related control arcs.

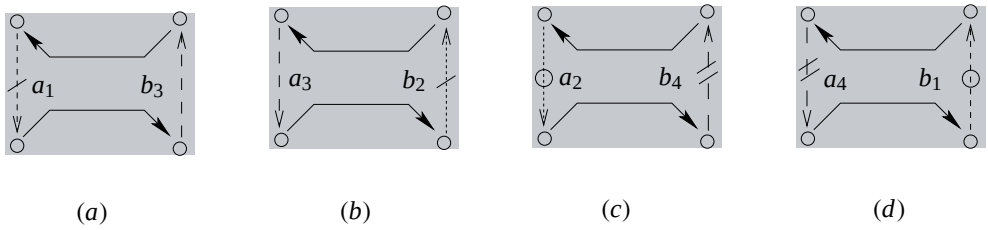

Figure 4.4: Combinations of optional control arcs—simplified

4.2 Feasible plans

For a system with several single-line track segments, different choices of control arcs result in different plans. It is first necessary to identify and eliminate plans that lead to blocking. Such plans are called infeasible. For cyclic systems, there are non-zero order arcs controlling the sequence of events in different cycles. Blocking may happen within a single cycle or between cycles.

In Section 2.2.3, we detected blocking by checking for the existence of circuits in a directed graph and identified infeasible plans by calculating $A_0^k, k \leq n$ (see 2.16). Unfortunately, this method is not appropriate for the more general class of system investigated now. This is due to the introduction of negative order arcs. For example, in Fig. 4.1, we know that there is a blocking if the arcs labelled by a_1 and b_1 are both included in a plan. This will not be reflected in A_0^k, since neither arc is zero order. On the other hand, because of the existence of positive order arcs, not every circuit in the directed graph necessarily corresponds to a blocking. For example, in Fig. 4.4, each combination contains a circuit, but none of them corresponds to an infeasible plan. To explain this more clearly, we investigate an extended graph, where events in different cycles are represented by different nodes. For example, the scenario shown in Fig. 4.4 (a) is now represented by the graph in Fig. 4.5. Fig. 4.6 shows the extended graph related to the scenario of Fig. 4.4 (c). The solid arcs between cycles are the first order travelling arcs.

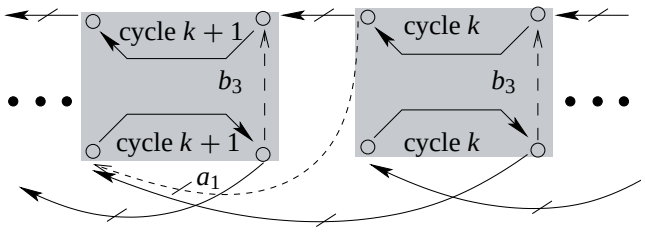

Figure 4.5: Sequence relation for Fig. 4.4 (a)

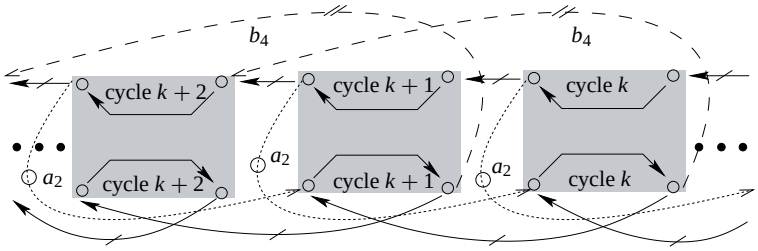

Figure 4.6: Sequence relation for Fig. 4.4 (c)

In the following, we will need to refer both to the extended graph (as in Fig. 4.5 and 4.6) and the underlying condensed graph (as in Fig. 4.4). To find a method for detecting blocking in the considered general class of systems, we introduce the definition of the order of a circuit or a path. We always assume the weight of the circuit or the path is a positive real number.

Definition 6 (Order of a circuit/path) *In a directed graph, the order of a circuit C_j (or a path), denoted as $od(C_j)$, is the sum of the orders of all arcs which form the circuit (or the path).*

For example, the order of the circuit in Fig. 4.4 (a) is 1. Note that in the extended graph in Fig. 4.5, this circuit corresponds to an open path.

Lemma 5 *In the directed graph of a strictly cyclic plan, a circuit C_j causes blocking if and only if the order of the circuit $od(C_j) \leq 0$.*[1]

Proof It is obvious that circuits with nonpositive order in the condensed graph correspond to circuits in the extended graph. Blocking occurs if and only if the extended graph contains circuits.

□

Circuits in the condensed graph will be referred to as "potential circuits". Some of them may correspond to a closed path in the extended graph. These will simply be referred to as "circuits" and will cause blocking. We introduce the notion of a "circuit pattern" by removing the order information from the control arcs in a potential circuit. Hence, different potential circuits may correspond to the same circuit pattern.

In the last section we concluded that the maximum number of nonblocking combinations

[1] To be more accurate, the order of a circuit is 0. The reason that sometimes it is negative is because we do not count the implicitly existing +1-order travelling arcs between cycles.

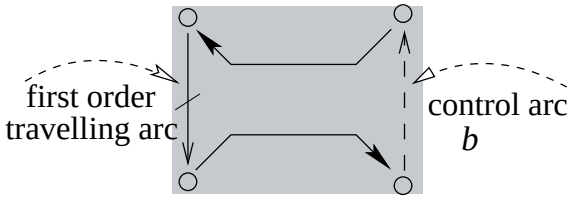

Figure 4.7: An ending / starting single-line track segment

of control arcs on a single-line track segment is $N_{com} = 2 \times (N_o + 1)$. If a single-line track segment is an ending and starting track segment (i. e. for a physical train, the segment is not only the final part of its route in cycle k, but also the starting part of its new route in cycle $k+1$), the number of possible non-blocking combinations is less than $2 \times (N_o + 1)$. There is a first order travelling arc at the ending (or starting) part of the track because a train in the previous cycle will start a new trip in the latter cycle. Thus the lowest order of arcs at this place is less or equal to 1, no matter which order a control arc at this place takes (such a control arc is called a *cycle-changing control arc*). For such a single-line track segment as shown in Fig. 4.7, from Lemma 5, it follows that blocking occurs if b is a negative order arc.

According to Lemma 5, we can find all infeasible plans for the generalised systems. The blockings are caused by control arcs on single-line track segments: either by combining this kind of control arcs for different single-line tracks, or by combining them with other kinds of arcs (such as travelling arcs as well as other control arcs). To check the feasibility of plans, we only need to check the order of circuits of the condensed graph containing any of these control arcs. Note that the condensed graphs corresponding to different plans will have the same travelling arcs but will differ with respect to the control arcs. This can be used to make required computation of the orders of circuits more efficient. For generalised systems, this method is applied for those plans which have negative order arcs. If the plans only have non-negative order arcs, the simpler method (2.16) is enough.

4.3 Strictly cyclic systems with negative order arcs

4.3.1 Basic idea — eliminating negative order

The introduction of negative order arcs implies that some events in a previous cycle depend on the occurrence of events in later cycles. For a plan where the lowest order arc has order $-K_n$ ($0 \leq K_n \leq N_o$), the possible maximum order is $K_n + 1$ (see page 60) and the

corresponding max-plus algebra model can be represented as:

$$\begin{aligned}\underline{X}(k) &= A_{-K_n}(k)\underline{X}(k+K_n) \oplus \cdots \oplus A_{-1}(k)\underline{X}(k+1) \oplus A_0(k)\underline{X}(k) \oplus \\ &\quad \oplus A_1(k-1)\underline{X}(k-1) \oplus \cdots \oplus A_{K_n+1}(k-K_n-1)\underline{X}(k-K_n-1) \oplus \\ &\quad \oplus Bu, \qquad (4.3)\\ Y(k) &= C \otimes \underline{X}(k). \qquad (4.4)\end{aligned}$$

The elements of $A_{-i}(k)$ correspond to the negative order arcs pointing from cycle $(k+i)$ to cycle k, $(0 < i \leq K_n)$. The elements of $A_i(k-i)$ correspond to the non-negative order arcs pointing from cycle $(k-i)$ to cycle k, $(0 \leq i \leq K_n+1)$. Since $\underline{X}(k)$ depends on $\underline{X}(k+i)$, $(1 \leq i \leq K_n)$, the control is difficult for such systems.

Fortunately, for many application fields such as manufacturing systems, chemical batch plants and transportation systems, a strictly cyclic operation style is widely used. This simplifies the task of control, even for plans with negative order arcs. In the strictly cyclic case, the system matrices of the max-plus model are constant, thus (4.3) becomes

$$\begin{aligned}\underline{X}(k) &= A_{-K_n}\underline{X}(k+K_n) \oplus \cdots \oplus A_{-1}\underline{X}(k+1) \oplus A_0\underline{X}(k) \oplus \\ &\quad \oplus A_1\underline{X}(k-1) \oplus \cdots \oplus A_{K_n+1}\underline{X}(k-K_n-1) \oplus Bu. \qquad (4.5)\end{aligned}$$

Geyer [42] proposed a simple control strategy for systems of the type (4.5). The idea behind it is to shift certain states (events) in each cycle several cycles forward or backward so that all negative order arcs are eliminated:

$$\begin{aligned}\text{shift one cycle forward:} &\quad \underline{x}_i(k) \Leftarrow \underline{x}_i(k-1), \text{ for some } i,\\ \text{shift one cycle backward:} &\quad \underline{x}_i(k) \Leftarrow \underline{x}_i(k+1), \text{ for some } i.\end{aligned}$$

Fig. 4.8 illustrates the method of eliminating negative order arcs by shifting some events one cycle forward. After event 1 and event 2 in cycle $k-1$ have been shifted one cycle forward, they are treated as events in a new cycle, Ncycle k. The original negative order arc a_2 now becomes a zero order arc. Besides, the original zero order arcs a_3 become first order arcs while the original $+2$-order arcs b_4 also become first order arcs. In fact, if an event is shifted l cycles forward, the orders of the arcs pointing to it increase by l while the orders of the arcs pointing from it decrease by l. The situation is the opposite when an event is shifted l cycles backward. As a result, although the cycle index for the shifted events have been changed, the sequence relations between events remain unchanged.

It is not necessary that all shifted events are shifted the same number of cycles. It also could happen that some events are shifted forward while some other events are shifted

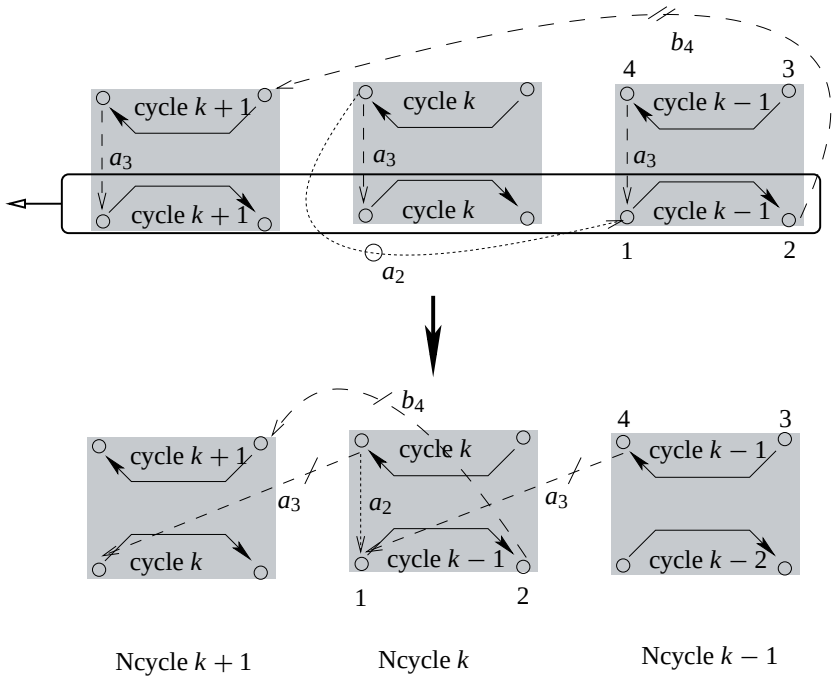

Figure 4.8: Eliminate negative order

backward. To solve the problem of eliminating negative order arcs, we can first shift every event i by f_i cycles forward (where $f_i \in \mathbb{Z}$). For any arc a_{ij} with order $oc(a_{ij})$, its new order is required to be greater than or equal to 0, i.e.

$$oc(a_{ij}) + f_i - f_j \geq 0, \quad 1 \leq i, j \leq n. \tag{4.6}$$

The solution f_i, $(i = 1, \cdots, n)$ satisfying the above inequalities can be used to eliminate negative orders. The solution, if it exists, may be non-unique. For the problem shown in Fig. 4.8, we have

$$\begin{aligned} -1 + f_1 - f_4 \geq 0, \qquad & 0 + f_1 - f_4 \geq 0, \\ 0 + f_2 - f_1 \geq 0, \qquad & 0 + f_4 - f_3 \geq 0, \\ 2 + f_3 - f_2 \geq 0. & \end{aligned}$$

Thus, $2 + f_3 \geq f_2 \geq f_1 \geq f_4 \geq f_3$ and $f_1 \geq 1 + f_4$. Among the possible solutions, $f_1 = f_2 = 1$, $f_3 = f_4 = 0$ is the solution shown in the figure. If $f_3 = f_4 = 0$ holds, another possible solution is $f_1 = f_2 = 2$ which means shifting event 1 and event 2 in each cycle by 2 cycles forward. Because of the +2-order arc b_4, two is the maximum number of cycles event 2 can be shifted forward when event 3 is kept un-shifted.

4.3.2 Model representation

The max-plus model (4.5), after eliminating negative orders, becomes

$$\begin{aligned}\underline{\widetilde{X}}(k) &= \widetilde{A_0}\underline{\widetilde{X}}(k) \oplus \widetilde{A_1}\underline{\widetilde{X}}(k-1) \oplus \widetilde{A_2}\underline{\widetilde{X}}(k-2) \oplus \\ &\quad \oplus \cdots \oplus \widetilde{A_{k_1}}\underline{\widetilde{X}}(k-k_1) \oplus \widetilde{B}u, \end{aligned} \tag{4.7}$$

where k_1, $\widetilde{B}$ and $\widetilde{A_j}(j = 1, \cdots, k_1)$ depend on the particular shifting solution. This event-shifting method for eliminating negative orders only works for strictly cyclic systems, where the system matrices A_i are constant and the sequences of events are fixed.

From the original system matrices A_i to the new $\widetilde{A_j}$, the process of eliminating negative orders can be interpreted as a matrix transformation. This is described in [42] and is briefly presented in the following. With the introduction of the γ-transformation,

$$\underline{X}(\gamma) = \bigoplus_{k\in\mathbb{Z}} \underline{X}(k) \otimes \gamma^k, \; U(\gamma) = \bigoplus_{k\in\mathbb{Z}} u(k) \otimes \gamma^k, \tag{4.8}$$

(4.5) can be represented as

$$\underline{X}(\gamma) = A(\gamma)\underline{X}(\gamma) \oplus BU(\gamma), \quad \text{where } A(\gamma) = \bigoplus_{i=-K_n}^{K_n+1} A_i\gamma^i. \tag{4.9}$$

The transformation matrix $\mathcal{T}$ is a diagonal matrix which satisfies

$$\underline{\widetilde{X}}(\gamma) = \mathcal{T} \otimes \underline{X}(\gamma) \quad \text{and} \quad \mathcal{T}^{-1} \otimes \underline{\widetilde{X}}(\gamma) = \underline{X}(\gamma),$$

where $\mathcal{T}^{-1}$ represents the max-plus inverse of $\mathcal{T}$, i. e. $\mathcal{T}^{-1} \otimes \mathcal{T} = I$.

With the transformation matrix $\mathcal{T}$, (4.9) can be transformed to

$$\underline{\widetilde{X}}(\gamma) = \widetilde{A}(\gamma)\underline{\widetilde{X}}(\gamma) \oplus \mathcal{T}BU(\gamma), \quad \text{with } \widetilde{A}(\gamma) = \mathcal{T}A(\gamma)\mathcal{T}^{-1}. \tag{4.10}$$

Then $\widetilde{A_j}$ can be directly derived since $\widetilde{A}(\gamma) = \bigoplus_{j=0}^{k_1} \widetilde{A_j}\gamma^j$. To eliminate the negative order, the elements of the diagonal matrix $\mathcal{T} = diag[\gamma^{f_1}, \cdots, \gamma^{f_n}]$ are determined such that $\widetilde{A_{-1}}, \widetilde{A_{-2}}, \cdots$ vanish in the max-plus sense. The transformation matrix represents the shifting of events. Specifically, $\mathcal{T}$ shifts event $i(i = 1, \cdots, n)$ f_i cycles forward. $\mathcal{T}A(\gamma)$ implies the order increasing of the arcs pointing to the forward shifted events while $A(\gamma)\mathcal{T}^{-1}$ represents the order decreasing of the arcs pointing from the forward shifted events. In Fig. 4.8, the corresponding $\mathcal{T}$, $\mathcal{T}^{-1}$, $A(\gamma)$, $\widetilde{A}(\gamma)$ are

$$\mathcal{T} = \begin{bmatrix} \gamma^{f_1} & \epsilon & \epsilon & \epsilon \\ \epsilon & \gamma^{f_2} & \epsilon & \epsilon \\ \epsilon & \epsilon & \gamma^{f_3} & \epsilon \\ \epsilon & \epsilon & \epsilon & \gamma^{f_4} \end{bmatrix} = \begin{bmatrix} \gamma^1 & \epsilon & \epsilon & \epsilon \\ \epsilon & \gamma^1 & \epsilon & \epsilon \\ \epsilon & \epsilon & e & \epsilon \\ \epsilon & \epsilon & \epsilon & e \end{bmatrix}, \; \mathcal{T}^{-1} = \begin{bmatrix} \gamma^{-1} & \epsilon & \epsilon & \epsilon \\ \epsilon & \gamma^{-1} & \epsilon & \epsilon \\ \epsilon & \epsilon & e & \epsilon \\ \epsilon & \epsilon & \epsilon & e \end{bmatrix},$$

$$A(\gamma) = \begin{bmatrix} \epsilon & \epsilon & \epsilon & (A_{-1})_{14}\gamma^{-1} \oplus (A_0)_{14} \\ (A_0)_{21} & \epsilon & \epsilon & \epsilon \\ \epsilon & (A_2)_{32}\gamma^2 & \epsilon & \epsilon \\ \epsilon & \epsilon & (A_0)_{43} & \epsilon \end{bmatrix},$$

$$\widetilde{A}(\gamma) = \begin{bmatrix} \epsilon & \epsilon & \epsilon & (A_{-1})_{14} \oplus (A_0)_{14}\gamma^1 \\ (A_0)_{21} & \epsilon & \epsilon & \epsilon \\ \epsilon & (A_2)_{32}\gamma^1 & \epsilon & \epsilon \\ \epsilon & \epsilon & (A_0)_{43} & \epsilon \end{bmatrix}.$$

From $\widetilde{A}(\gamma)$, we can get $\widetilde{A_0}$ and $\widetilde{A_1}$ which means $k_1 = 1$. In many cases, however, $k_1 > 1$. To simplify control, (4.7) is transformed further by introducing new state variables. For a system with n events, suppose there is one event, say event 3, whose event time in cycle k (i. e. $x_3(k)$) depends on some event times in cycle $k-2$, e.g. $\tilde{x}_3(k) = (\widetilde{A_2})_{35} \otimes \tilde{x}_5(k-2)$. Then a new state variable $\tilde{x}_{n+1}(k) = (\widetilde{A_2})_{35} \otimes \tilde{x}_5(k-1)$ is introduced so that $\tilde{x}_3(k)$ depends on event time in cycle $k-1$, i.e. $\tilde{x}_3(k) = \tilde{x}_{n+1}(k-1)$.

Note that this parallels the standard procedure in continuous control to transform a higher difference equation to a first order equation.

Generally, for system with $A_0, \cdots, A_{k_1}(k_1 \geq 2)$, if a matrix element $(A_k)_{ji} \neq \epsilon$, $1 < k \leq k_1$, then there are correspondingly $k-1$ new state variables to be introduced to make $(A_k)_{ji} = \epsilon$. The resulting matrices $A_{0\text{new}}$, B_{new} and $A_{1\text{new}}$ are:

$$A_{0\text{new}} = \left[\begin{array}{c|c} A_0 & \boldsymbol{\epsilon} \\ \hline \boldsymbol{\epsilon} & N_{(k-1)\times(k-1)} \end{array}\right], \quad B_{\text{new}} = \left[\begin{array}{c} B \\ \hline \boldsymbol{\epsilon} \end{array}\right],$$

$$A_{1new} = \left[\begin{array}{ccc|c|c} & & & \epsilon & \\ & & & \vdots & \\ & & & \epsilon & \\ & A_1 & & e & \boldsymbol{\epsilon} \\ & & & \epsilon & \\ & & & \vdots & \\ & & & \epsilon & \\ \hline & \boldsymbol{\epsilon} & & \epsilon & I_{(k-2)\times(k-2)} \\ \hline \epsilon & \cdots & \epsilon \quad (A_k)_{ji} \quad \epsilon \quad \cdots \quad \epsilon & \epsilon & \boldsymbol{\epsilon} \end{array}\right] \begin{array}{l} \\ \\ \\ \leftarrow \\ j^{th} \text{ line} \\ \\ \\ \\ \\ \\ \\ \uparrow i^{th} \text{ row} \end{array}$$

Now, system (4.7) with $k_1 \geq 2$, can be transformed finally to the following form:

$$\underline{\widehat{X}}(k) = \widehat{A}_0 \underline{\widehat{X}}(k) \oplus \widehat{A}_1 \underline{\widehat{X}}(k-1) \oplus \widehat{B}u. \tag{4.11}$$

With this new system representation, system performance can be conveniently analysed through the eigenvalue of $\widehat{A}_0$. Recall that the derivation of the new system representation is based on the idea of event-shifting. After shifting different events several cycles either forward or backward, different original negative order models may be transformed to the same non-negative order system representation $\widetilde{A}$ and therefore the same $\{\widehat{A}_0, \widehat{A}_1\}$ representation. This means that different plans could have the same $\{\widehat{A}_0, \widehat{A}_1\}$ model. Since event shifting only works for strictly cyclic systems, i. e. for systems in which the event relations are identical for different cycles, plans cannot be changed during the run-time of the system. The delay caused by unexpected events can therefore only be recovered by the introduction of a safety margin between the cycle time and the system eigenvalue. In order to make it possible to change plans when a delay happens, a more general modelling method for systems with negative order arcs needs to be studied. This topic will be discussed in the next sections.

4.4 Feasible plan list

Strictly cyclic patterns benefit from simple structure and steady progress, but they might not make the best use of resources. In many applications, a non-strictly cyclic pattern is preferred. For systems operating without a regular pattern, the modelling method based on event shifting is not applicable. For strictly cyclic systems, the method does not support a possible change of plan when delays happen. Before discussing a more general way to describe systems with negative order arcs, we first need to find feasible plan lists.

We discuss two methods to find feasible plan lists. The "circuit order checking" method is based on Lemma 5 and is an extension of Section 4.2, while the "big-matrix checking" method is based on the method described in Section 2.2.3.

4.4.1 Circuit order checking method

For cyclicly repeated systems, because of the existence of negative order arcs, not every feasible plan can be connected to arbitrary feasible plans. It is no problem to connect any feasible plan to a previous non-negative order plan though. Things are different when the

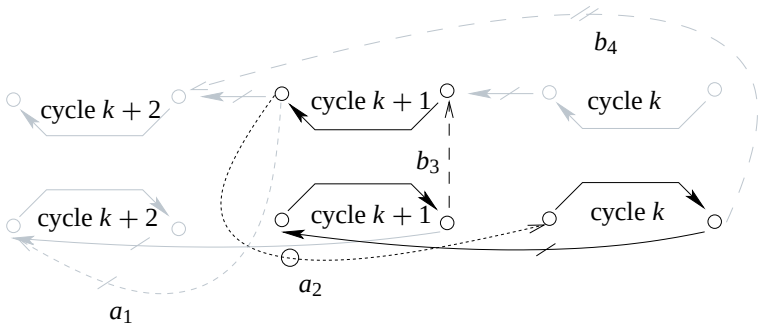

Figure 4.9: Blocking

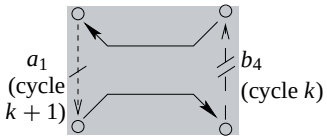

Figure 4.10: Nonblocking

plan in the previous cycle has negative order arcs. Although a feasible negative order plan (i. e. a plan with negative order control arcs) can be repeated periodically without causing blocking, it is not safe to operate a different plan immediately after the previous cycle. As negative order arcs create a path from the following cycle to the previous cycle, the combination of different plans in sequential cycles may cause blocking. Although both are feasible plans, if the negative order plan 4.4 (c) (in cycle k) is followed by the plan shown in 4.4(a) (in cycle $k+1$), -1-order arc a_2 of cycle k and zero order arc b_3 in cycle $k+1$, together with some travelling arcs will result in blocking, see Fig.4.9.

Lemma 5 is used to check if a plan can follow a negative order plan without causing blocking. Similar to the identification of a feasible negative order plan by checking the orders of its circuits, now we need to check the orders of every potential circuit which satisfies the following requirements:

1. The potential circuit contains more than one control arc.
2. The plan implemented in the first cycle contains at least one negative order control arc.
3. The plan implemented in the second cycle contains at least one control arc.

For example, in the condensed graph shown in Fig.4.10, we do not need to check the potential circuit composed of control arc b_4 in cycle k and control arc a_1 in cycle $k+1$, because requirement 2 fails.

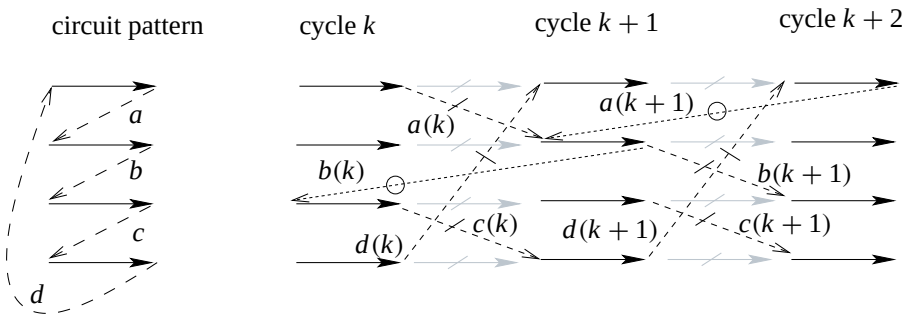

Figure 4.11: A circuit pattern and its selections

As discussed in Section 4.1, different plans have the same circuit patterns pac_j. For cycle k, which implements a negative order plan $Pl(k)$, and cycle $(k+1)$, which implements a plan $Pl(k+1)$, there are the potential circuits $C_j(k)$ and $C_j(k+1)$ respectively, both of them corresponding to the same circuit pattern pac_j. Because the two cycles are connected sequentially, some control arcs of $C_j(k)$ and some other control arcs of $C_j(k+1)$ may form more potential circuits which still have the same pattern pac_j. In other words, a specific pattern pac_j composed of k_c control arcs, can correspond to different combinations of control arcs, e.g. i control arcs from cycle k and $(k_c - i)$ control arcs from cycle $(k+1)$ for $1 \leq i \leq k_c - 1$. Generally, for two sequential cycles, without considering $C_j(k)$ and $C_j(k+1)$, a specific pattern pac_j may correspond to up to $K_{sel} = \sum_{i=1}^{k_c-1} C_{k_c}^i$ different selections, where $C_{k_c}^i = k_c!/(i!(k_c - i)!)$. But as stated by requirement 2, we only need to consider potential circuits with at least one negative order arc in the plan implemented in the first cycle. So K_{sel} can be reduced. Let the number of negative order arcs in cycle k be k_{nc}, $(k_{nc} \leq k_c)$, then the number of combinations is

$$K_{sel} = \sum_{j=1}^{\min(k_{nc},k_c-1)} (C_{k_{nc}}^j \sum_{i=0}^{k_c-j-1} C_{k_c-j}^i).$$

Note that not each combination corresponds to a potential circuit. Sometimes a combination is even not a path. An example is shown in Fig. 4.11. For a pattern containing control arcs a, b, c, d and plans implemented in cycle k and cycle $k+1$ as shown in Fig. 4.11, the combination of $\{b(k), d(k), a(k+1), c(k+1)\}$ is not a circuit: although $a(k+1), b(k), c(k+1)$ lead to a path in the extended graph, the control arc $d(k)$ starts from cycle k, while $c(k+1)$, the expected predecessor of d according to the circuit pattern pac_j, points to a later cycle $k+2$ instead of cycle k. There is no path from $c(k+1)$ to $d(k)$.

To find out if a combination corresponds to a circuit, we introduce the following notation:

$cca(i, j)$:	the number of cycle changes between a control arc i and its successor control arc in circuit pattern pac_j
$sel(l, j)$:	a combination of control arcs corresponding to circuit pattern $pac_j, l \leq K_{sel}$
$i_{suc}(i, j)$:	the index of successor control arc for control arc i in pac_j
$i_{pre}(i, j)$:	the index of predecessor control arc for control arc i in pac_j
$oc(i, k)$:	the order of control arc i chosen by the plan in cycle k
$CI(i, sel)$:	cycle index of control arc i in combination sel
$Ia(i, k)$:	the index of the cycle to which the control arc i in cycle k points
$Oa(i, k)$:	the index of the cycle from which the control arc i in cycle k points

In a combination sel, for control arc i, we have

$$\begin{aligned} Ia(i, CI(i, sel)) &= CI(i, sel) + \max(0, oc(i, CI(i, sel))), \\ Oa(i, CI(i, sel)) &= CI(i, sel) - \min(0, oc(i, CI(i, sel))). \end{aligned}$$

A combination $sel(i, j)$ corresponds to circuit (i. e. causes blocking) if and only if it has the following property:

$$\forall i, Ia(i, CI(i, sel)) + cca(i, j) \leq Oa(i_{suc}(i, j), CI(i, sel)). \tag{4.12}$$

For the circuit pattern shown in Fig. 4.11, $cca = 0$ holds for all four control arcs. The successors of control arcs are $i_{suc}(a) = b, i_{suc}(b) = c, i_{suc}(c) = d, i_{suc}(d) = a$. In combination $\{b(k), d(k), a(k+1), c(k+1)\}$,

$$\begin{aligned} CI(b, sel) = CI(d, sel) = k, &\qquad CI(a, sel) = CI(c, sel) = k+1, \\ oc(b, k) = -1; \quad oc(d, k) = 1, &\qquad oc(a, k+1) = -1; \quad oc(c, k+1) = 1, \\ Ia(b, k) = k + \max(0, oc(b, k)) = k, &\quad Oa(b, k) = k - \min(0, oc(b, k)) = k+1, \\ Ia(d, k) = k + \max(0, oc(d, k)) = k+1, &\qquad Oa(d, k) = k - \min(0, oc(d, k)) = k, \end{aligned}$$

$$\begin{aligned} Ia(a, k+1) &= k+1 + \max(0, oc(a, k+1)) = k+1, \\ Oa(a, k+1) &= k+1 - \min(0, oc(a, k+1)) = k+2, \\ Ia(c, k+1) &= k+1 + \max(0, oc(c, k+1)) = k+2, \\ Oa(c, k+1) &= k+1 - \min(0, oc(c, k+1)) = k+1. \end{aligned}$$

Since

$$Ia(c, k+1) + cca(c) = k+2+0 > Oa(d, k) = k,$$

the combination $\{b(k), d(k), a(k+1), c(k+1)\}$ cannot form a circuit. Another combination $\{b(k), c(k), a(k+1), d(k+1)\}$ satisfies (4.12). It must be a circuit.

For two sequential cycles with different plans,

$$\begin{aligned} &\text{if } \exists\ pac_j, \text{ and } \exists \text{ a combination } sel(l, j) \text{ which has the property (4.12),} \\ &\Longrightarrow \quad \text{the plan list is infeasible.} \end{aligned} \tag{4.13}$$

Using (4.13), for every negative order plan, we can find all plans which can not directly follow it. A matrix Pf is used to store the related results:

$$(Pf)_{ij} = \begin{cases} 1 & \text{if plan } j \text{ can directly follow plan } i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.14}$$

Of course, for a non-negative order plan i, $(Pf)_{ij}$ is always 1.

4.4.2 Determining the maximum number of cycles to be checked

The matrix Pf provides the feasibility information of plan lists which contain two plans. Suppose that in the first cycle of such a feasible list a negative order plan is implemented. If the order of the negative order plan is lower than -1 or if the feasible plan implemented in the second cycle is also of negative order, i. e. if there is a negative order control arc pointing from another plan attached to the entirety of two sequential cycle, it is still necessary to check the feasibility of lists with k_l sequential cycles ($k_l > 2$). We can always computes the maximum number of cycles, denoted as k_{lm}, over which the feasibility test needs to extend.

For a given system, k_{lm} not only depends on the number of control arcs which form each circuit pattern pac_j, but also on the lowest negative order (i. e. $-N_o$) of the control arcs in the system.

To find out the maximum possible value k_{lm}, we always consider the worst case. Suppose $K_n = N_o$. If the plan implemented in the first cycle is a negative order plan, there is a negative order control arc a in the first cycle with $-K_n$ as its order. Suppose a circuit pattern pac_j contains a.

- For the simplest case, suppose pac_j contains only two control arcs a and b.

  1. If $-K_n = -1$, control arc a in the first cycle points from the second cycle. To form a circuit, control arc b should also point to the second cycle. Since the

lowest negative order of the system is -1, the possible orders of b are 0 or -1, and b is in the second cycle (Note that with order higher than 0, b is in first cycle or even earlier, there is no need to consider such situations). Apparently $k_{lm} = 2$.

2. If $-K_n = -2$, control arc a in the first cycle points from the third cycle. To form a circuit, control arc b should point to the third cycle. If $-K_n \leq oc(b) \leq 0$, b is in the third cycle; if $oc(b) = 1$, b is in the second cycle and there is no need to consider other possible orders. Thus $k_{lm} = 3$.
3. Clearly, for a with order $-K_n$, it points from cycle $(K_n + 1)$. The possible orders of b are $-K_n \leq oc(b) \leq K_n - 1$ and $k_{lm} = K_n + 1$.

This argument can be easily extended to k_c control arcs with minimum order $-K_n$.

- Generally, for a circuit pattern pac_j with k_c control arcs, if the lowest possible order of the system is $-K_n$, the worst case for which a circuit may exist is that there is a corresponding negative order control arc of order $-K_n$ points from cycle $(1 + K_n)$ to cycle 1, from cycle $(1+2\times K_n)$ to cycle $(1+K_n)$, $\cdots$, from cycle $(1+k_c \times K_n)$ to cycle $(1 + (k_c - 1) \times K_n)$. The maximum possible number of cycles a plan list needs to be checked is then

$$k_{lm} = 1 + (k_c - 1) \times K_n. \tag{4.15}$$

Thus for a plan list starting with a negative order plan, if the cycle number of the list $k_l > k_{lm}$, only the feasibility of subparts of the original list need to be verified instead of doing so in the whole list. Each subpart starts from a negative order plan of the original list and contains its successive $(k_{sl} - 1)$ plans, where $k_{sl} - 1 = \min(k_{lm} - 1, k_l - k_p)$ and k_p is the cycle index of the plan.

4.4.3 Feasibility checking procedure

Suppose $2 < k_{lm} < k_l$. The method of checking the feasibility of a list containing two sequential cycles is similar to the method of checking the feasibility of a list containing k_{lm} cycles, both are based on (4.13). Up to $(k_{lm} - 1)$ steps are required to find all feasible lists of k_{lm} cycles. In each step $k = 1, \cdots, k_{lm} - 1$, feasible lists with $(k + 1)$ cycles are found based on the result of both the last step accordingly and (4.13).

Step k: For a feasible list fl containing k cycles, let the cycle indices be $[1, 2, \cdots, k]$ and the plan of cycle k be plan i_k.

1. Assign a plan j_{k+1} to cycle $(k+1)$ if $(Pf)_{i_k j_{k+1}} = 1$ and if the k-cycle plan list $[fl' j_{k+1}]$ is feasible. fl' is the same as fl except that it does not include the first cycle (i. e. cycle 1) of fl. So we have a plan list of $(k+1)$ cycles: $[1, 2, \cdots, k, k+1]$.
2. Similar to two-cycle case, check the feasibility of the new $(k+1)$-cycle plan list. If the added plan j_{k+1} is the same one as the last plan in the k-cycle list fl, it can be directly concluded that the new $(k+1)$-cycle list is feasible. If plan j_{k+1} is different to the last plan in fl, we use the same requirements corresponding to the feasibility test of two cycle case (page 69). Consider every circuit pattern pac_m which contains more than one control arc. Within these control arcs, it is required that:
   - the plan implemented in the first cycle contains at least one negative order control arc, i. e.
   $$\exists i_1 \in \{i | i \in AC_m, oc(i,1) < 0\}, \quad CI(i_1, sel) = 1, \tag{4.16}$$
   where AC_m is the index set of control arcs contained in pac_m.
   - the plan implemented in the last cycle contains at least one control arc, i. e.
   $$\exists i_2 \in AC_m, \quad CI(i_2, sel) = k+1, \ i_2 \neq i_1. \tag{4.17}$$

   If these requirements are met, we use (4.13) to check the feasibility of the new plan list.
3. For each plan j_{k+1} which could be implemented in cycle $(k+1)$, use the feasibility test in step $k(2)$ to find all feasible $(k+1)$-cycle lists whose first k cycles are fl.
4. For each feasible list fl, find all corresponding feasible $(k+1)$-cycle lists.

Since $k_l > k_{lm}$, based on all $(k+1)$-cycle feasible lists $(k = 1, \cdots, k_{lm}-1)$, the feasibility of a k_l-cycle list can be judged by checking its sublists starting with the negative order plans.

4.4.4 Big-matrix checking method

In the above step k (2), for the case that plan j_{k+1} is different to the previous cycle's plan i_k, if the number of circuits containing control arcs is very large, checking the feasibility of the plan list using (4.13) may take a very long time. To speed up feasibility checking for such systems especially when the dimension of the system is small, we build a block matrix $\mathbf{A}$ for a plan list PL so that we can use (2.16) to check the feasibility of the list, i. e.

$$\exists k \leq n, \mathbf{A}^k(i,i) > \epsilon \quad \Leftrightarrow \quad \text{the plan list } PL \text{ is infeasible.}$$

For system (4.3), let the maximum number of the cycles in a plan list be k_l, and denote

$$\underline{\mathbf{X}} = [\underline{X}^T(1) \;\; \underline{X}^T(2) \;\; \cdots \;\; \underline{X}^T(k_l-1) \;\; \underline{X}^T(k_l)]^T. \tag{4.18}$$

The corresponding system matrix can be represented as

$$\mathbf{A} = \begin{bmatrix} A_0(1) & A_{-1}(1) & \cdots & A_{-(k_l-2)}(1) & A_{-(k_l-1)}(1) \\ A_1(1) & A_0(2) & \cdots & A_{-(k_l-3)}(2) & A_{-(k_l-2)}(2) \\ & & \ddots & & \\ A_{(k_l-2)}(1) & A_{(k_l-3)}(2) & \cdots & A_0(k_l-1) & A_{-1}(k_l-1) \\ A_{(k_l-1)}(1) & A_{(k_l-2)}(2) & \cdots & A_1(k_l-1) & A_0(k_l) \end{bmatrix}. \tag{4.19}$$

Each block $(\mathbf{A})_{ij}$ shows the time influence of cycle j on cycle i,

$$(\mathbf{A})_{ij} = A_{i-j}(\min(i,j)). \tag{4.20}$$

Note that if $-K_n$ is the most negative order of the plan implemented in cycle k, then

$$\begin{aligned} A_{-i}(k) &= N, \text{ for } i > K_n, \\ A_i(k) &= N, \text{ for } i > K_n + 1. \end{aligned}$$

This *big-matrix checking method* can also be used effectively to find infeasible plans by holding all $A_i(k)$ constant. Many factors influence the efficiency of the feasibility checking methods: system dimension n, number of feasible plans, number of feasible negative order plans, number of cycles, number of circuit patterns with multi control arcs etc. Generally, this big-matrix checking method is faster than the circuit order checking method for relative smaller system. For example, for a system ($n = 16$) with 29 multi-control arc circuit patterns and with 64 feasible plans (in which 40 plans are of negative order), the big-matrix checking approach takes less time than the circuit order checking approach when $k_l = 2$. But if $k_l > 2$, the circuit order checking approach takes less time. To make better use of two methods, they can be combined.

4.5 List set updating

For a rail transportation application, the number of feasible plan lists is usually very large for high order systems. However, once a train enters a shared single-line track segment, a lot of lists can be deleted from the list set. In Section 2.2.3, for system with only $+1$-order arcs, the updating of the feasible plan set is based on the key in-node set $Kins$ and the corresponding matrix KN. If a train j has passed the location corresponding to the key in-node $(KN)_{ij}$, it is impossible to change from the currently implemented plan to plan i, i. e. plan i can be deleted from the feasible plan set. This idea is also applied here to update the set of feasible plan lists.

For cyclic systems, the plans are different because the orders of the control arcs on shared single-line track segments are different. A control arc with order oc_o on the shared single-line track segment also satisfies the constraints described by control arcs on the same location with order $oc \geq oc_o$. However, it violates the constraints of the ones with order $oc < oc_o$. Suppose the currently operated plan in cycle k has a control arc with order oc_o and another plan j has the same control arc but it has a order oc which is less than oc_o. If the in-node event corresponding to the control arc happens, then all plan lists which operate plan j in cycle k should be deleted from the list set.

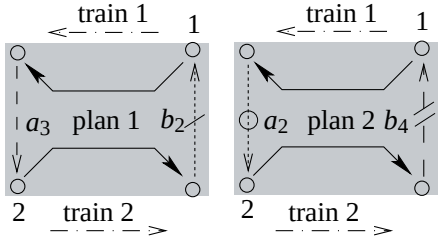

Figure 4.12: Key in-node

Fig. 4.12 shows the control arcs of plan 1 and 2 on the same single-line track. Plan 1 is shown in the left part of the figure, plan 2 in the right part. The in-node set $Ins1 = \{1 \;\; 2\}$. In plan 1, the order of the control arcs related to the in-node set is $Oin1 = [1 \;\; 0]$. Correspondingly, for plan 2: $Ins2 = \{1 \;\; 2\} = Ins1$, $Oin2 = [2 \;\; -1]$. Suppose the currently operated plan in cycle k is plan 1 and train 1 in cycle k has passed in-node 1. Since b_2 has order 1, train 2 in cycle $(k-1)$ has already left the track. As the order of b_2 is less than the order of b_4, it does not violate control arc b_4. It is still possible to change current plan 1 to plan 2. On the other hand, if train 2 in cycle k has passed in-node 2, plan 2 will become infeasible since the order of a_3 is greater than the order of a_2. Thus

the key in-node (of train 2) for changing plan 1 to plan 2 is in-node 2. The in-node 1 is not a key in-node in this case.

The corresponding KN matrix is

$$KN = \begin{array}{c} \begin{array}{cc} \text{train 1} & \text{train 2} \\ \downarrow & \downarrow \end{array} \\ \left[\begin{array}{cc} +\infty & +\infty \\ +\infty & 2 \end{array}\right] \end{array} \begin{array}{ll} \leftarrow & \text{plan 1} \\ \leftarrow & \text{plan 2} \end{array}$$

In general, for the currently operated plan in cycle k, its in-node set is Ins. Each element in Ins (i. e. $(Ins)_i$) corresponds to an event on a shared single-line track segment and is pointed to by a control arc. The order of the control arc related to the in-node $(Ins)_i$ is $(Oin)_i$. Also, let the order of the same control arc of another plan j be $(Oinj)_i$. Then, in order to change to plan j, $(Ins)_i$ is an element of the key in-node set $kins$ if

$$(Oin)_i > (Oinj)_i. \tag{4.21}$$

As we discussed before, the $kins$ can be simplified to a "$Kins$" set and results in KN matrix. When the event corresponding to a key in-node $(KN)_{ij}$ happens, all plan lists containing plan j in cycle k should be deleted from the list set.

4.6 Upper level model

For a cyclicly-running system with $-N_o$ as the lowest order of control arc, an appropriate model is given by

$$\underline{X}(k) = A_0(k)\underline{X}(k) \bigoplus_{i=1}^{N_o+1} A_i(k-i)\underline{X}(k-i) \bigoplus_{i=1}^{N_o} A_{-i}(k)\underline{X}(k+i). \tag{4.22}$$

It is an implicit model containing event times in future cycles, $\underline{X}(k+i)$. As in the previous chapters, the corresponding explicit model is required. For a given feasible plan list, if there is no negative order plan, each cycle can be viewed as an independent unit in the sense that the unit does not depend on its successor cycles, i. e. $A_{-i} = N$ for $i \geq 1$. If there are negative order plans in the list, besides the single-cycle units, there are units which contain a sequence of cycles. For such a unit containing cycles k: $k_1, \cdots, k_u$, the first cycle (i. e. cycle k_1) is a negative order plan. k_u is determined in such a way that starting from cycle k_1, cycle k_u is the first cycle satisfying the following condition: there

is no negative order arc pointing from its subsequent cycles to any cycle $k \in \{k_1, \cdots, k_u\}$. Thus, as an independent unit, it does not depend on its successor cycles. This situation happens when either

- there is no negative order arc pointing from subsequent cycles into the unit, this, of course, implies that cycle k_u implements a non-negative order plan.
- or, the cycle sequence $k = [k_1, \cdots, k_u]$ is the last part of the original feasible plan list so that there is no subsequent cycle and cycle k_u is the last cycle of the plan list. In this case, cycle k_u may also implement a negative order plan.

In both cases we have

$$\begin{aligned} \underline{X}(k_u) &= A_0(k_u)\underline{X}(k_u) \bigoplus_{i=1}^{N_o+1} A_i(k_u - i)\underline{X}(k_u - i) \\ &= A_0^*(k_u) \otimes \left[\bigoplus_{i=1}^{N_o+1} A_i(k_u - i)\underline{X}(k_u - i) \right]. \end{aligned} \tag{4.23}$$

Of course, (4.23) also can be written as $\underline{X}(k_u) = \bigoplus_{i=1}^{N_o+1} A_0^*(k_u)A_i(k_u - i)\underline{X}(k_u - i)$.

Thus, for cycle $(k_u - 1)$, its lowest possible order would be -1. Since the highest order of the system is $N_o + 1$, $A_i = N$, for $i > N_o + 1$. Hence, we get for cycle $(k_u - 1)$:

$$\begin{aligned} \underline{X}(k_u - 1) &= A_0(k_u - 1)\underline{X}(k_u - 1) \bigoplus_{i=1}^{N_o+1} A_i(k_u - 1 - i)\underline{X}(k_u - 1 - i) \\ &\quad \oplus A_{-1}(k_u - 1)\underline{X}(k_u). \end{aligned}$$

By inserting (4.23), we get

$$\begin{aligned} \underline{X}(k_u - 1) &= \left[A_0(k_u - 1) \oplus A_{-1}(k_u - 1)A_0^*(k_u)A_1(k_u - 1)\right] \underline{X}(k_u - 1) \ \oplus \\ &\quad \bigoplus_{i=1}^{N_o+1} \Big\{ \left[A_{-1}(k_u - 1)A_0^*(k_u)A_{i+1}(k_u - 1 - i) \oplus A_i(k_u - 1 - i)\right] \otimes \\ &\quad \otimes \underline{X}(k_u - 1 - i) \Big\} \\ &= \widetilde{A_0}(k_u - 1)\underline{X}(k_u - 1) \bigoplus_{i=1}^{N_o+1} \widetilde{A}_i(k_u - 1 - i)\underline{X}(k_u - 1 - i), \end{aligned} \tag{4.24}$$

where $\widetilde{A_0}(k_u - 1) = A_0(k_u - 1) \oplus A_{-1}(k_u - 1)A_0^*(k_u)A_1(k_u - 1)$ contains both direct and indirect time influences of cycle $(k_u - 1)$. $\widetilde{A}_i(k_u - 1 - i) = A_{-1}(k_u - 1)A_0^*(k_u)A_{i+1}(k_u -$

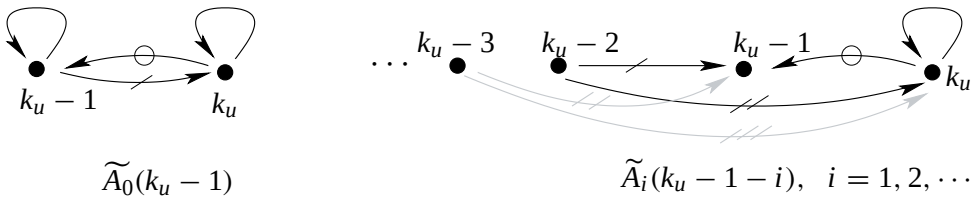

Figure 4.13: Direct and indirect time influence

$1 - i) \oplus A_i(k_u - 1 - i)$ represents the direct time influence of cycle $(k_u - 1 - i)$ on cycle $(k_u - 1)$ as well as the indirect time influence of cycle $(k_u - 1 - i)$ on cycle $(k_u - 1)$ via cycle k_u, see Fig. 4.13 in which each black dot corresponds to a vector $\underline{X}$ in each cycle.

By inserting (4.24) into itself repeatedly, we get

$$\begin{aligned}
\underline{X}(k_u - 1) &= \widetilde{A_0}^2(k_u - 1) \otimes \underline{X}(k_u - 1) \oplus \\
&\quad \oplus\big[\widetilde{A_0}(k_u - 1) \oplus I\big] \otimes \left[\bigoplus_{i=1}^{N_o+1} \widetilde{A}_i(k_u - 1 - i)\underline{X}(k_u - 1 - i)\right] \\
&\ \ \vdots \\
&= \widetilde{A_0}^n(k_u - 1) \otimes \underline{X}(k_u - 1) \oplus \big[\widetilde{A_0}^{n-1}(k_u - 1) \oplus \cdots \oplus \\
&\quad \oplus\widetilde{A_0}(k_u - 1) \oplus I\big] \otimes \left[\bigoplus_{i=1}^{N_o+1} \widetilde{A}_i(k_u - 1 - i)\underline{X}(k_u - 1 - i)\right] \\
&= \widetilde{A_0}^*(k_u - 1) \otimes \left[\bigoplus_{i=1}^{N_o+1} \widetilde{A}_i(k_u - 1 - i)\underline{X}(k_u - 1 - i)\right]. \qquad (4.25)
\end{aligned}$$

The last identity results from $\widetilde{A_0}^n(k_u - 1) = N$ which is a consequence of the feasibility of the plan list.

For cycle $(k_u - 2)$, the possible lowest order of control arc is -2, thus

$$\begin{aligned}
\underline{X}(k_u - 2) &= A_0(k_u - 2)\underline{X}(k_u - 2) \bigoplus_{i=1}^{N_o+1} A_i(k_u - 2 - i)\underline{X}(k_u - 2 - i) \\
&\quad \oplus A_{-1}(k_u - 2)\underline{X}(k_u - 1) \oplus A_{-2}(k_u - 2)\underline{X}(k_u).
\end{aligned}$$

We can replace the last two items in the above formula by

$$\begin{aligned}
A_{-2}(k_u - 2)\underline{X}(k_u) &= A_{-2}(k_u - 2)A_0^*(k_u)A_1(k_u - 1)\underline{X}(k_u - 1) \oplus \\
&\quad \oplus A_{-2}(k_u - 2)A_0^*(k_u) \otimes \left[\bigoplus_{i=2}^{N_o+1} A_i(k_u - i)\underline{X}(k_u - i)\right],
\end{aligned}$$

$$A_{-1}(k_u-2)\underline{X}(k_u-1)\oplus A_{-2}(k_u-2)\underline{X}(k_u) =$$
$$\widetilde{A_{-1}}(k_u-2)\underline{X}(k_u-1)\oplus A_{-2}(k_u-2)A_0^*(k_u)\otimes\left[\bigoplus_{i=2}^{N_o+1}A_i(k_u-i)\underline{X}(k_u-i)\right],$$

where $\widetilde{A_{-1}}(k_u-2) = A_{-1}(k_u-2)\oplus A_{-2}(k_u-2)A_0^*(k_u)A_1(k_u-1)$.

Therefore,

$$\begin{aligned}\underline{X}(k_u-2) &= A_0(k_u-2)\underline{X}(k_u-2)\bigoplus_{i=1}^{N_o+1}A_i(k_u-2-i)\underline{X}(k_u-2-i)\\ &\quad\oplus\widetilde{A_{-1}}(k_u-2)\underline{X}(k_u-1)\oplus\\ &\quad\oplus A_{-2}(k_u-2)A_0^*(k_u)\otimes\left[\bigoplus_{i=2}^{N_o+1}A_i(k_u-i)\underline{X}(k_u-i)\right].\end{aligned}$$

Inserting (4.25) then gives

$$\begin{aligned}\underline{X}(k_u-2) &= \Big[A_0(k_u-2)\oplus\widetilde{A_{-1}}(k_u-2)\widetilde{A_0}^*(k_u-1)\widetilde{A_1}(k_u-2)\oplus\\ &\quad\oplus A_{-2}(k_u-2)A_0^*(k_u)A_2(k_u-2)\Big]\underline{X}(k_u-2)\oplus\\ &\quad\left[\bigoplus_{i=1}^{N_o+1}A_i(k_u-2-i)\underline{X}(k_u-2-i)\right]\oplus\\ &\quad\oplus\widetilde{A_{-1}}(k_u-2)\widetilde{A_0}^*(k_u-1)\otimes\left[\bigoplus_{i=2}^{N_o+1}\widetilde{A}_i(k_u-1-i)\underline{X}(k_u-1-i)\right]\\ &\quad\oplus A_{-2}(k_u-2)A_0^*(k_u)\otimes\left[\bigoplus_{i=3}^{N_o+1}A_i(k_u-i)\underline{X}(k_u-i)\right].\end{aligned}\tag{4.26}$$

Denote the coefficients of $\underline{X}(k_u-2-i)$ as $\widetilde{A}_i(k_u-2-i)$ for $i=0,\cdots,N_o+1$, i. e.

$$\begin{aligned}\widetilde{A}_i(k_u-2-i) &= A_i(k_u-2-i)\oplus\\ &\quad\oplus\widetilde{A_{-1}}(k_u-2)\widetilde{A_0}^*(k_u-1)\widetilde{A_{i+1}}(k_u-2-i)\oplus\\ &\quad\oplus A_{-2}(k_u-2)A_0^*(k_u)A_{i+2}(k_u-2-i).\end{aligned}$$

Finally, $\underline{X}(k_u-2-i)$ can be described as (4.27). Fig. 4.14 shows the components of $\widetilde{A_{-1}}(k_u-2)$, of $\widetilde{A_0}(k_u-2)$ and of $\widetilde{A}_i(k_u-2-i)$, $i=1,\cdots,N_o+1$.

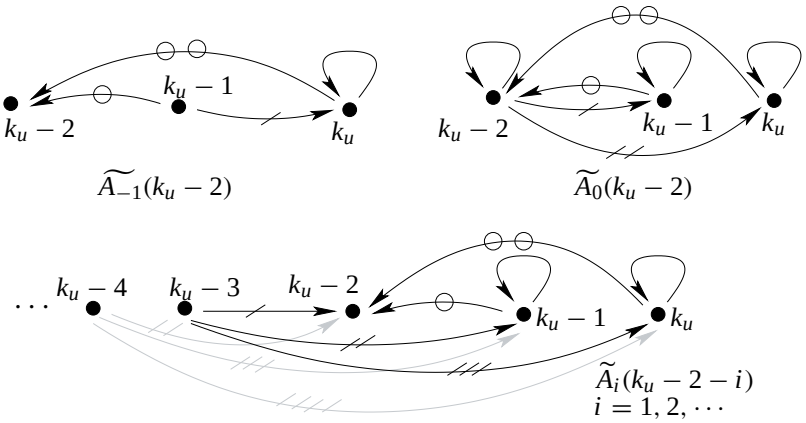

Figure 4.14: Components: $\widetilde{A_{-1}}(k_u - 2)$, $\widetilde{A_0}(k_u - 2)$ and $\widetilde{A}_i(k_u - 2 - i)$

$$\begin{aligned}
\underline{X}(k_u - 2) &= \widetilde{A_0}(k_u - 2)\underline{X}(k_u - 2) \bigoplus_{i=1}^{N_o+1} [A_i(k_u - 2 - i) \oplus \\
&\quad \oplus \widetilde{A_{-1}}(k_u - 2)\widetilde{A_0}^*(k_u - 1)\widetilde{A_{i+1}}(k_u - 2 - i) \oplus \\
&\quad \oplus A_{-2}(k_u - 2)A_0^*(k_u)A_{i+2}(k_u - 2 - i)]\underline{X}(k_u - 2 - i) \\
&= \widetilde{A_0}(k_u - 2)\underline{X}(k_u - 2) \bigoplus_{i=1}^{N_o+1} \widetilde{A}_i(k_u - 2 - i)\underline{X}(k_u - 2 - i) \\
&= \widetilde{A_0}^*(k_u - 2) \otimes [\bigoplus_{i=1}^{N_o+1} \widetilde{A}_i(k_u - 2 - i)\underline{X}(k_u - 2 - i)]. \qquad (4.27)
\end{aligned}$$

Now it is clear that inside an independent unit the influence of subsequent cycles on previous cycles can be included into corresponding new system matrices $\widetilde{A}_i(k)$. In general, given an implicit model

$$\underline{X}(k) = A_0(k)\underline{X}(k) \bigoplus_{i=1}^{N_o+1} A_i(k - i)\underline{X}(k - i) \bigoplus_{i=1}^{N_o} A_{-i}(k)\underline{X}(k + i) \oplus \underline{X}\_in(k), \quad (4.28)$$

for an independent unit $k = [k_1, \cdots, k_u]$, which, by definition, does not have negative order arcs pointing into the unit from the outside, the corresponding explicit model is:

$$\underline{X}(k) = \widetilde{A_0}^*(k)\widetilde{\underline{X}\_in}(k) \bigoplus_{i=1}^{N_o+1} \widetilde{A_0}^*(k)\widetilde{A}_i(k - i)\underline{X}(k - i), \qquad (4.29)$$

or equivalently, if we denote $\widetilde{A_0}^*(k)\widetilde{A_{k-j}}(j)$, $j < k$ by $AKJ(k, j)$,

$$\underline{X}(k) = \widetilde{A_0}^*(k)\widetilde{\underline{X}\_in}(k) \bigoplus_{j=k-N_o-1}^{k-1} AKJ(k,j)\underline{X}(j). \tag{4.30}$$

The following equations and figures show the corresponding $\widetilde{\underline{X}\_in}(k)$, $\widetilde{A_{-i}}(k)$ and $\widetilde{A}_i(k-i)$ for $k = [k_u, k_u - 1, \cdots, k_1]$.

$$\widetilde{\underline{X}\_in}(k) = \underline{X}\_in(k) \bigoplus_{i=1}^{N_o} \widetilde{A_{-i}}(k)\widetilde{A_0}^*(k+i)\widetilde{\underline{X}\_in}(k+i). \tag{4.31}$$

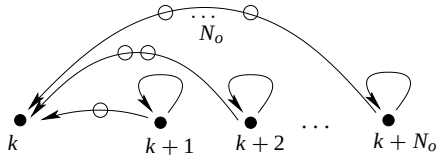

Figure 4.15: Components of $\widetilde{\underline{X}\_in}(k)$

$$\widetilde{A_{-i}}(k) = A_{-i}(k) \bigoplus_{j=i+1}^{N_o} \widetilde{A_{-j}}(k)\widetilde{A_0}^*(k+j)\widetilde{A_{j-i}}(k+i), \quad i = N_o, N_o-1, \cdots, 1, 0. \tag{4.32}$$

$$\begin{aligned} \widetilde{A}_i(k-i) &= A_i(k-i) \bigoplus_{j=i+1}^{N_o+1} \widetilde{A_{-(j-i)}}(k)\widetilde{A_0}^*(k-i+j)\widetilde{A_j}(k-i) \\ &= A_i(k-i) \bigoplus_{j=i+1}^{N_o+1} \widetilde{A_{-(j-i)}}(k)AKJ(k-i+j, k-i), \\ &\qquad\qquad i = 1, \cdots, N_o+1. \end{aligned} \tag{4.33}$$

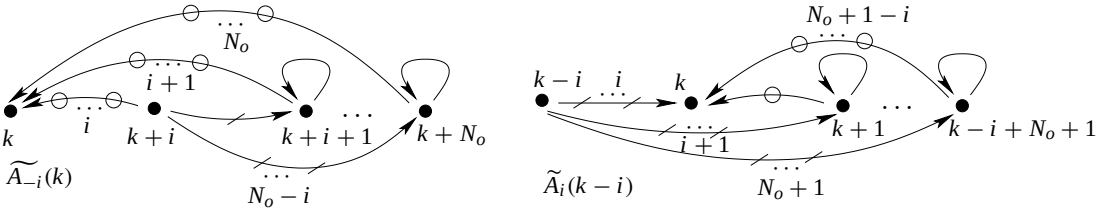

Figure 4.16: Components: $\widetilde{A_{-i}}(k)$ and $\widetilde{A}_i(k-i)$

Note that not every plan contains control arcs of every order between $-N_o$ and $N_o + 1$, thus the actual number of max-plus algebra addition ($\oplus$) in the above equations may be less than stated. For example, for cycle k_u, there is no negative order arc, i. e.

$$\widetilde{\underline{X}\_in}(k_u) = \underline{X}\_in(k_u), \quad \widetilde{A_{-i}}(k_u) = A_{-i}(k_u) = N, \quad \widetilde{A}_i(k-i) = A_i(k-i).$$

In conclusion, if the maximum order of cycle k is $mp(k)$, the minimum order of cycle k is $-mn(k)$, $(mn(k) \geq 0)$, the explicit model for an independent unit consisting of cycles $k = [k_1, \cdots, k_u]$ is

---

$$\underline{X}(k) = \widetilde{A_0}^*(k)\widetilde{\underline{X}\_in}(k) \bigoplus_{j=k-N_o-1}^{k-1} AKJ(k, j)\underline{X}(j), \tag{4.34}$$

where, for $k = [k_u, k_u - 1, \cdots, k_1]$,

$$\widetilde{\underline{X}\_in}(k) = \underline{X}\_in(k) \bigoplus_{i=1}^{mn(k)} \widetilde{A_{-i}}(k)\widetilde{A_0}^*(k+i)\widetilde{\underline{X}\_in}(k+i), \tag{4.35}$$

$$\widetilde{A_{-i}}(k) = A_{-i}(k) \bigoplus_{j=i+1}^{mn(k)} \widetilde{A_{-j}}(k)\widetilde{A_0}^*(k+j)\widetilde{A_{j-i}}(k+i), \quad i = mn(k), mn(k)-1, \cdots, 1, 0 \tag{4.36}$$

$$\widetilde{A_0}^*(k) = I \bigoplus_{t=1}^{n-1} \widetilde{A_0}^t(k), \tag{4.37}$$

$$\begin{aligned} \widetilde{A}_i(k-i) &= A_i(k-i) \bigoplus_{j=i+1}^{mp(k-i)} \widetilde{A_{-(j-i)}}(k)\widetilde{A_0}^*(k-i+j)\widetilde{A_j}(k-i) \\ &= A_i(k-i) \bigoplus_{j=i+1}^{mp(k-i)} \widetilde{A_{-(j-i)}}(k)AKJ(k-i+j, k-i), \qquad i = 1, \cdots, N_o+1 \end{aligned} \tag{4.38}$$

$$AKJ(k, j) = \widetilde{A_0}^*(k)\widetilde{A_{k-j}}(j),\ j < k. \tag{4.39}$$

Although $k \geq k_1$ in (4.38), $k - i$ could be less than k_1, i. e. a cycle $(k - i)$ could preceed the independent unit $[k_1, \cdots, k_u]$. It may have an indirect influence on cycle k via some negative order arcs inside the unit. In addition, although $i = 1, \cdots, N_o + 1$ in (4.38),

if $mp(k-i) < i$, there is no need to calculate $\widetilde{A}_i(k-i)$, i. e. $\widetilde{A}_i(k-i) = A_i(k-i)$. $AKJ(k, j)$ represents the overall influence of a previous cycle j to a subsequent cycle k.

For a cyclicly-running system with a required number (k_l) of cycles, the simulation is not based on each cycle, but based on each independent unit. An entire plan list assigned for the k_l cycles may consist of several independent units. The number (N_{uni}) of cycles contained in each unit may be different. The minimum possible value of N_{uni} could be 1 which means an independent non-negative order cycle. Not every non-negative order cycle is independent though, it could be a component cycle inside an independent unit.

As mentioned in Section 4.1, for a system with N_t trains, denote the number of cycles existing concurrently by N_c. Then the maximum value for N_c is N_t. At a certain time instance, if the first currently existing cycle is the one which starts an independent unit, and if $N_c < N_{uni}$, then some cycles in the unit are future cycles for which the corresponding initial vector $\underline{X}\_in(k)$ can be set to a ϵ-vector. To facilitate the backward simulation of the lower level, the states of these future cycles are also passed to the lower level.

4.7 Plan list optimisation

Within runtime, the upper level of the proposed hierarchical control architecture will automatically update the optimal plan list if unexpected events happen. The real time information on status of current events, $\underline{X}\_in(k)$ is generated by the C/D block which is described in Section 2.3. The optimisation objective could be the same one as (3.13) or (3.12), i. e. either to catch up with the timetable which is interrupted by the delay, or to finish the total required number of cycles as fast as possible. If there are several plan lists which optimise the same objective, the one which has the minimum sum of all arrival times (thus the minimum overall running time of trains) will be implemented via the lower level, i. e. we have a secondary optimisation problem with cost function

$$J_{secondary} = \min_{\substack{\text{all feasible} \\ \text{plan lists}}} \sum_{k=1}^{k_l} \sum_{i=1}^{N_t} Y_i(k), \tag{4.40}$$

where k_l is the required total number of cycles, N_t is the total number of trains and $Y_i(k)$ is the arrival time of train i in the k-th cycle.

Assume the delay occurs during cycle k_d. If several cycles exist concurrently with cycle k_d, the plans of previous cycles k ($k < k_d$) do not need to be changed when there is a fixed timetable. The search space only covers feasible plan lists which coincide with the

currently implemented plan list for ($k < k_d$). With the exception of (3.18) and (3.19), the procedure for reducing the search space discussed in Section 3.3 can also be applied here.

4.8 Lower level

With the event times specification provided by the upper level, it is possible to apply Corollary 2 proposed in Section 3.4 to get the corresponding LNET specification. In general, Corollary 2 means that if the influence of a vector of event times $Z_j (j = 1, 2, \ldots, m)$ on several other vectors of event times $V_i (i = 1, 2, \ldots, l)$ is described by

$$V_i = \bigoplus_{j=1}^{m} G_{ij} Z_j, \forall i = 1, 2, \ldots, l,$$

then the LNET specification for Z_j is

$$\overline{Z}_j = \bigoplus_{i=1}^{l} {}'((-G_{ij})^T \otimes' V_i).$$

Because of the existence of cycles with negative order arcs, the upper level simulation is based on independent units instead of cycles. An independent single cycle influences both itself and the subsequent cycles. A cycle inside an independent unit containing more than one cycle may influence both the subsequent cycles and—via the term $\underline{X}\_in(k)$ (see (4.34) and (4.35))—the previous cycles.

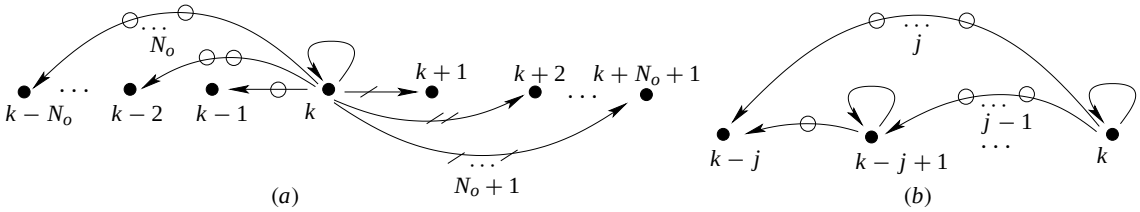

Figure 4.17: The influence of cycle k

As shown in Fig. 4.17(a), a cycle inside an independent unit can influence at most the following ($N_o + 1$) cycles and the previous N_o cycles. The numbers may vary according to different plans. Assume cycle k influences all its previous N_o cycles, then according to (4.35),

$$\widetilde{\underline{X}\_in}(k - j) = \underline{X}\_in(k - j) \bigoplus_{i=1}^{mn(k-j)} \widetilde{A_{-i}}(k - j)\widetilde{A_0}^*(k - j + i)\widetilde{\underline{X}\_in}(k - j + i),$$

i. e. the influence of $\underline{X}\_in(k)$ in cycle k on cycle $(k-j)$ is indicated not only by $\widetilde{A_{-j}}(k-j)\widetilde{A_0}^*(k)\underline{X}\_in(k)$, but also by $\widetilde{A_{-k_j}}(k-j)\widetilde{A_0}^*(k-j+k_j)\otimes\widetilde{\underline{X}\_in}(k-j+k_j)$, $(1 \le k_j \le j-1)$. This fact is due to the influence of $\underline{X}\_in(k)$ on $\widetilde{\underline{X}\_in}(k-j+k_j)$ as shown in Fig. 4.17(b). $\hat{A}_{-j}(k-j)$ (the "influence matrix") is used to indicate the total influence of $\underline{X}\_in(k)$ on cycle $(k-j)$, thus,

$$
\begin{aligned}
&\text{for } j=1, \qquad \hat{A}_{-1}(k-1) = \widetilde{A_{-1}}(k-1)\widetilde{A_0}^*(k),\\
&\text{for } j=2, \qquad \hat{A}_{-2}(k-2) = \widetilde{A_{-2}}(k-2)\widetilde{A_0}^*(k) \oplus \widetilde{A_{-1}}(k-2)\widetilde{A_0}^*(k-1)\hat{A}_{-1}(k-1),\\
&\qquad\qquad \vdots
\end{aligned}
$$

In general, for an independent unit with cycles $k = [k_u, k_u-1, \cdots, k_1]$, the influence matrix from cycle $(k+j)$ to cycle k is

$$\hat{A}_{-j}(k) = \widetilde{A_{-j}}(k)\widetilde{A_0}^*(k+j)\bigoplus_{i=1}^{j-1}\widetilde{A_{-i}}(k)\widetilde{A_0}^*(k+i)\hat{A}_{-(j-i)}(k+i),\ \ j=1,2,\ldots,mn(k). \tag{4.41}$$

Then,

$$\widetilde{\underline{X}\_in}(k) = \underline{X}\_in(k)\bigoplus_{j=1}^{mn(k)}\hat{A}_{-j}(k)\underline{X}\_in(k+j), \tag{4.42}$$

For $k = [k_1, \cdots, k_u]$, by inserting (4.42) into (4.34), we get

$$
\begin{aligned}
\underline{X}(k) &= \widetilde{A_0}^*(k)\widetilde{\underline{X}\_in}(k)\bigoplus_{j=k-N_o-1}^{k-1} AKJ(k,j)\underline{X}(j)\\
&= \widetilde{A_0}^*(k)\left[\underline{X}\_in(k)\bigoplus_{j=1}^{mn(k)}\hat{A}_{-j}(k)\underline{X}\_in(k+j)\right]\bigoplus_{j=k-N_o-1}^{k-1} AKJ(k,j)\underline{X}(j)\\
&= \widetilde{A_0}^*(k)\underline{X}\_in(k)\bigoplus_{j=1}^{mn(k)}\widehat{A_{-j}}(k)\underline{X}\_in(k+j)\bigoplus_{j=k-N_o-1}^{k-1} AKJ(k,j)\underline{X}(j)
\end{aligned}
$$

where $\widehat{A_{-j}}(k) = \widetilde{A_0}^*(k)\hat{A}_{-j}(k)$.

Seen from another point of view, cycle k influences (apart from itself) subsequent cycles via

$$
\begin{aligned}
\underline{X}(k+1) &= \cdots \oplus AKJ(k+1,k)\underline{X}(k) \ \oplus \cdots,\\
&\ \ \vdots\\
\underline{X}(k+mp(k)) &= \cdots \oplus AKJ(k+mp(k),k)\underline{X}(k) \ \oplus \cdots,
\end{aligned}
$$

and may influence previous cycles via

$$\begin{aligned}
\underline{X}(k-1) &= \cdots \oplus \widehat{A_{-1}}(k-1)\underline{X}_in(k) \ \oplus \cdots, \\
&\vdots \\
\underline{X}(k-k1) &= \cdots \oplus \widehat{A_{-k1}}(k-k1)\underline{X}_in(k) \ \oplus \cdots,
\end{aligned}$$

where $k1$ satisfies $1 \leq k1 \leq N_o$ and $k-k1 \geq k_1$. Note that for cycle j ($1 \leq j \leq k1$) to be influenced by cycle k, it is necessary that $mn(k-j) \geq j$, i. e. cycle $(k-j)$ has negative order arcs pointing from cycle k.

Considering all influenced cycles listed above, the LNET for cycle k can be derived directly from Corollary 2. For train m, in cycle k, LNET should not violate the EPET schedule of the other trains and the EPET schedule of the final event of each train. In addition, the EPET schedules of the corresponding sequential cycles and the corresponding previous cycles should also be met.

Suppose that for a certain time instance, there are N_c concurrently existing cycles with cycle indices $k = [1, \cdots, N_c]$. Note that the cycles are not necessarily required to be in the same independent unit. They may belong to several sequential independent units Iu_i with $i = 1, \ldots, n_{Iu}$, where n_{Iu} is the number of independent units within the N_c concurrently existing cycles. Suppose unit Iu_i contains cycle $[ki_1, \cdots, ki_{u_i}]$. For train m, the LNET specifications of each cycle $k = [ki_{u_i}, ki_{u_i}-1, \cdots, ki_1](i = n_{Iu}, \ldots, 1)$ are

$$\begin{aligned}
\overline{X}_m(k) &= (-(\widetilde{A_0}^*(k))^T)\underline{XR}_m(k) \bigoplus_{j=1}^{mp(k)}{}'(-(AKJ(k+j,k))^T)\underline{X}(k+j) \ \oplus' \\
&\quad \bigoplus_{j=1}^{k1}{}'(-(\widehat{A_{-j}}(k-j))^T)\underline{X}(k-j), \qquad (4.43)
\end{aligned}$$

where

$$[(\underline{XR}_m)_i(k)] = \begin{cases} [\underline{x}_i(k)], & i \in QS(k) \text{ or } i \notin PS_m(k) \\ +\infty, & \text{otherwise.} \end{cases} \qquad (4.44)$$

Note that $QS(k)$ is the event index set for all final events of trains in cycle k and $PS_m(k)$ is the index set for all events related to train m in cycle k.

With EPET and LNET specifications for train m, the velocity signal can then be optimised locally by exploiting the remaining degrees of freedom as discussed before.

4.9 Simulation

For the three-train network shown in Fig. 2.19, the maximum number of concurrently existing cycles is 3 when each train belongs to a different cycle. The lowest order of control arcs is $N_o = N_t - 2 = 1$ while the maximum order is $N_o + 1 = 2$. With the introduction of negative order arcs on single-line track segments OM_1 and OM_2, besides the original three non-negative order plans, there additionally are 2 feasible negative order plans shown in Fig. 4.18 and Fig. 4.20, respectively.

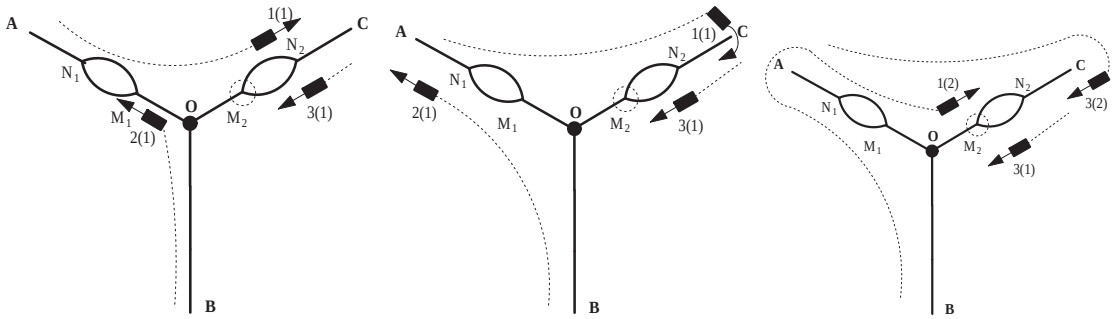

Figure 4.18: Plan 4 (OM_1: train1 train2, OM_2: train1_cycle$k+1$ train3_cycle$k$)

In plan 4, after train 2 in cycle 1, denoted by 2(1), arrives at its destination, it becomes

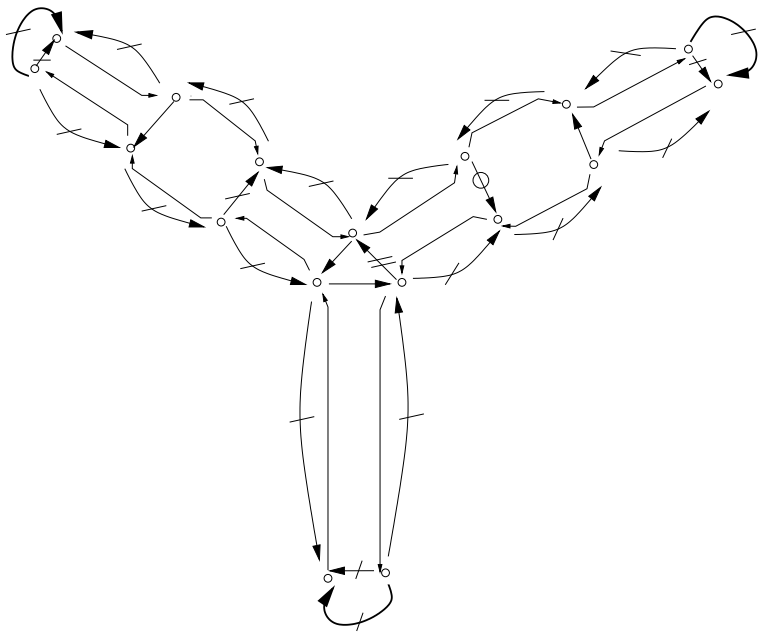

Figure 4.19: Directed graph for plan 4

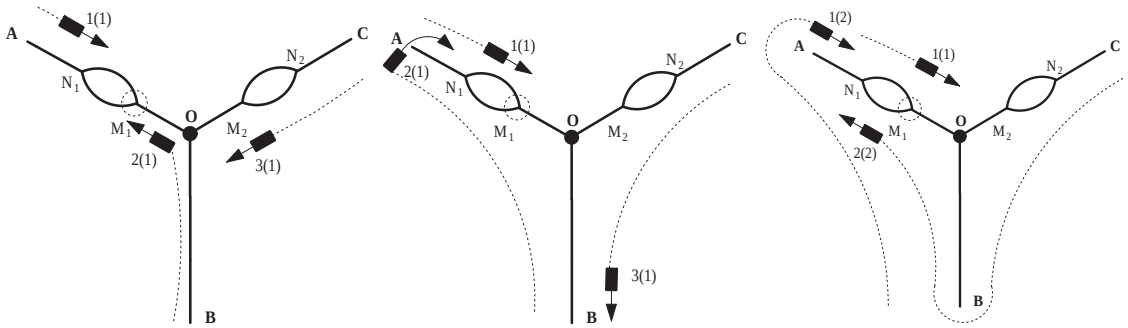

Figure 4.20: Plan 5 (OM_1: train2_cycle$k+1$ train1_cycle$k$, OM_2: train3 train1)

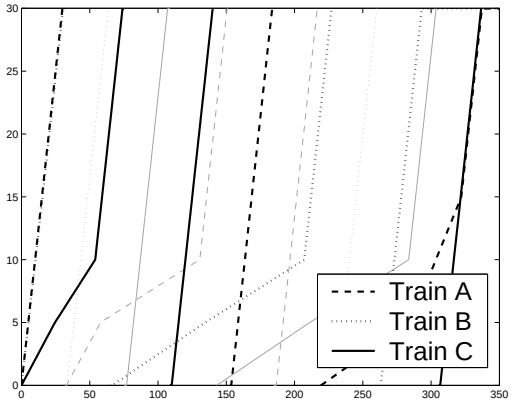

Figure 4.21: Plan 4

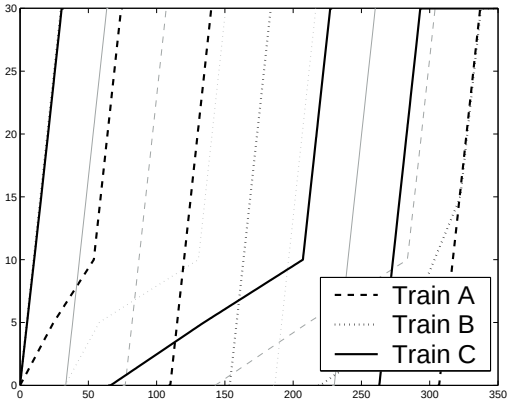

Figure 4.22: Plan 5

train 1 in cycle 2, denoted 1(2), and starts a new journey in which it passes train 3 in cycle 1 at point M_2. Fig. 4.19 shows the directed graph for plan 4. The dotted arcs are the corresponding control arcs on single-line track segments OM_1 and OM_2. These arcs distinguish plan 4 from other plans. In plan 5, after train 3 in cycle 1 arrives at its destination, it becomes train 2 in cycle 2 and passes train 1 in cycle 1 at M_1. Fig. 4.21 and Fig. 4.22 show the 5 cycles simulation results for plan 4 and plan 5, respectively. Note that in these figures, physical trains are represented. Train A starts as train 1 in cycle 1, train B as train 2 in cycle 1, and train C as train 3 in cycle 1.

If there are no disturbances, it takes plan 4 or plan 5 more time to finish all required cycles than the non-negative order plan 1. However, during runtime, if an unexpected delay happens, plan 4 or plan 5 could be better in terms of minimal final arrival time. In Fig. 4.23, the system initially operates based on the plan list [1 1 1 1 1] implying that a train in a later cycle may not enter a shared track segment until all trains in previous cycles have left it. During runtime, the system sticks to the original plan list although train 1 in cycle 1 (i. e. the physical train A) is blocked on the segment M_1N_1 from time

unit 12 to 50. Thus, train 2 in cycle 2 (i. e. the physical train C) has to slow down to wait until train 1 in cycle 1 leaves OM_1. It takes a total of 227 time units to finish all 5 cycles. If the system changes to plan list [5 3 2 1 2] (Fig. 4.24) after blocking happens, train 2 in cycle 2 (the physical train C) occupies OM_1 before train 1 in cycle 1 enters it. In this case, the required 5 cycles finish in 217 time units. In fact, the new plan list is the one that optimises objectives (3.12) and (4.40).

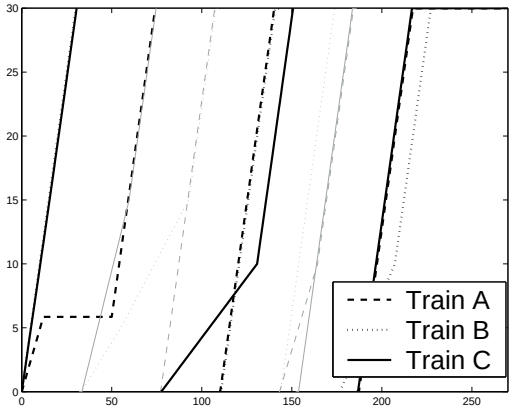

Figure 4.23: Plan list [1 1 1 1 1]

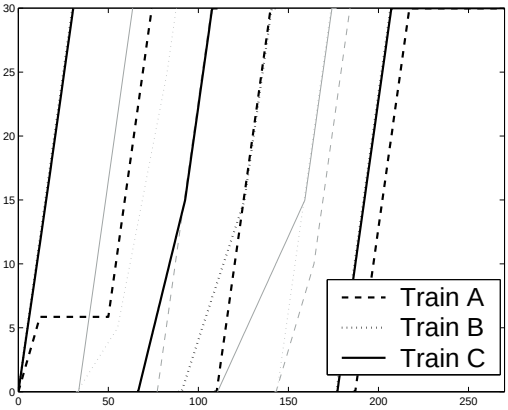

Figure 4.24: Plan list [5 3 2 1 2]

We now investigate the the strictly cyclic case, where a timetable is fixed. All trains start their trips according to the timetable and a plan strictly repeats itself in every cycle. For example, in Fig. 4.25, plan 2 is repeated within 5 cycles. In each cycle k, train 1 (in cycle k) and train 3 (in cycle k) pass each other at M_2. In Fig. 4.26, train 3 is delayed from time unit 18 to 67 and it becomes impossible for trains to meet the timetable. To recover the timetable as fast as possible, the supervisory block on the upper level switches the original strictly cyclic plan 2 to plan list [4 2 1 2 2]. In the new plan list, train 1 in cycle 2 (physical train B) occupies the segment OM_2 before train 3 in cycle 1 (physical train C) enters it. With the new plan list, the timetable is recovered at cycle 5, and from this cycle on, the system is back to normal working status and is able to repeat plan 2 according to the timetable. For comparison, Fig. 4.27 shows the situation where nothing has been done to handle the unexpected delay, clearly the timetable cannot be recovered within 5 cycles.

4.10 Conclusion

This chapter discusses the control of cyclic systems in a more general setting: it allows that events in a later cycle happen before events in previous cycles. Such a feature is

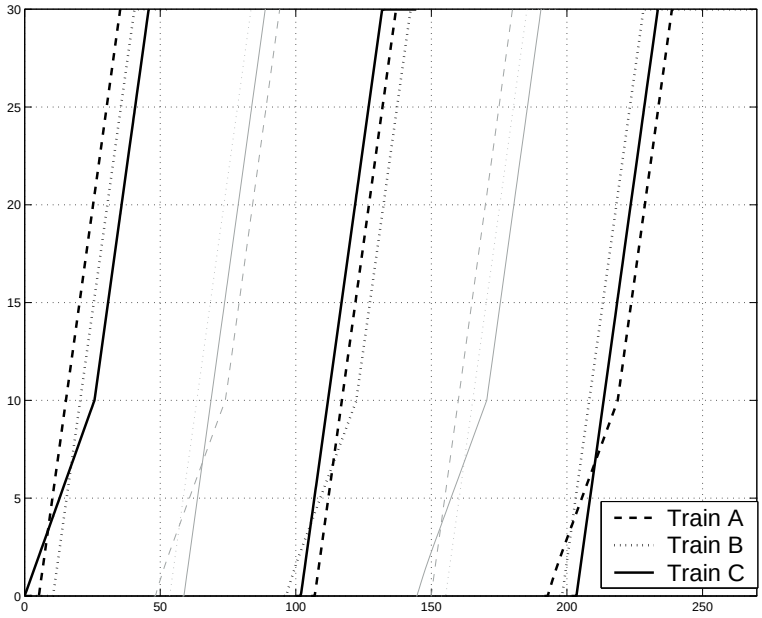

Figure 4.25: Strictly cyclic system under plan 2without disturbance

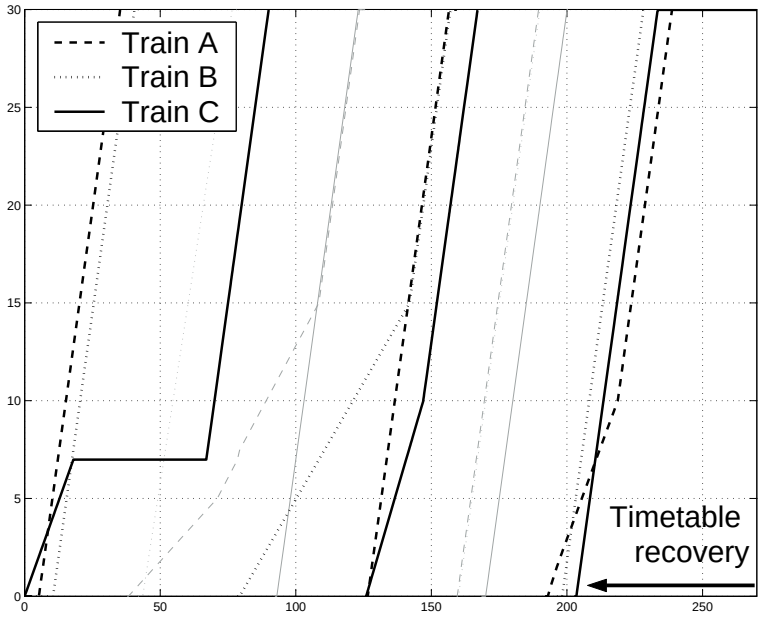

Figure 4.26: Change of plan after disturbance, new plan list [4 2 1 2 2], timetable is recovered at cycle 5

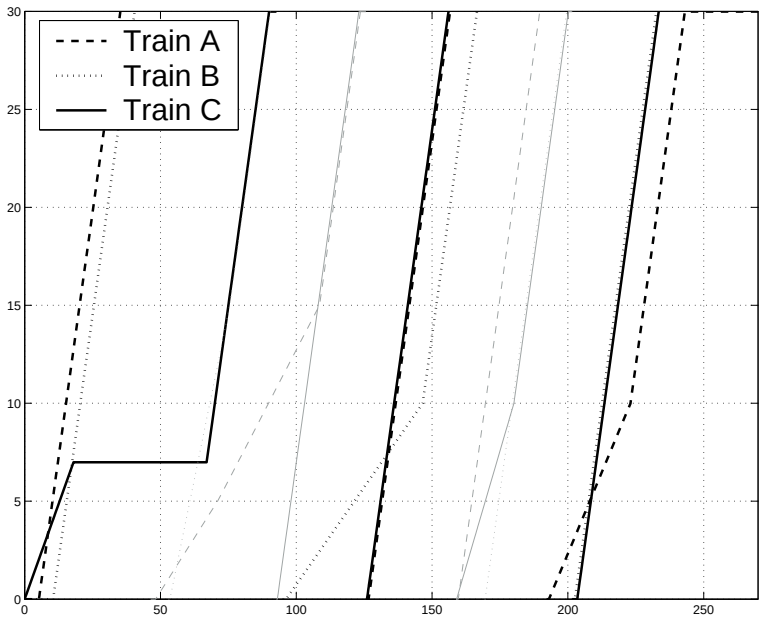

Figure 4.27: No change of plan, no timetable recovery within 5 cycles

represented by negative order control arcs. By introducing the concept of negative order arcs, there are more options to express sequences of events on a shared single resource. To check feasibility of the combinations of control arcs on different single-line track segments, the orders of circuits are calculated. Two methods of finding feasible plan lists are discussed in detail. A typical way to analyse and control strictly cyclic systems with negative order arcs is based on the idea of cycle shifting so that there are no negative order system matrices in the system model. Unfortunately, this method is not applicable for general systems which have negative order control arcs. Additionally, cycle shifting makes it impossible for the system to react to unexpected events. Facing such difficulties, an independent unit based model is proposed in this chapter to solve these problems. A train track network is used to show the effectiveness of the resulting control strategy.

It should be pointed out that although the control mechanism for this more general setting is useful in many fields, it is likely to run into complexity problems for large systems. With the introducing of negative order control arcs, the number of feasible plans “explodes” for large systems.

Chapter 5

Case study: high throughput screening plant

In the previous chapters, the proposed hierarchical control structure has been applied to railway transportation systems by using train-track examples. It can also be used in other applications such as manufacturing systems, chemical batch plants, and High Throughput Screening (HTS) plants.

A typical control task for an HTS system is to adjust timing parameters of some activities so that the throughput is maximised. High throughput screening plants are used, e. g. in the pharmaceutical industries to analyse a large number of substances. It is often required that HTS systems are operated in a strictly cyclic way. Then, throughput maximisation is equivalent to minimisation of the cycle time. A strictly cyclic operation pattern implies of course that there is no ability of automatically reacting to unexpected delays. However, in reality, unexpected delays are likely to happen and they decrease the throughput of the HTS plant. Hence, an automatic online reaction ability embedded in the adopted control mechanism is strongly desired in order to cope with a temporally changing environment. In this context, the proposed hierarchical control structure is an attractive option for improving the HTS operating performance.

In this chapter, the proposed control structure is applied to a specific scheduling problem derived from an HTS application [69]. The chapter is arranged as follows: Section 5.1 provides a detailed case description. We are especially interested in the similarities between a train-track system and the studied example. In Section 5.2, it is pointed out how the solution for the train track example from the previous chapters can be carried over to

the HTS example at hand. Finally, conclusions are given in Section 5.3.

5.1 Problem description

A typical HTS plant contains several resources such as incubators, readers, transport devices etc. Substances are aggregated in batches, which are realised by microplates. Each batch undergoes a sequence of activities, and each activity allocates a resource. Hence, the batches occupy the resources according to a time scheme which is usually identical for each single batch. It may well be possible that a single batch occupies the same resource several times.

The starting time o_i (or the ending time r_i) of activity i is not necessarily the entering time (or the leaving time) of the corresponding batch on the resource because there may exist either pre-processing or post-processing time requirements. Fig. 5.1 shows the time scheme of a specific single batch. There are 3 different resources and 6 activities, 4 of which are related to resource 1. It is inspired by the example in [69]. In [69], the task is to determine certain timing parameters (such as the one showing the time relation between r_2 and o_3) and the event sequences (under normal conditions) so that the cycle time is minimal. Here, we assume that the corresponding timing parameters have been fixed at their optimal values, and our task is to find optimal control for the resulting system under disruptive conditions.

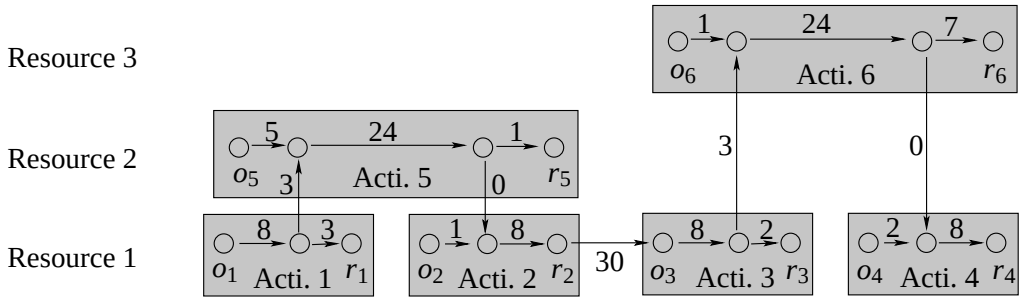

Figure 5.1: Time scheme for a single batch

For a single batch shown in Fig. 5.1, the 6 activities are performed in the following order: activity 1 → activity 5 → activity 2 → activity 3 → activity 6 → activity 4. At a given time instance, there could be more than one batch existing in the system, e. g. there could be three batches, with each batch locating one resource. When compared to the train-track problems in the previous chapters, a "batch" corresponds to a train and a

resource to a track segment. We now translate the HTS problem into a corresponding train-track network. For a given single batch time scheme, there may exist different corresponding train-track systems involving different numbers of trains. For example, the six train system in Fig. 5.2(a) and the four train system in Fig. 5.2(b) both correspond to the time scheme shown in Fig. 5.1 with the exception of the pre- and postprocessing times. Note that the additional arcs at the bottom of Fig. 5.2 (a) and (b) reflect the additional requirement that a new batch may start 3 time units after the previous is finished.

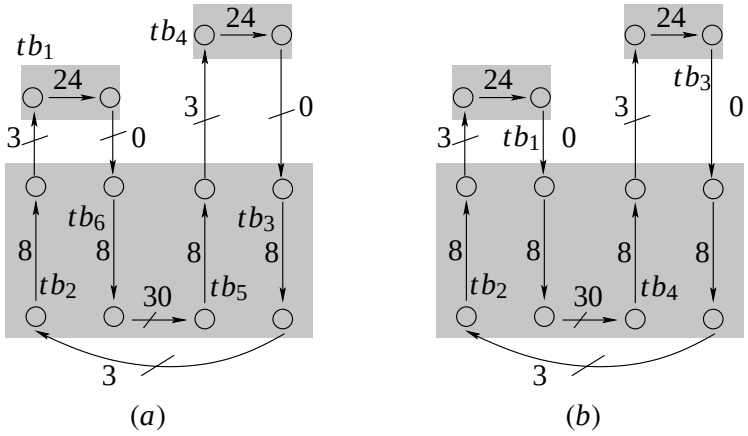

Figure 5.2: Corresponding train-track systems

Fig. 5.2(a) consists of 6 different tbs ("train-batches"), each of them has a "trip" to take. Four tbs, i. e. tb_2, tb_3, tb_5 and tb_6, may not exist concurrently since all of them use the same resource. Moreover, all "trips" of tbs consist of only two events. Correspondingly, there is only one travelling arc (one activity) on each trip. In previous chapters, for a train-track system with N_t trains, we always assumed that it is possible for all N_t different trains, each in a different cycle, to take the same route concurrently. This is because in our train-track examples, the number of travelling arcs on the route is usually larger than the number of trains. For such systems, the maximum possible cycle difference of trains taking different routes is as high as $(N_t - 2)$. But for a train-track system like Fig. 5.2(a), it is impossible for more than one train to travel on the same route at a same time. Considering the small number of activities on a trip, the maximum number of cycles existing concurrently on the same route is

$$N_{mcr} = \min(N_a, N_t), \tag{5.1}$$

where N_a is the maximum number of activities (or zero order travelling arcs) on any route. For Fig. 5.2(a), $N_{mcr} = 1$, thus there is no train in a different cycle taking the same

trip concurrently. However, this does not always mean that there is no need to consider non-zero order control arcs. If tb_1 and tb_2 of cycle 1 in Fig. 5.2(a) start operating at the same time, after tb_2 finishes its trip, it cannot start its next cycle trip because the required resource is still occupied by tb_1. But if there is a buffer between resource 1 and resource 2, tb_2 can leave resource 1 so that other tbs, including some tb in a later cycle, can use it. For example, before tb_5 starts its first cycle trip, tb_3 can start cycle 2 directly after its first cycle. In fact, if there is no constraint on the number of buffers, there is no limitation on the order of control arcs. In this chapter, in order to clarify the relation between HTS plants and train-track systems, we consider systems in which there is no negative order control arc.

For the 6-tb system shown in Fig. 5.2(a), the different combinations of control arcs on resource 1 result in different plans. Specifically, the possible combinations of control arcs will provide 24 feasible plans. A 4-tb system shown in Fig. 5.2(b) has the same 24 feasible plans since there is no additional constraint when compared to the 6-tb system. Although the number of tb (i. e. N_t) contained in a cycle decreases, it will not lower the performance of the HTS plant. In the 4-tb system, a tb still represents a single batch. However, instead of the fact that each tb only has one activity in a cycle, now there are two activities to be finished on the "trips" of tb_1 and tb_3, respectively. The different time interdependencies inside a cycle of Fig. 5.2 (a) and (b) both correspond to the single batch's time scheme shown in Fig. 5.1.

Note that, because the 4-tb and the 6-tb systems in Fig. 5.2 correspond to the HTS example in Fig. 5.1, they retain all degrees of freedom. Specifically, in both cases we will get 24 feasible plans corresponding to 24 ways of constructing a cyclic scheme out of the single batch time scheme. Obviously, for the original single batch time scheme, there could be other corresponding train-track systems which provide the same number of feasible plans. For example, there must be at least one 5-tb system that has the same 24 plans as the two systems in Fig. 5.2. However, the number of tbs (N_t) may not be too small — it is clear that the resources will be not fully used if N_t is less than the number of resources. Furthermore, compared to the systems in Fig. 5.2, there should not exist additional control arcs as they imply a loss in the degree of freedom. For example, if we neglect the time distance between r_2 and o_3, consider another similar train-track system with 3 tbs, i. e. a system having the same tb_2 and tb_3 as shown in Fig. 5.2(b) and a new tb_1 containing 3 activities, two of which are the same activities as the ones of tb_1 in Fig. 5.2(b), the remaining activity of tb_1 as the activity of tb_4 in Fig. 5.2(b). Such a system has less degrees of freedom.

There are three activities on the trip of the new tb_1. Unlike normal train-track systems, in this 3-tb system, since two out of three activities (called *resource repeated activities*) are allocated sequentially on the same resource, the later activity (the one corresponding to tb_4 of (b)) in cycle k should always happen before the earlier activity (the one corresponding to tb_6 of (a)) in cycle $(k+1)$. The possible negative order control arcs do not exist. For the original time scheme of the single batch, this 3-tb system loses some possible plans.

In the following, we apply the proposed hierarchical control structure to the 4-tb system to demonstrate the control results for the HTS example.

5.2 Control of an HTS plant

For an HTS problem with the single batch time scheme shown in Fig. 5.1, the corresponding train-track systems (a) and (b) shown in Fig. 5.2 have the same operating sequences, i. e. the same feasible plans. In the following, we consider the control on the 4-tb system.

5.2.1 Running mode: strictly cyclic vs. non-strictly cyclic

In many cases, the strictly cyclic-running mode is preferred for its simplicity. There are totally 24 feasible plans, each corresponding to a specific A-matrix. A-matrix with the minimal max-plus algebra eigenvalue allow minimal cycle time and therefore maximal throughput. Fig. 5.3 shows the tb trajectories of the corresponding optimal plan for the 4-tb system and the corresponding Gantt chart which shows the cyclic schedule for the original HTS problem. The numbers in the Gantt chart are the indices of different batches. The pre- and post-processing times of the HTS problem are considered in the Gantt chart.

The 4-tb system operates for 5 cycles and tb trajectories are shown in black lines or gray lines for different cycles, respectively. In the Gantt chart, the corresponding cycles are shown in gray or white. A cycle contains 6 different activities of 4 different tbs. On the other hand, from the Gantt chart of Fig. 5.3, it can be seen clearly that a single batch consisting of 6 activities covers 4 cycles. For example, the first activity of batch 4 happens on resource 1 in the first cycle. The second and the third activities of batch 4 both happen in cycle 2 (also see tb_2 trajectory of cycle 2). In cycle 3, batch 4 has its fourth activity which is allocated on resource 1. Finally, on resource 3 and resource 1 respectively, the last 2 activities of batch 4 are in cycle 4. After batch 4 has been finished, in cycle 5, a new

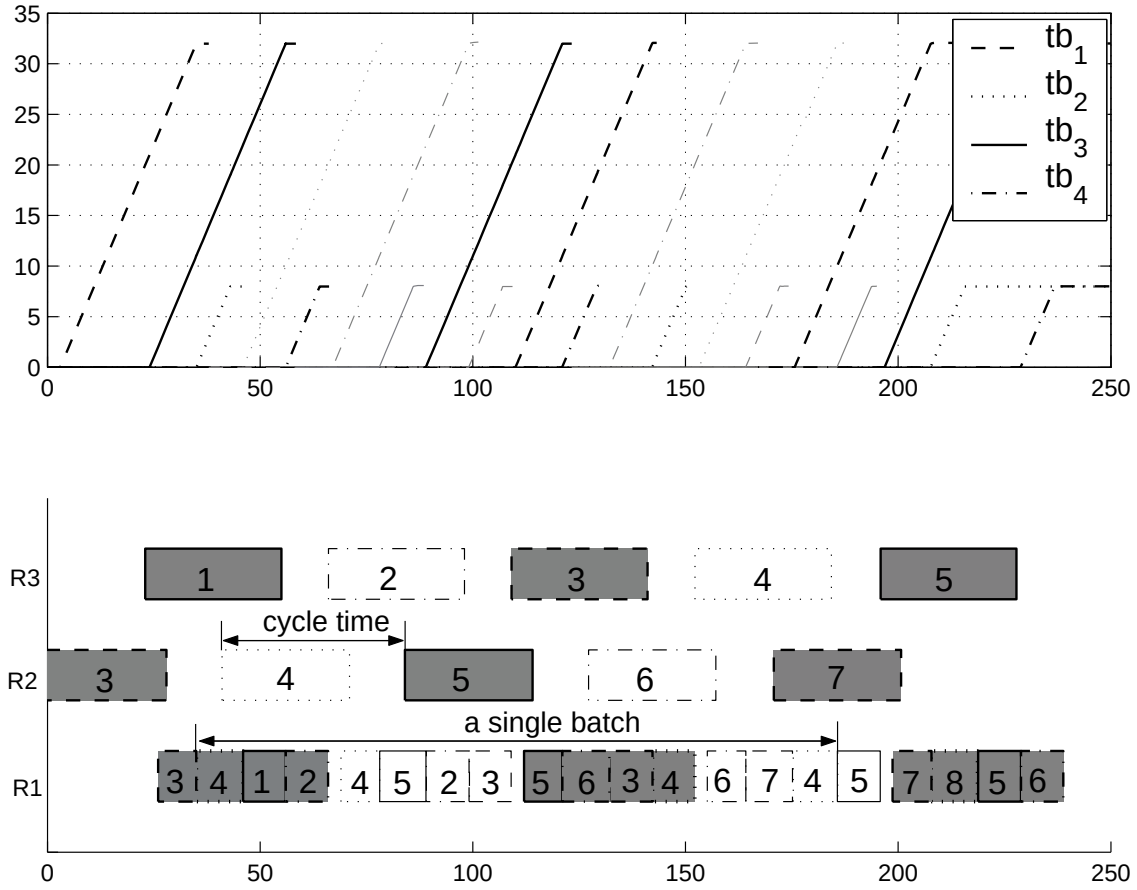

Figure 5.3: Strictly cyclic plan for maximal throughput

batch 8 starts with its first activity occupying resource 1. Generally, if a single batch i has been finished, a new batch $(i + 4)$ will start provided that the system continues running.

Although there is no negative order control arc in the 4-tb system and the activities in a subsequent cycle may not occupy a resource before the ones in all previous cycles have released it, for a specific resource, the activities of a later batches may happen before the activities of a previous batch happen. For example, in Fig. 5.3 on resource 1 of cycle 1, the activity of batch 1 happens after the activities of batch 4 and batch 3 have finished.

In strictly cyclic mode, not only the plans are repeated strictly, the batches are also strictly cyclic operated. For example, in each cycle both resource 2 and resource 3 are occupied by a new batch and the batch indices are increased monotonically. On resource 1, which is shared by 4 kinds of activities, the batch index for each kind of activity is also monotonically increased, see Table 5.1. From left to right, the order of the activities in the table shows their sequences in a single batch.

	R1, acti. 1	R2, acti. 5	R1, acti. 2	R1, acti. 3	R3, acti. 6	R1, acti. 4
cycle 1	batch 4	batch 3	batch 3	batch 2	batch 1	batch 1
cycle 2	batch 5	batch 4	batch 4	batch 3	batch 2	batch 2
cycle 3	batch 6	batch 5	batch 5	batch 4	batch 3	batch 3
cycle 4	batch 7	batch 6	batch 6	batch 5	batch 4	batch 4
cycle 5	batch 8	batch 7	batch 7	batch 6	batch 5	batch 5

Table 5.1: Cycles and batches

As illustrated in Fig. 5.3, there are several cycles as well as several other active batches during the time interval needed for a single batch. The resources (especially resource 1) are used as fully as possible, therefore the cycle time is less than the time a single batch requires. This strictly cyclic plan has the maximum throughput. However, there is still some free time for this resource that can be used if we give up the requirement of strict cyclicity.

If we do not stick to the strictly cyclic mode, it is possible to make use of the resources in a more efficient way. For a total of 5 cycles, with the objective function (3.12), i. e.

$$J = \min_{\substack{\text{all feasible} \\ \text{plan lists}}} \Big(\bigoplus_{i} Y_i(5) \Big),$$

the optimal plan list is [20 9 19 4 13]. It makes full use of resource 1, see Fig. 5.4. The indices of the batches are still monotonically increased not only on the non-shared resources 2 and 3 but also on the shared resource 1 for each kind of activities. The relations of cycles and batches shown in Table 5.1 still hold. However, unlike in Fig. 5.3, the order of different kinds of activities on resource 1 is not always the same for different cycles.

For the optimal strictly cyclic mode in the Gantt chart of Fig. 5.3, the maximum number of batches existing at a certain time instance is only 2, i. e. at any time, the number of batches using all 3 resources is no more than 2. Note that this fact does not depend on the number of tbs contained in the system.

In the Gantt chart of Fig. 5.4, the maximum number of batches existing at a certain time instance is 3. The operating mode with the plan list [20 9 19 4 13] makes a better use of resources. For both resource 2 and 3, during the processing of 5 different batches, the free time is less than that of Fig. 5.3. Although there is still free time for resource 2 or resource 3, it can not be reduced further because there is no free time for resource 1.

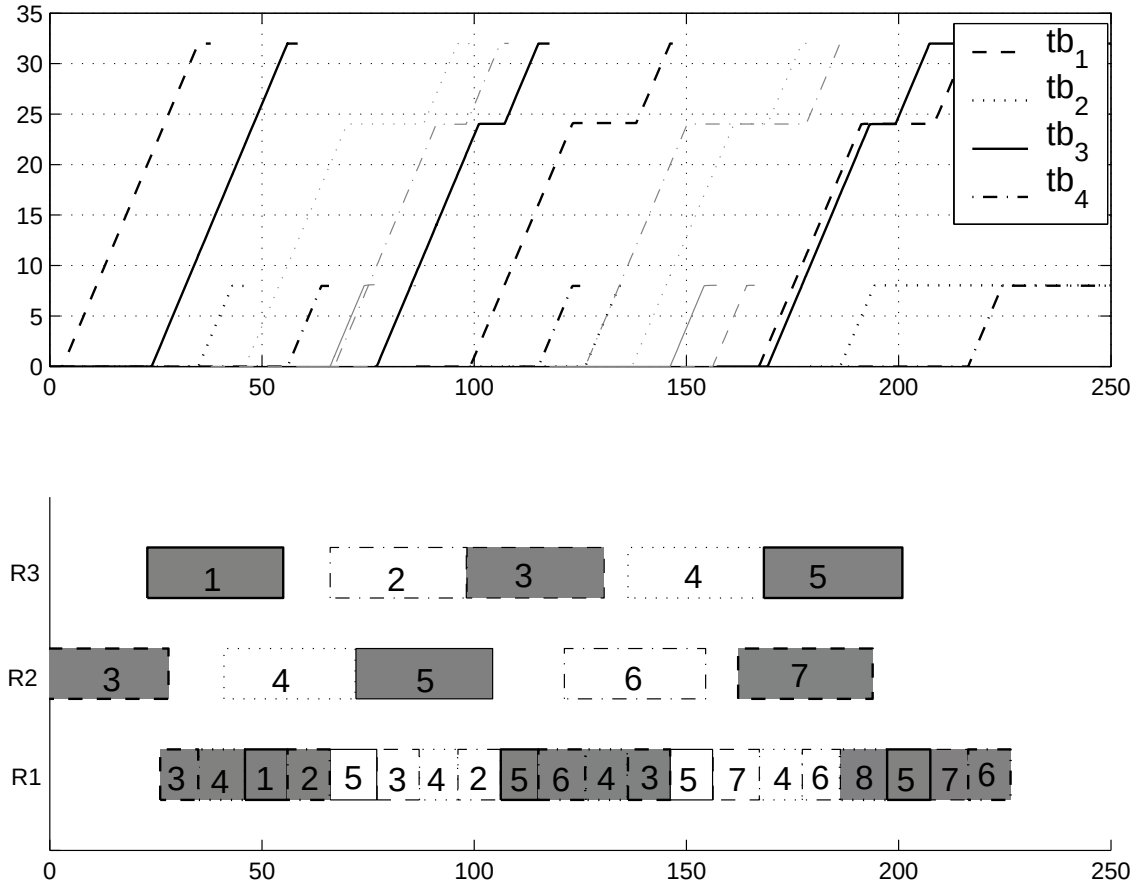

Figure 5.4: Resource 1 is fully used, plan list [20 9 19 4 13]

5.2.2 Reacting to disturbances

For the strictly cyclic mode of the optimal plan shown in Fig. 5.3, the cycle time is set to the eigenvalue of the system's A-matrix. Based on a corresponding eigenvector, a timetable can be fixed. The system is operated according to the fixed timetable. If an unexpected delay occurs and interrupts the timetable, the strictly cyclic system has no ability to recover from the delay.

Normally, without any delay, the system can finish 7 cycles in 322 time units. If some batch is delayed unexpectedly, for example, in cycle 2, batch 2 (i. e. tb_4 in cycle 2) is delayed on resource 3 for 10 time unites (from 70 to 80), the strictly cyclic system finishes 7 cycles in 332 time unites as depicted in Fig. 5.5.

After delay occurs, the supervisory block updates the plan list to [20 20 9 20 9 19 12] with the objective function (3.12) in order to finish 7 cycles as soon as possible. As shown in

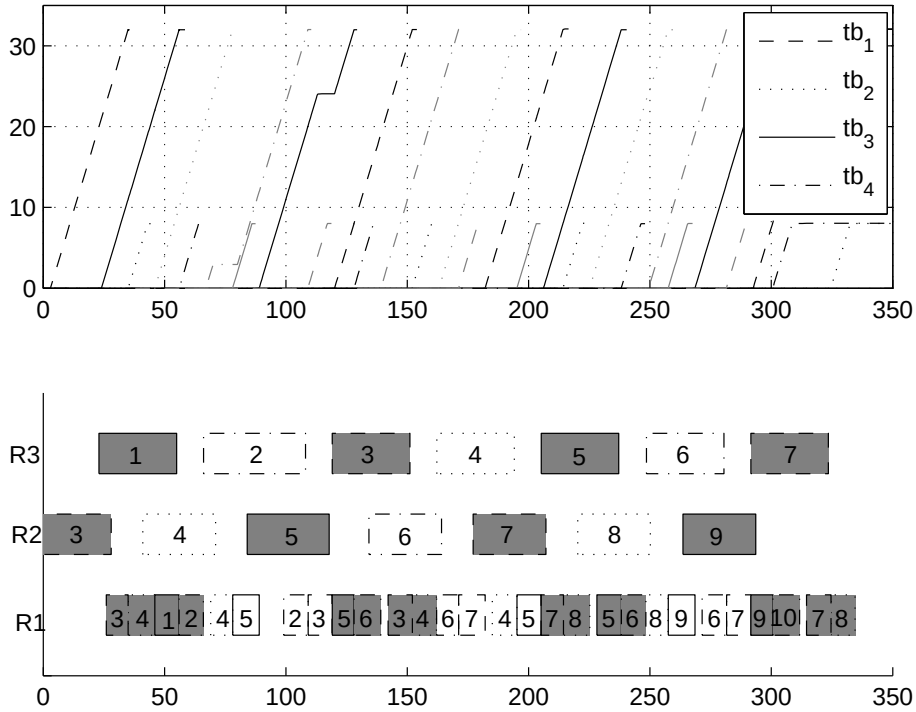

Figure 5.5: Strictly cyclic plan with delay

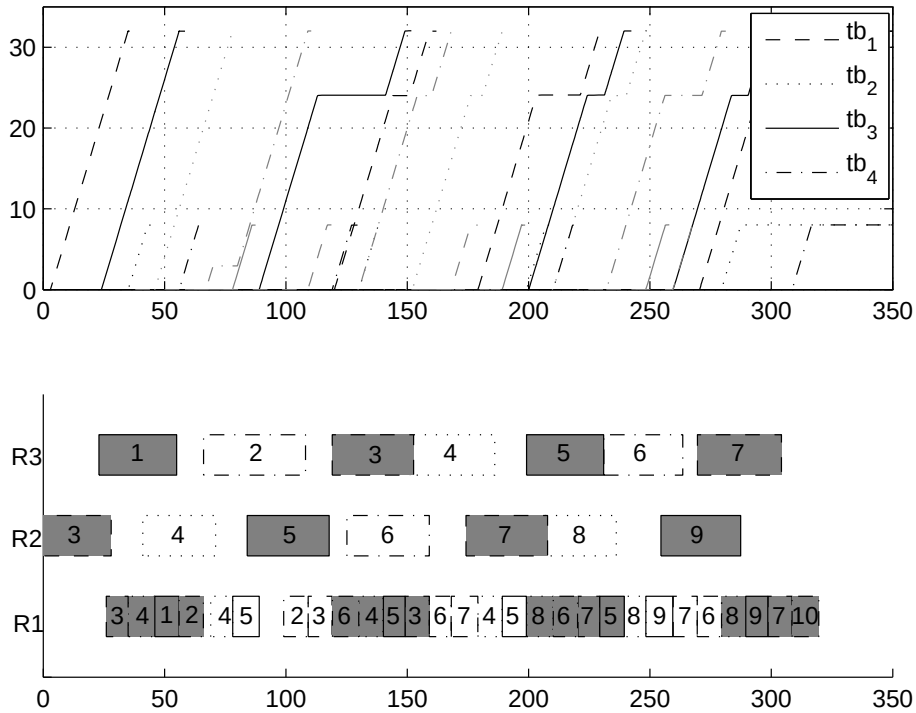

Figure 5.6: New plan list: [20 20 9 20 9 19 12]

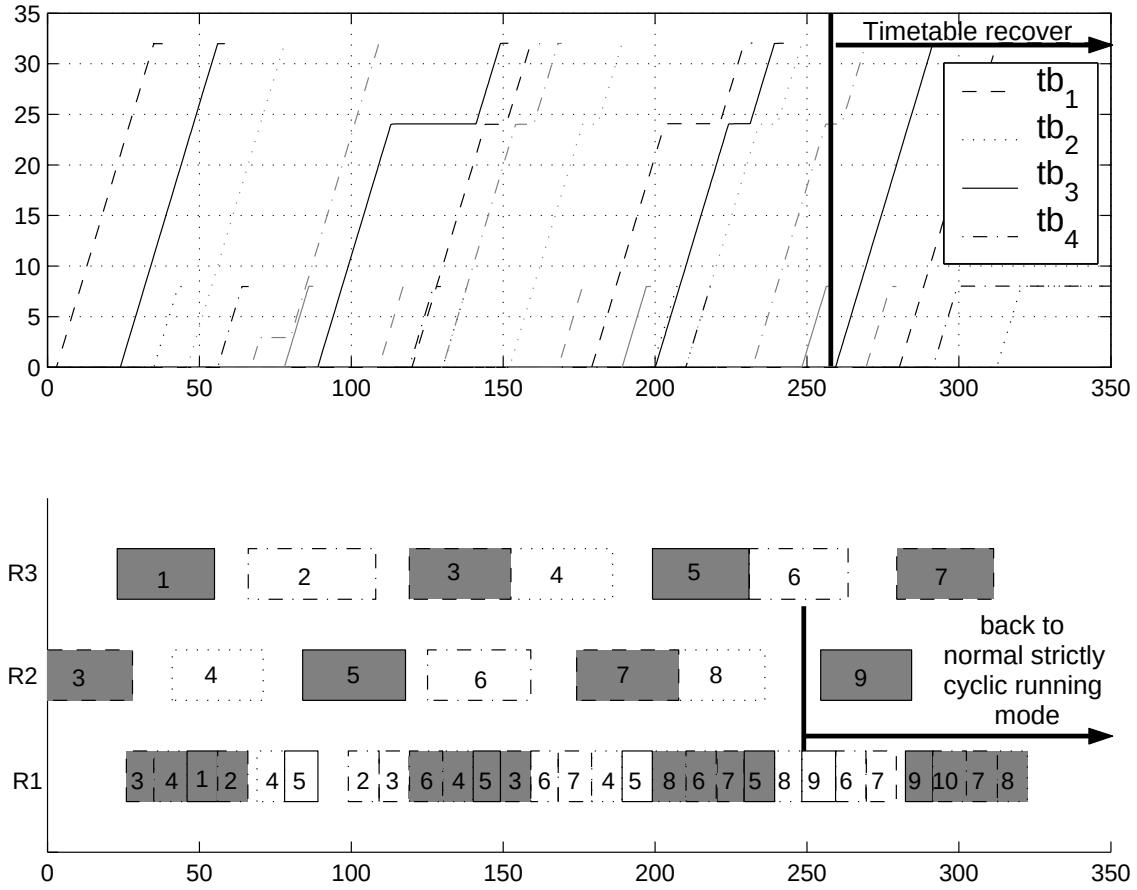

Figure 5.7: Timetable recover

Fig. 5.6, the system finishes 7 cycles in 316 time units and resource 1 is fully used with the new plan list.

Although the strictly cyclic mode itself cannot recover the timetable, it is possible to eliminate the influence of the delay and to recover the timetable with the help of an upper level supervisory block in the proposed control structure. The cost function (3.13) (earliest recovery of timetable) is being minimised by the plan list [20 20 9 20 9 20 20]. Then timetable can be recovered in cycle 7 (Fig. 5.7). And as in the normal case (i. e. strictly cyclic mode without unexpected delay), all 7 cycles are finished in 322 time units. If the system is expected to run for a large number of cycles, all later cycles can operate according to the timetable. Note that for the original HTS plant, during the first 6 cycles, it is possible for some activities of some single batches to start operating according to the timetable. But this does not mean that the entire plant is able to return to the normal operating orbit, because later on there are still other activities which do not start on time.

In addition, although the timetable is recovered at cycle 7, which contains activities of 4 batches (batch 7∼10), at least two batches (batch 7 and 8) do not operate normally from an entire single batch point of view. In cycle 7, it is batch 10 whose first activity belongs to this cycle, therefore only the later batches (including batch 10) are ensured to operate as in normal situation. Moreover, since in cycle 6, tb_3, which corresponds to the first activity of batch 9, is already able to start operating according to the fixed timetable, the batch plant actually returns to the normal strictly cyclic mode with batch 9.

5.3 Conclusion

In this chapter, we examined the applications of the proposed hierarchical control structure to an HTS plant processing area. To control the HTS plant, a single batch consisting of several sequential activities can be considered, for example, as a "train" (tb) travelling in several sequential cycles while in each cycle there is only one resource as well as one activity for the "train". N_t, i. e. the number of "trains", is identical to the number of activities contained in a single batch. It is also possible to consider the HTS plant as another train-track system with less "trains". In general, the number of tbs should be no less than the number of the resources.

For the control of HTS plants, even when there is no negative order control arc, it is still possible for activities in latter batches to happen before the activities in previous batches do, for a batch is not the same as a cycle.

In many cases, the operating schemes of HTS plants are strictly cyclic, with the event sequences specified by the plan which provides the maximum throughput. Sometimes with this plan, the resources are still not fully used. Instead, a list of different plans may make a better use of the resources. The supervisory block of the control structure also makes it possible for the HTS plant to reduce or eliminate the influence of unexpected delays. These effects are illustrated by simulation results.

Chapter 6

Conclusion

6.1 Summary

In this work, a hierarchical control structure is proposed for the control of discrete event systems. Application fields are traffic systems, especially train-track networks, and High Throughput Screening plants. It can be used for DESs with or without cyclicly repeated features. Given the topology of the DES, i. e. the information about resources and users, competition and cooperation amongst users and cycles are automatically organised by the proposed control structure so that the system provides optimal operation even under disruptive conditions. The system can return to the normal strictly cyclic schedule in minimum time after being delayed unexpectedly. The new approach also makes it possible to control a specific DES with non-strictly cyclic mode to provide a higher throughput in a given time period. Furthermore, the proposed method is not restricted to the case where events in a later cycle may not happen before events in previous cycles.

Major contributions of this research are as follows:

1. Instead of simply implementing the "Just-in-Time" solution, or LNET, this thesis proposes an energy optimal implementation within the corridor of EPET and LNET specifications by exploiting the remaining degrees of freedom.

2. Competition and cooperation issues between users as well as cycles are handled by introducing of appropriate control arcs. While travelling arcs represent the activities of the users, control arcs of different orders ensure the safe sharing of

resources and nonblocking.

3. The proposed structure has the ability of optimal reaction to unexpected events. The supervisory level has the goal of finding online an optimal operating strategy. It updates the optimal EPET specifications for the lower level. The lower level then generates the LNET and provides the energy optimal implementation accordingly.

4. For small systems, we extend the structure to a more generalised setting, introduce and analyse the resulting negative order systems which could provide better performance. For this class of generalised systems, including non-strictly cyclic systems, we provide the corresponding max-plus models and control.

5. Besides train-track systems, we also discuss the application of the proposed structure in the field of High Throughput Screening. The differences and similarities between applications in these fields are presented by comparing a specific HTS plant to its corresponding train-track counterpart.

6. A novel intuitive algorithm for efficiently finding the shortest path of a class of simple polygons (also for finding the energy optimal trajectory) is developed for the lower level of the proposed control structure. On the upper level, to enhance the efficiency of optimisation, a procedure for reducing the search space is provided.

6.2 Future work

As for the future, there are several aspects to improve or to extend the work in this thesis. First, for the control of DESs containing negative order arcs, more efficient optimisation algorithms and more efficient methods for checking feasible plan lists are strongly desirable. The current approach works well for "small" negative order systems. With more shared resources, there are more feasible plans, and this combinatoric explosion is the most important problem which limits the work from being used in large scale systems. Although this problem cannot be completely avoided, there are possible ways to enhance optimisation efficiency. For example, in [69], computational efficiency is improved by

using mixed integer optimisation. This approach can also be used for the problems discussed in this work.

Second, although there are several different definitions on the stability of max-plus systems, this issue has been studied neither deeply nor widely so far. Some thoughts are as follow. For instance, for the TEG (timed event graph) corresponding to a max-plus system, stability means that tokens do not accumulate indefinitely inside the graph ([22]). Heidergott et al. give the definition of stability for a cyclic system with a timetable in [50]: "a scheduled system is called stable if any finite delay settles in finite time." Similar to the definition of Lyapunov stability in conventional continuous-time systems, Necoara et al. [74] proposed a definition of stability for strictly cyclic systems with a state feedback controller: "a closed-loop system $X(k) = AX(k-1) \oplus BU(X(k-1))$ is stable iff the state remains bounded, i. e. for every $\delta > 0$ there exists a real-valued function $\theta(\delta) > 0$ such that $\|X(0) - X_{el}\|_\infty \leq \delta$ implies $\|X(k) - X_{el}\|_\infty \leq \theta(\delta)$ for all $k \geq 0$." In general, for a fixed timetable, the scheduled event time $X_t(k) = X_t(1) + (k-1) \times T_{ct}$, where T_{ct} is the cycle time. Note that in this definition, time has been normalised such that the scheduled event times take the same value for different cycles: $X_t(k) = X_t(1)$. In addition, $\|X(k) - X_{el}\|_\infty = \max_{i \in n}\{(X(k) - X_{el})_i, (X_{el} - X(k))_i\}$, where X_{el} is the largest equilibrium state corresponding to the normalised setpoint X_t, $X_{el} \leq X_t$. Stability of max-plus system is an interesting topic for strictly cyclic systems and may also be extended to systems which have no regular operating pattern.

Finally, it is expected that the proposed control structure can be applied to other DES application areas such as the manufacturing industry.

Appendix A

Proofs

A.1 Shortest path and minimum energy trajectory

1. Solution to the shortest path problem

Proof We only discuss the case when M is located on the right of line OF. The case when M is located on the left of line OF is analogous.

As shown in Fig. A.1, the horizontal distance from M to OF is x with $x \in [b_1, \frac{b}{a} - b)$. We want to show that the length of the path OMF is minimal for $x = b_1$. If we denote the point $(b + b_1, a)$ by M^*, this is equivalent to showing that $length(OM^*F) < length(OMF)$ for all $M \neq M^*$. Denote the intersection of lines

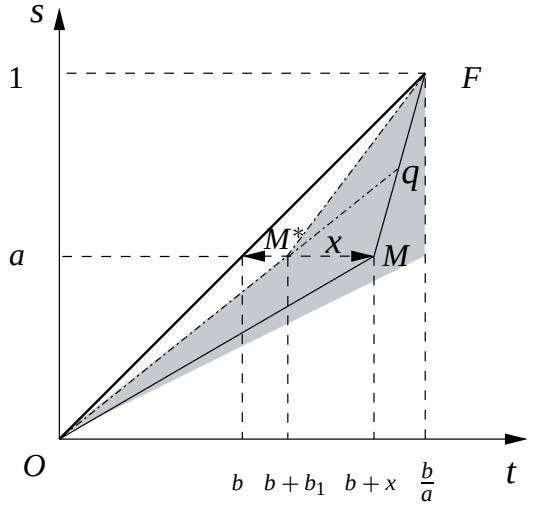

Figure A.1: Shortest path

OM^* and FM ($M \neq M^*$) by q. Then,

$$\begin{aligned} length(OMF) &= length(OMq) + length(qF) \\ &> length(Oq) + length(qF) \\ &= length(OqF) \\ &= length(OM^*) + length(M^*qF) \\ &> length(OM^*) + length(M^*F) \\ &= length(OM^*F) \end{aligned}$$

□

2. Solution to the minimum energy trajectory problem

Proof For the trajectory OMF, no matter whether M is located on the right of OF ($x \in [b_1, \frac{b}{a} - b)$ with $0 < b_1 < (\frac{b}{a} - b)$), or on the left of OF (i. e. $x \in (-b, -b_2]$ with $0 < b_2 < b$),

$$\begin{aligned} J_m &= \int v_m^2 dt \\ &= \int_0^{b+x} \Big(\frac{a}{b+x}\Big)^2 dt + \int_{b+x}^{\frac{b}{a}} \Big(\frac{1-a}{\frac{b}{a} - b - x}\Big)^2 dt \\ &= \frac{a^2}{b+x} + \frac{(1-a)^2}{\frac{b}{a} - b - x} \\ &= \frac{a^2}{b+x} + \frac{a(1-a)^2}{b - ab - ax}, \end{aligned} \tag{A.1}$$

$$\begin{aligned} J'_m := \frac{dJ_m}{dx} &= \frac{-a^2}{(b+x)^2} + \frac{a^2(1-a)^2}{(b-ab-ax)^2} \\ &= \frac{(\frac{a}{b})^2}{(1+\frac{x}{b})^2\Big(1 - \frac{ax}{b(1-a)}\Big)^2}\Big[(1+\frac{x}{b})^2 - \Big(1 - \frac{ax}{b(1-a)}\Big)^2\Big]. \end{aligned} \tag{A.2}$$

When $x \in [b_1, \frac{b}{a} - b)$, $(1+\frac{x}{b}) > 1, 0 < (1 - \frac{ax}{b(1-a)}) < 1$, thus $(1+\frac{x}{b})^2 - (1 - \frac{ax}{b(1-a)})^2 > 0$ and $J'_m > 0$, i. e. J_m increases strictly monotonically in x. The trajectory OMF with $x = b_1$ is therefore the minimum energy trajectory.

When $x \in (-b, -b_2]$, $0 < (1 + \frac{x}{b}) < 1$, $(1 - \frac{ax}{b(1-a)}) > 1$ since $a < 1$ (see Fig. 2.7 or Fig. A.1), thus $(1 + \frac{x}{b})^2 - (1 - \frac{ax}{b(1-a)})^2 < 0$ and $J'_m < 0$, i. e. J_m decreases strictly monotonically in x. Therefore the trajectory OMF with $x = -b_2$ is the minimum energy trajectory.

Finally, the above two results also imply that if M is located on the straight line OF, i. e. when M is located on the "right" of OF with $x = b_1 = 0$ or if M is located on the "left" of OF with $x = b_2 = 0$, both the corresponding J_m have the (same) minimum value. Therefore, the minimum energy trajectory between event O and F without any constraints is the straight line OF. In conclusion, the smallest absolute value $|x|$ gives the minimum energy trajectory. □

A.2 Shortest path of subpolygon P_i of P

Proof Based on Solution 1"kv_{i-1}-$rtkv_i$" (page 27), for a subpolygon P_i with the source vertex kv_{i-1} and the target vertex kv_{i+1}, we prove the key vertex path "kv_{i-1}-kv_i-kv_{i+1}" is the shortest path by considering the following different cases in terms of the properties of key vertices, e. g. the types (EPET or LNET), locations, etc. Here we suppose kv_i is an LNET. For an EPET type kv_i, the proof is similar.

Case 1 kv_{i+1} is the auxiliary temporal key vertex related to kv_i (i. e. $rtkv_i$).
According to step 3 of the key vertices' search procedure, "kv_{i-1}-kv_i-kv_{i+1}" obviously is the shortest path from kv_{i-1} to kv_{i+1} within P_i.

Case 2 kv_{i+1} and the auxiliary temporal key vertex $rtkv_i$ are same level vertices, but not identical. This case has two possibilities.

1. $rtkv_i$ is LNET and kv_{i+1} is EPET.
Since both kv_i and kv_{i+1} are key vertices, according to step 3 of the key vertices'

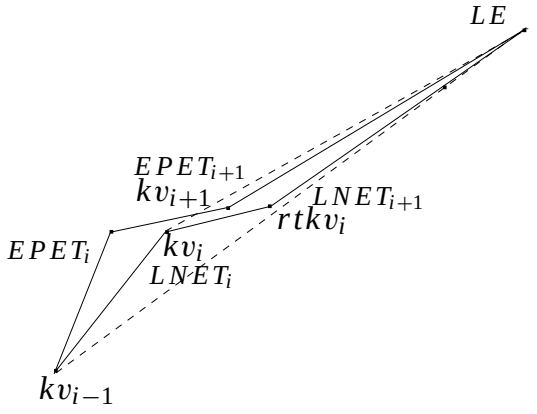

Figure A.2: $rtkv_i$: LNET, kv_{i+1}: EPET. kv_{i+1} is at the right of "kv_{i-1}-kv_i"

search procedure, "kv_{i-1}-kv_i-kv_{i+1}" is either inside P_i or part of the border of P_i. $rtkv_i$

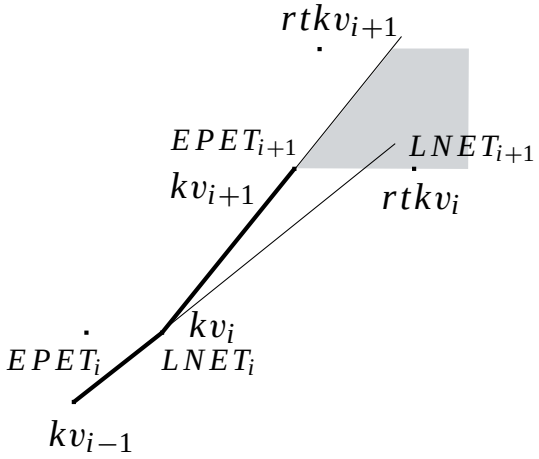

Figure A.3: $rtkv_i$: LNET, kv_{i+1}: EPET. kv_{i+1} is at the left of "kv_{i-1}-kv_i"

(LNET$_{i+1}$) is located at the right of "kv_{i-1}-kv_i" since kv_i is an LNET (otherwise the straight line "kv_{i-1}-$rtkv_i$" is inside P_i). If EPET$_{i+1}$, i. e. kv_{i+1}, is also at the right of "kv_{i-1}-kv_i"(see for example Fig. A.2), according to Solution 1, it is clear that from kv_{i-1} to kv_{i+1}, the shortest path of P_i is "kv_{i-1}-kv_i-kv_{i+1}".

Now we assume that EPET$_{i+1}$ is at the left of "kv_{i-1}-kv_i", see Fig. A.3. In the following, we show that such an EPET$_{i+1}$ does not exist, i. e. the assumption is not true.

As kv_{i+1} is an EPET, $rtkv_{i+1}$ has to be on the left of "kv_i-kv_{i+1}" and of course also on the left of both "kv_{i-1}-kv_i" and "kv_{i-1}-$rtkv_i$". Because EPET$_{i+1}$ is kv_{i+1}, at the right of "kv_i-kv_{i+1}", there are no EPET vertices located in the gray area below $rtkv_{i+1}$ but above EPET$_{i+1}$ and LNET$_{i+1}$. In other words, if there are vertices at this area, all of them are LNETs. On the other hand, since LNET$_{i+1}$ is $rtkv_i$, tkv_x, the kv_i-related tkv[1] which is the one located right above LNET$_{i+1}$ is at the right of "kv_{i-1}-$rtkv_i$". If this tkv_x is located below $rtkv_{i+1}$, as it is an LNET, there is tkv_y, another kv_i-related tkv which is located above tkv_x and to the right of "kv_{i-1}-tkv_x" and "kv_{i-1}-$rtkv_i$", ...

In brief, if there are kv_i-related tkvs located in the gray area, all of them are LNETs. If tkv_b is located above tkv_a, it also located to the right of the line "kv_{i-1}-tkv_a". Thus, no matter whether there are kv_i-related tkvs located in this area or not, the kv_i-related tkv (denoted as tkv_{aa}) located just above the area, if it exists, has to be at the right of "kv_{i-1}-$rtkv_i$".

With the above preparations, we can prove that the assumption cannot be true.

[1] kv_i-related tkv are all the temporal key vertices involved in the procedure of finding kv_i, together with lkv

First, since it is at the left of "kv_{i-1}-$rtkv_i$", $rtkv_{i+1}$ is not tkv_{aa} and thus it cannot be a kv_i-related tkv, tkv_{aa} (if it exists) is located above $rtkv_{i+1}$. Second, $rtkv_{i+1}$ is not an LNET. Otherwise it becomes a kv_i-related tkv because the straight line "kv_{i-1}-tkv_{aa}" violates the restriction of $rtkv_{i+1}$. Furthermore, $rtkv_{i+1}$ is not the target vertex of P, i. e. LE, the EPET of the last event. Otherwise there is an immediate contradiction, because from the kv_i-searching point of view, it is a kv_i-related tkv and has to be at the right of "kv_{i-1}-$rtkv_i$" as well. For such a non-LE EPET $rtkv_{i+1}$, there should exist a $2rtkv_{i+1}$, i. e. the kv_{i+1}-related tkv which is the one just above $rtkv_{i+1}$ and located at the left of "kv_i-$rtkv_{i+1}$", "kv_{i-1}-kv_i" and "kv_{i-1}-$rtkv_i$".

Similar to the above analysis, it can be concluded that $2rtkv_{i+1}$ is an EPET (but not LE) and there should also exist $jrtkv_{i+1}$ ($j = 3, \ldots, n$), i. e. all kv_{i+1}-related tkvs. They are non-LE EPETs and located at the left of "kv_{i-1}-$rtkv_i$". Apparently, as the highest $jrtkv_{i+1}$, $nrtkv_{i+1}$ has to be LE, this contradicts the feature of non-LE. Thus, there is no corresponding $jrtkv_{i+1}$ ($j = 2, \ldots, n$) and $rtkv_{i+1}$ which make EPET_{i+1} located at the left of "kv_{i-1}-kv_i" be kv_{i+1}, the assumption is not true.

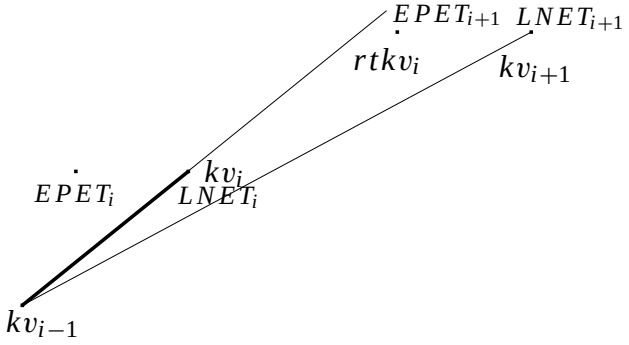

Figure A.4: $rtkv_i$: EPET, kv_{i+1}: LNET

2. $rtkv_i$ is EPET and kv_{i+1} is LNET.
Assume such a kv_{i+1} exists. Being $rtkv_i$, EPET_{i+1} is at the right of "kv_{i-1}-kv_i". Hence kv_{i+1} (i. e. LNET_{i+1}) must be also at the right of "kv_{i-1}-kv_i", see Fig. A.4. Obviously, the straight line "kv_{i-1}-kv_{i+1}" is not contained in P_i. Therefore, according to Solution 1, the path "kv_{i-1}-kv_i-kv_{i+1}" is the shortest path of P_i. Moreover, such a kv_{i+1} actually does not exist if we analyse in a similar way as shown in Fig. A.3.

Hence, for Case 2, in P_i, the shortest path from kv_{i-1} to kv_{i+1} is "kv_{i-1}-kv_i-kv_{i+1}".

Case 3 kv_{i+1} is located above kv_i but below $rtkv_i$.
The fact that kv_{i+1} is located above kv_i but below $rtkv_i$ implies that EPET_{i+1} is at the

left of "kv_{i-1}-$rtkv_i$" and LNET$_{i+1}$ is at the right of "kv_{i-1}-$rtkv_i$" as well as "kv_{i-1}-kv_i". If LNET$_{i+1}$ is the kv_{i+1}, according to Solution 1, obviously the shortest path from kv_{i-1} to kv_{i+1} is "kv_{i-1}-kv_i-kv_{i+1}". The result also holds if EPET$_{i+1}$ which is located at the right of "kv_{i-1}-kv_i" is the $(i+1)$st key vertex kv_{i+1}.

For EPET$_{i+1}$ which is kv_{i+1} and is located at the left of "kv_{i-1}-kv_i" (see e. g. Fig. A.5), its $rtkv_{i+1}$ has to be at the left of "kv_i-kv_{i+1}". In the following, we show that such an $rtkv_{i+1}$ and thus the corresponding kv_{i+1} do not exist.

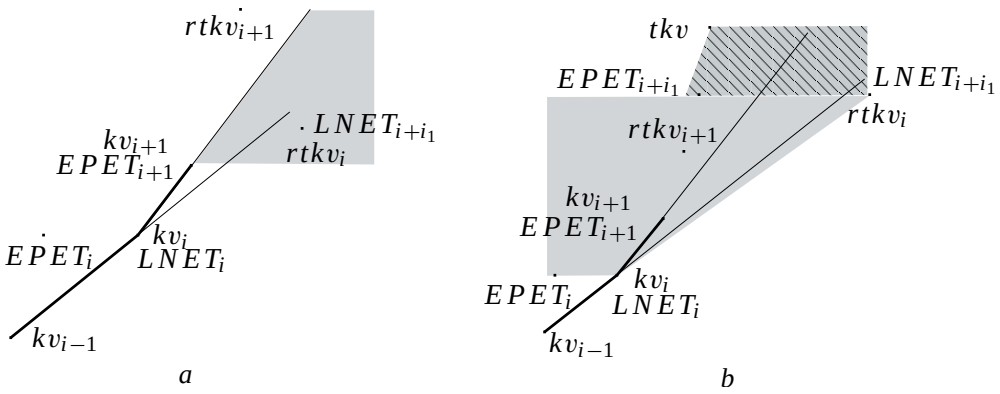

Figure A.5: $rtkv_i$: LNET, kv_{i+1}: EPET

1. Assume $rtkv_i$ is an LNET.
In Fig. A.5a, $rtkv_{i+1}$ is located above $rtkv_i$ and it is similar to Fig. A.3 except that the gray area of Fig. A.5a includes the additional area below $rtkv_i$ which does not influence the problem. It can be concluded that such $rtkv_{i+1}$ and kv_{i+1} do not exist.

In Fig. A.5b, $rtkv_{i+1}$ is located below $rtkv_i$ (here $rtkv_i$ is LNET$_{i+i_1}$). According to step 3 of the kv_i search procedure, at the left of "kv_i-$rtkv_i$", all vertices including $rtkv_{i+1}$ in the gray area below $rtkv_i$ but above LNET$_i$ and EPET$_i$ are EPETs. Similarly, suppose the kv_{i+1}-related tkv which is located above the area is tkv_{aa}, it could be either EPET$_{i+i_1}$ or some other vertex which is located above EPET$_{i+i_1}$. Since EPET$_{i+i_1}$ is an EPET, no matter which one tkv_{aa} is, right above EPET$_{i+i_1}$, there is always a kv_{i+1}-related tkv at the left of "kv_i-$rtkv_{i+1}$" and "kv_i-kv_{i+1}". Since at the right of the line "kv_i-tkv", the vertices in the bold line area below tkv but above EPET$_{i+i_1}$ and LNET$_{i+i_1}$ are all LNETs, the rest of the problem is similar to Fig. A.5a and Fig. A.3. Therefore, finally it can be concluded that the corresponding kv_{i+1} does not exist, if $rtkv_i$ is an LNET.

2. Assume $rtkv_i$ is an EPET.
If $rtkv_{i+1}$ is located above $rtkv_i$ as shown in Fig. A.5a (instead of LNET$_{i+i_1}$, now $rtkv_i$

is EPET$_{i+i_1}$), all vertices in the gray area are LNETs. This contradicts the $rtkv_i$. Thus the corresponding kv_{i+1} does not exist.

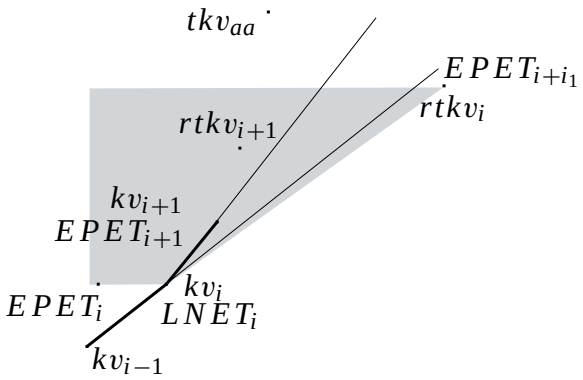

Figure A.6: $rtkv_i$: EPET, kv_{i+1}: EPET

If $rtkv_{i+1}$ is located below $rtkv_i$ as shown in Fig. A.6, which is similar to Fig. A.5*b*, all vertices in the gray area are EPETs and all the kv_{i+1}-related tkvs in the area are at the left of "kv_i-kv_{i+1}". Suppose the kv_{i+1}-related tkv which is located right above the area is tkv_{aa} which is higher than EPET$_{i+i_1}$ and at the left of "kv_i-$rtkv_{i+1}$" and "kv_i-kv_{i+1}". Located at the right of "kv_i-kv_{i+1}", EPET$_{i+i_1}$ is also a kv_{i+1}-related tkv which contradicts the kv_{i+1}-related tkvs in the gray area. Hence the corresponding kv_{i+1} does not exist.

Therefore, for Case 3, the shortest path from kv_{i-1} to kv_{i+1} is "kv_{i-1}-kv_i-kv_{i+1}".

Case 4 kv_{i+1} is located above $rtkv_i$.
Since "kv_i-kv_{i+1}" is not outside P_i and kv_{i+1} is located above $rtkv_i$, kv_{i+1} is at the right of "kv_i-EPET$_{i+i_1}$" but at the left of "kv_i-LNET$_{i+i_1}$", EPET$_{i+i_1}$ and LNET$_{i+i_1}$ are the EPET and the LNET of $rtkv_i$ event.

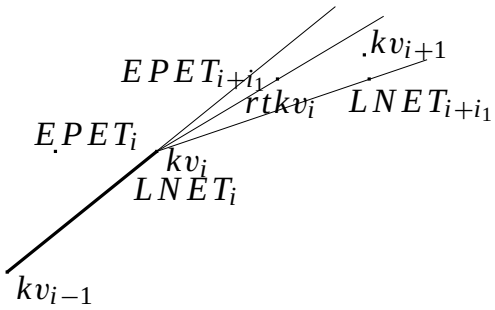

Figure A.7: $rtkv_i$: EPET

1. Assume $rtkv_i$ is an EPET (EPET_{i+i_1}).
Since $rtkv_i$ is located at the right of "kv_{i-1}-kv_i" (note that kv_i is an LNET), kv_{i+1} is also located at the right of "kv_{i-1}-kv_i", see Fig. A.7. As kv_i is an LNET, according to Solution 1, the shortest path from kv_{i-1} to kv_{i+1} is "kv_{i-1}-kv_i-kv_{i+1}".

2. Assume $rtkv_i$ is an LNET (LNET_{i+i_1}).
2.a Assume kv_{i+1} is at the left of "kv_i-$rtkv_i$" but not at the left of "kv_{i-1}-$rtkv_i$".

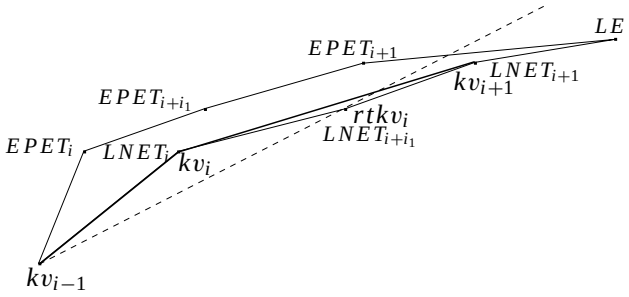

Figure A.8: $rtkv_i$: LNET

As an example shown in Fig. A.8, because of the vertex $rtkv_i$, i. e. LNET_{i+i_1}, from kv_{i-1} to kv_{i+1}, inside P_i, a direct path "kv_{i-1}-kv_{i+1}" is impossible. For the best situation, when the straight line "$rtkv_i$-kv_{i+1}" is not outside P_i, considering the constraints of vertices EPET_{i+i_1} and LNET_{i+i_1}, the shortest path is "kv_{i-1}-LNET_{i+i_1}-kv_{i+1}". However, the lower half part: "kv_{i-1}-LNET_{i+i_1}" is outside P_i because of the vertex kv_i (i. e. LNET_i). Considering the constraints of vertices EPET_i and LNET_i, "kv_{i-1}-LNET_i-kv_{i+1}" is the shortest path and it is not outside P_i, i. e. the shortest path from kv_{i-1} to kv_{i+1} is "kv_{i-1}-kv_i-kv_{i+1}".

2.b Assume kv_{i+1} is at the right of "kv_i-EPET_{i+i_1}" but at the left of "kv_{i-1}-$rtkv_i$".
In the following, we prove that this kind of kv_{i+1} does not exist.

Suppose kv_{i+1} is at the right of "kv_i-EPET_{i+i_1}" but at the left of "kv_{i-1}-$rtkv_i$". As "kv_i-kv_{i+1}" is not outside P_i, all vertices in the gray area shown in Fig. A.9 are LNETs. In the area, if above $rtkv_i$, there are kv_i-related tkvs, they are all at the right of "kv_{i-1}-$rtkv_i$". tkv_{aa}, the kv_i-related tkv which is located right above the gray area has to be at least at the right of "kv_{i-1}-$rtkv_i$". Thus kv_{i+1} is not a kv_i-related tkv and tkv_{aa} (if it exists) is located above kv_{i+1}. Moreover, such a kv_{i+1} has to be a non-LE EPET. Otherwise there is an immediate contradiction. Then the corresponding $rtkv_{i+1}$ is located at the left of "kv_i-kv_{i+1}" and the vertices of P in the bold line area (at the right of "kv_i-$rtkv_{i+1}$",

above kv_{i+1} but below $rtkv_{i+1}$) are all LNETs. Therefore tkv_{ab}, the kv_i-related tkv which is located right above the bold line area, is at the right of "kv_{i-1}-$rtkv_i$". Again, $rtkv_{i+1}$ is not a kv_i-related tkv and it has to be a non-LE EPET. Similarly, there should be $jrtkv_{i+1}(j = 2, \ldots, n)$ which are non-LE EPETs and are located at the left of "kv_i-kv_{i+1}" and therefore also at the left of "kv_{i-1}-$rtkv_i$". As the highest $jrtkv_{i+1}$, $nrtkv_{i+1}$ has to be LE which contradicts the feature of non-LE. Thus, there is no corresponding $jrtkv_{i+1}(j = 2, \ldots, n)$, $rtkv_{i+1}$ and kv_{i+1}.

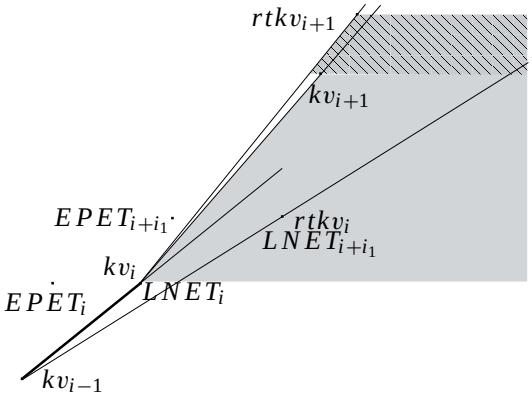

Figure A.9: $rtkv_i$: LNET, kv_{i+1} is located above $rtkv_i$

Thus, for Case 4, the shortest path from kv_{i-1} to kv_{i+1} is "kv_{i-1}-kv_i-kv_{i+1}".

Considering all 4 cases, we can conclude that for P_i, the shortest path from kv_{i-1} to kv_{i+1} is "kv_{i-1}-kv_i-kv_{i+1}". □

Appendix B

Notation

Abbreviations

C/D	Continuous/discrete
CPT	Current position
DES	Discrete event systems
EPET	Earliest possible event time
HTS	High throughput screening
LE	EPET of the last event
LNET	Latest necessary event time
TEG	Timed event graph
tb	train-batch

Latin Symbols

$A_0(k)$	0-order system matrix of cycle k
$A_{01}(k)$	system matrix of cycle k corresponding to zero order travelling arcs
$A_{02}(k)$	system matrix of cycle k corresponding to zero order control arcs
$A_1(k)$	1-order system matrix of cycle k
$A_{11}(k)$	system matrix of cycle k corresponding to first order travelling arcs
$A_{12}(k)$	system matrix of cycle k corresponding to first order control arcs
$A_j(k)$	j-order system matrix of cycle k
$A(\gamma)$	system matrix after γ-transformation on A_j with all j
$\widetilde{A_j}$	j-order system matrix after eliminating negative orders of control arcs
$\widetilde{A}(\gamma)$	transformed $A(\gamma)$ with transforming matrix $\mathcal{T}$

Continued on next page

Continued from previous page

$\widehat{A_0}$	0-order system matrix after eliminating high (i. e. $j \geq 2$) orders in $\widetilde{A_j}$
$\widehat{A_1}$	1-order system matrix after eliminating high (i. e. $j \geq 2$) orders in $\widetilde{A_j}$
$\hat{A}_{-j}(k)$	$-j$ order influence matrix, $1 \leq j \leq mn(k)$
$\widehat{A_{-j}}(k)$	$-j$ order synthesised influence matrix, $1 \leq j \leq mn(k)$
AC_j	index set of control arcs contained in the circuit pattern pac_j
B	input matrix
C	output matrix
C_j	a circuit or a potential circuit
$cca(i, j)$	number of cycle changes between control arc i and its successor in pac_j
$CI(i, sel)$	cycle index of control arc i in a combination sel
$d_j(t_k)$	distance a train has to go to reach event j at time t_k
e	0, neutral element of multiplication in max-plus algebra
f_i	number of cycles a event i being shifted forward
I	identity matrix in max-plus algebra
Ins	In-node set
$i_{pre}(i, j)$	index of predecessor control arc for control arc i in pac_j
$i_{suc}(i, j)$	index of successor control arc for control arc i in pac_j
$Ia(i, k)$	index of the cycle to which the control arc i in cycle k points
$K_\triangle$	index of the timetable recovered cycle
k_c	number of control arcs contained in a circuit pattern pac_j
k_d	index of the cycle where a delay occurs
k_l	number of cycles in a plan list
k_{lm}	maximum number of cycles over which the feasibility test needs to extend
k_{nc}	number of negative order arcs contained in cycle k
kv	key vertex on the shortest path
K_n	absolute value of the lowest order of a plan
$kins$	key in-node set
$Kins$	refined kins
KN	matrix storing Kins information
$mn(k)$	absolute value of the minimum order of the plan in cycle k
$mp(k)$	maximum positive order of the plan in cycle k
N	null matrix in max-plus algebra
N_a	maximum number of activities on the routes of train-batches
N_c	number of cycles existing concurrently during runtime

Continued on next page

Continued from previous page

N_{com}	number of combinations of control arcs on a shared resource
N_{mc}	maximum number of cycles existing concurrently
N_{mcr}	maximum number of cycles existing concurrently on a same route
N_o	absolute value of the lowest negative order of control arcs in a system
N_{re}	number of cycles needed to recover timetable without plan changing
N_t	number of trains
N_{uni}	number of cycles contained in an independent cycle unit
$Oa(i,k)$	index of the cycle from which the control arc i in cycle k points
$oc(i,k)$	order of control arc i chosen by the plan in cycle k
$od(C_j)$	order of the circuit C_j
$(Oin)_i$	order of the $(Ins)_i$ related control arc
$(Oinj)_i$	order of the $(Ins)_i$ related control arc of plan j
pac_j	circuit pattern j
P	simple polygon
P_i	subpolygon of P
Pf	matrix storing feasibility information of 2 sequential plans
Pl_i	plan i
$PS_m(j)$	index set for all events related to train m in cycle j
$QS(j)$	index set for destination-arrival events in cycle j
$\mathbb{R}$	set of real numbers
$\mathbb{R}_{max}$	$\{\mathbb{R}, -\infty, \oplus, \otimes\}$, max-plus algebra structure
$\mathbb{R}_{min}$	$\{\mathbb{R}, +\infty, \oplus', \otimes'\}$, min-plus algebra structure
$\mathbb{R}_{mm}$	$\{\mathbb{R}, -\infty, +\infty, \oplus, \otimes, \oplus', \otimes'\}$
$rtkv_i$	auxiliary temporary key vertex of kv_i
$sel(l, j)$	combination l of control arcs corresponding to pac_j
sp	set of all key vertices of the shortest path
tkv	temporal key vertex
t_{rel}	estimated release time for the blocked train
T_{ct}	cycle time
T_{sm}	safe margin between eigenvalue and cycle time
$\mathcal{T}$	transforming matrix, eliminates negative orders in $A(\gamma)$ to get $\widetilde{A}(\gamma)$
u	system input
v	vertex of P
v_e	eigenvector

Continued on next page

Continued from previous page

v_m	velocity of train m
v_{max}	maximum velocity
x_i	event time for event i
$\underline{x}_i$	EPET (earliest possible event time) of event i
$\underline{X}$	EPET (earliest possible event time) vector
$\underline{X}(\gamma)$	EPET vector after γ-transformation on $\underline{X}$
$\underline{XR}_m$	reformed $\underline{X}$ for train m
$\underline{X}\_in$	reinitialised state vector
$\overline{X}_m$	LNET (latest possible event time) vector for train m
$\widetilde{\underline{X}}(\gamma)$	transformed $\underline{X}(\gamma)$ with transforming matrix $\mathcal{T}$
$\widehat{\underline{X}}$	EPET vector corresponding to system matrices $\widehat{A}_0$ and $\widehat{A}_1$
$\widetilde{\underline{X}\_in}$	reinitialised state vector after eliminating negative orders of control arcs
Y	system output
$\mathbb{Z}$	set of integers
$\mathbb{Z}^+$	set of positive integers

Greek Symbols

ϵ	$-\infty$, neutral element of addition in max-plus algebra
λ	eigenvalue
$\pi(\text{CPT}, v)$	shortest path from CPT to v inside P
τ	a path inside P

Operations

$\oplus$	addition in max-plus algebra, $a \oplus b = \max(a, b)$
$\otimes$	multiplication in max-plus algebra, $a \otimes b = a + b$
$\oplus'$	addition in min-plus algebra, $a \oplus' b = \min(a, b)$
$\otimes'$	multiplication in min-plus algebra, $a \otimes' b = a + b$
$*$	for physically meaningful max-plus system matrix A, $A^* = I \oplus A \oplus \cdots \oplus A^{n-1}$

Subscripts

0	0-order
1	1-order

Bibliography

[1] M. Akian, R. Bapat, and S. Gaubert. Asymptotics of the perron eigenvalue and eigenvector using max-algebra. *C.R.A.S.*, 327:927 – 932, 1998.

[2] F. Baccelli. Ergodic theory of stochastic petri networks. *The Annals of Probability*, 20(1):375 – 396, 1992.

[3] F. Baccelli, G. Cohen, G.J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1992. Available online: http://www-rocq.inria.fr/metalau/cohen/SED/book-online.html.

[4] F. Baccelli, B. Gaujal, and D. Simon. Analysis of preemptive periodic real-time systems using the (max, plus) algebra with applications in robotics. *IEEE Transactions on Control Systems Technology*, 10:368 – 380, 2002.

[5] F. Baccelli and D. Hong. Tcp is max-plus linear. In *Proc. of ACM-SIGCOMM'00*, volume 30, pages 219 –230, Stockholm, September 2000.

[6] F. Baccelli and J. Mairesse. Ergodic theory of stochastic operators and discrete event networks. In J. Gunawardena, editor, *Idempotency*, volume 11, pages 171 – 208. Publications of the Isaac Newton Institute, Cambridge University Press, Cambridge, 1998.

[7] R. B. Bapat. A max version of the perron-frobenius theorem. *Linear Algebra and Its Applications*, 275-276:3 – 18, 1998.

[8] R. B. Bapat, D. P. Stanford, and P. van den Driessche. Pattern properties and spectral inequalities in max algebra. *SIAM Journal on Matrix Analysis and Applications*, 16(3):964 – 976, 1995.

[9] M. DE Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[10] J.-L. Boimond and J.-L. Ferrier. Internal model control and max-algebra: Controller design. *IEEE Transactions on Automatic Control*, 41(3):457 – 461, 1996.

[11] J. G. Braker and G. J. Olsder. The power algorithm in max algebra. *Linear Algebra and its Applications*, 182:67 – 89, 1993.

[12] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999.

[13] C. Chang, R. Nelson, and D.D. Yao. Scheduling parallel processors: Structural properties and optimal policies. *Mathematical and Computer Modelling*, 23 (11/12):93 –114, 1996.

[14] W. Chen, J. Xu, and Q. Liang. Period of processing on serial production line, optimal scheduling and application. *Discrete Event Dynamic Systems*, 9:9 – 21, 1999.

[15] Y. Cheng and D. Zheng. A cycle time computing algorithm and its application in the structural analysis of min-max systems. *Discrete Event Dynamic Systems*, 14 (1):5 – 30, 2004.

[16] Y. Cheng and D. Zheng. Min-max inequalities and the timing verification problem with max and linear constraints. *Discrete Event Dynamic Systems*, 15(2):119 – 143, 2005.

[17] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. Gettrick, and J. P. Quadrat. Numerical computation of spectral elements in max-plus algebra. In *Proceedings 5th IFAC Conference on System Structure and Control 2*, pages 667 – 674. Elsevier, 1998.

[18] J. Cochet-Terrasson, S. Gaubert, and J. Gunawardena. Dynamics of min-max functions. Technical Report HPL-BRIMS-97-13, HP Laboratories Bristol, August 1997. Available online: http://www.hpl.hp.com/techreports/97/HPL-BRIMS-97-13.pdf.

[19] J. Cochet-Terrasson, S. Gaubert, and J. Gunawardena. A constructive fixed point theorem for min-max functions. *Dynamics and Stability of Systems*, 14(4):407 – 433, 1999.

[20] G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot. A linear system-theoretic view of discrete event processes and its use for performance evaluation in manufacturing. *IEEE Transactions on Automatic Control*, AC-30(3):210 –220, 1985.

[21] G. Cohen, S. Gaubert, and J.-P. Quadrat. From first to second-order theory of linear discrete event systems. In *Proc. 12th IFAC World Congress*, Sydney, Australia, July 1993.

[22] G. Cohen, S. Gaubert, and J.-P. Quadrat. Max-plus algebra and system theory: Where we are and where to go now. *Annual Reviews in Control*, 23:207 – 219, 1999.

[23] G. Cohen, P. Moller, J.P. Quadrat, and M. Viot. Linear system theory for discrete-event systems. In *Proceedings of the 23rd IEEE Conference on Decision and Control*, volume 1, pages 539 – 544, Las Vegas, NV., December 1984.

[24] B. Cottenceau, L. Hardouin, J.-L. Boimond, and J.-L. Ferrier. Synthesis of greatest linear feedback for timed event graphs in dioid. *IEEE Transactions on Automatic Control*, 44(6):1258 – 1262, 1999.

[25] B. Cottenceau, L. Hardouin, J.-L. Boimond, and J.-L. Ferrier. Model reference control for timed event graphs in dioids. *Automatica*, 37:1451 – 1458, 2001.

[26] R. Cuninghame-Green. *Minimax-Algebra*, volume 166 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 1979.

[27] R. A. Cuninghame-Green. Minimax algebra and applications. *Fuzzy Sets and Systems*, 41(3):251– 267, 1991.

[28] R. A. Cuninghame-Green and Y. Lin. Maximum cycle-means of weighted digraphs. *Appl. Math. JCU*, 11:225 – 234, 1996.

[29] A. Dasdan and R. K. Gupta. Faster maximum and minimum mean cycle algorithms for system performance analysis. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 17(10):889 – 899, 1998.

[30] R. de Vries, B. De Schutter, and B. de Moor. On max-algebraic models for transportation networks. In *Proc. 4th Workshop on Discrete Event Systems (WODES'98)*, pages 457 – 462, Cagliari, Italy, August 1998.

[31] E. V. Denardo and B. L. Fox. Multichain markov renewal programs. *SIAM Journal on Applied Mathematics*, 16(3):468 – 487, 1968.

[32] I. Elmahi, O. Grunder, and A. Elmoudni. A max plus algebra approach for modeling and control of lots delivery: Application to a supply chain case study. In *2004 IEEE International Conference on Industrial Technology*, pages 926 – 931, Hammamet, Tunisia, December 2004.

[33] L. Elsner and P. van den Driesche. On the power algorithm in max algebra. *Linear Algebra and its Applications*, 302-303:17 – 32, 1999.

[34] L. Elsner and P. van den Driesche. Modifying the power method in max algebra. *Linear Algebra and Its Applications*, 332-334:3 – 13, 2001.

[35] A. Di Febbraro, S. Grosso, and R. Minciardi. Max-plus algebra and incremental scheduling problems in manufacturing systems. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 1583–1587, Lake Buena Vista, FL, USA, December 1994.

[36] A. Di Febbraro, R. Minciardi, G. Parodi, A. Prati, M. Profumo, and S. Grosso. Performance optimization in manufacturing systems by use of max-plus algebraic techniques. In *Proc. IEEE Int. Conf. on Systems Man and Cybernetics*, pages 1995–2001, San Antonio, TX, October 1994.

[37] R. Franke, M. Meyer, and P. Terwiesch. Optimal control of the driving of trains. *At - Automatisierungstechnik*, 50(12):606–613, 2002.

[38] S. Gaubert and J. Gunawardena. The duality theortem for min-max functions. *C. R. Acad. Sci.*, 326:43 – 48, 1998.

[39] S. Gaubert and Max Plus. Methods and applications of (max,+) linear algebra. In R. Reischuk and M. Morvan, editors, *14th Symp. on Theoretical Aspects of Computer Science, Lübeck, Germany*, volume 500 of Lect. Notes in Comp. Sc., pages 261 – 282, Berlin, 27 Feb.- 1 Mar. 1997. Springer-Verlag.

[40] Martin Gavalec. Monotone eigenspace structure in max-min algebra. *Linear Algebra and its Applications*, 345:149 – 167, 2002.

[41] M. J. Gazarik and E. W. Kamen. Reachability and observability of linear systems over max-plus. *Kybernetika*, 35(1):2 – 12, 1999.

[42] F. Geyer. Analyse und Optimierung zyklischer ereignisdiskreter Systeme mit Reihenfolgealternativen. Diplomarbeit, IFAT, Otto-von-Guericke-Universität Magdeburg, 2004.

[43] M. Gondran and M. Minoux. Linear algebra in dioids: A survey of recent results. In R. E. Burkard, R. A. Cuninghame-Green, and U. Zimmermann, editors, *Algebraic and Combinatorial Methods in Operations Research*, volume 19 of *Annals of Discrete Mathematics*, pages 147 – 164. Elsevier Science Publishers B. V. (North-Holland), 1984.

[44] H. Goto, K. Takeyasu, S. Masuda, and T. Amemiya. A gain scheduled model predictive control for linear-parameter-varying max-plus-linear systems. In *Proceedings of the American Control Conference*, pages 4016 – 4021, Denver,Colorado, USA, June 2003.

[45] R. M. P. Goverde and M. A. Odijk. Performance evaluation of network timetables using PETER. In J. Allan, E. Andersson, C. A. Brebbia, R. J. Hill, G. Sciutto, and S. Sone, editors, *Computers in Railways VIII*, pages 731 – 740. WIT Press, Southampton, 2002.

[46] J. Gunawardena. Cycle times and fixed points of min-max functions. In G. Cohen and J.-P. Quadrat, editors, *Proceedings of the 11th International Conference on Analysis and Optimization of Systems*, volume 199 of Lecture Notes in Control and Information Sciences, pages 266 – 272, Sophia-Antipolis, France, June 1994. Springer-Verlag: London.

[47] J. Gunawardena. Min-max functions. *Discrete Event Dynamic Systems*, 4(4):377 – 406, 1994.

[48] M. Hartmann and J. B. Orlin. Finding minimum cost to time ratio cycles with small integral transit times. *Networks*, 23(6):567 – 574, 1993.

[49] B. Heidergott and R. de Vries. Towards a (max, +) control theory for public transportation networks. *Discrete Event Dynamic Systems*, 11:371 – 398, 2001.

[50] B. Heidergott, G. J. Olsder, and J. van der Woude. *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press, 2006.

[51] B. Heidergott, J. van der Woude, and G. J. Olsder. An ergodic theorem for stochastic max-plus linear systems. In *Proceedings the IFAC Workshop on Discrete Event Systems (WODES'04)*, pages 97 – 102, Reims, France, September 2004.

[52] P. Howlett. The optimal control of a train. *Annals of Operations Research*, 98: 65–87, 2000.

[53] P.G. Howlett, I.P. Milroy, and P.J. Pudney. Energy-efficient train control. *Control Eng. Practice*, 2(2):193–200, 1994.

[54] K. Ito and K. K. Parhi. Determining the minimum iteration period of an algorithm. *J. VLSI Signal Processing*, 11(3):229 – 244, 1995.

[55] A. Jean-Marie and G. J. Olsder. Analysis of stochastic min-max-plus systems: Results and conjectures. *Mathematical and Computer Modelling*, 23(11/12):175 – 189, 1996.

[56] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309 – 311, 1978.

[57] R. M. Karp and J. B. Orlin. Parametric shortest path algorithms with an application to cyclic staffing. *Discrete Applied Mathematics*, 3(1):37 – 45, 1981.

[58] J. R. Kok and N. Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *Proceedings of the 9th International RoboCup Symposium*, Osaka, Japan, Jul. 2005. Available online: http://staff.science.uva.nl/∼jellekok/publications/2005/Kok05robocup.pdf.

[59] S. Lahaye, J.-L. Boimond, and L. Hardouin. Optimal control of (min,+) linear time-varying systems. In *8th International Workshop on Petri Nets and Performance Models*, pages 170 – 178, Saragosse, Spain, September 1999.

[60] S. Lahaye, J.-L. Boimond, and L. Hardouin. Analysis of periodic discrete event systems in (max,+) algebra. In R. Boel and G. Stremersch, editors, *Discrete Event Systems: analysis and control*, pages 93 – 102. Kluwer Academic Publishers, 2000.

[61] T.-E. Lee. Stable earliest starting schedules for cyclic job shops: a linear system approach. *International Journal of Flexible Manufacturing Systems*, 12(1):59 – 80, 2000.

[62] D. Li, E. Mayer, and J. Raisch. A novel hierarchical control architecture for a class of discrete-event systems. In *7th IFAC Workshop on DISCRETE EVENT SYSTEMS (WODES'04)*, pages 415 – 420, Reims, France, September 2004.

[63] D. Li, E. Mayer, and J. Raisch. A new hierarchical control scheme for a class of cyclically repeated discrete-event systems. In *Proceedings of the 2nd International Conference on Informatics in Control, Automation and Robotics*, volume IV, pages 30 – 36, Barcelona, Spain, September 2005.

[64] P. A. Lotito, E. M. Mancinelli, and J.-P. Quadrat. A min-plus derivation of the fundamental car-traffic law. *IEEE Transactions on Automatic Control*, 50(5):699 – 705, 2005.

[65] C. A. Maia, L. Hardouin, R. Santos-Mendes, and B. Cottenceau. Optimal closed-loop control of timed event graphs in dioids. *IEEE Transactions on Automatic Control*, 48(12):2284 – 2287, 2003.

[66] J. Mairesse. Products of irreducible random matrices in the (max,+) algebra. *Advances of Applied Probability*, 29(2):444 – 477, 1997.

[67] S. Masuda, H. Goto, T. Amemiya, and K. Takeyasu. An inverse system for linear parameter-varying max-plus-linear systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 4549 – 4554, Las Vegas, Nevada, USA, December 2002.

[68] E. Mayer. Ereignisdiskrete Modellierung und Analyse eines Schienennetzes. Studienarbeit, ISR, Universität Stuttgart, 1997.

[69] E. Mayer and J. Raisch. Time-optimal scheduling for high throughput screening processes using cyclic discrete event models. *Mathematics and Computers in Simulation*, 66:181 – 191, 2004.

[70] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414 – 424, 1979.

[71] E. Menguy, J.-L. Boimond, L. Hardouin, and J.-L. Ferrier. A first step towards adaptive control for linear systems in max algebra. *Discrete Event Dynamic Systems*, 10:347 – 367, 2000.

[72] E. Menguy, J.-L. Boimond, L. Hardouin, and J.-L. Ferrier. Just-in-time control of timed event graphs: Update of reference input, presence of uncontrollable input. *IEEE Transactions on Automatic Control*, 45(9):2155 – 2159, 2000.

[73] A. Nait Sidi Moh, M.-A. Manier, and A. El Moudni. A control policy for a public transport network modelled by petri nets and max-plus alebra. In *Proceedins of the 5th Biannual World Automation Congress*, pages 59 – 64, 2002.

[74] I. Necoara, T.J.J. van den Boom, B. De Schutter, and J. Hellendoorn. Stabilization of max-plus-linear systems using receding horizon control — The unconstrained case. In *Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'06)*, pages 148 –153, Alghero, Italy, June 2006.

[75] G. J. Olsder. Applications of the theory of stochastic discrete event systems to array processors and scheduling in public transportation. In *Proceedings of the 28th Conference on Decision and Control (CDC89)*, pages 2012 – 2017, Tampa, Florida, December 1989.

[76] G. J. Olsder. Eigenvalues of dynamic max-min systems. *Discrete Event Dynamic Systems*, 1(2):177 – 207, 1991.

[77] G. J. Olsder, J. A. C. Resing, R. E. DE Vries, M. S. Keane, and G. Hooghiemstra. Discrete event systems with stochastic processing times. *IEEE Transactions on Automatic Control*, 35(3):299 – 302, 1990.

[78] G. J. Olsder, K. Roos, and R.-J. van Egmond. An efficient algorithm for the critical circuits and finite eigenvectors in the max-plus algebra. *Linear Algebra and its Applications*, 295:231 – 240, 1999.

[79] Max Plus. Second order theory of min-linear systems and its application to discrete event systems. In *Proc. 30th Conf. Dec. and Cont.*, Brighton, England, December 1991. Available online: http://www-rocq.inria.fr/metalau/quadrat/cdc91.pdf.

[80] J.-M. Prou and E. Wagneur. Controllability in the max-algebra. *Kybernetika*, 35 (1):13 – 24, 1999.

[81] J.-P. Quadrat and M. Plus. Max-plus algebra and applications to system theory and optimal control. In *Proceedings of the International Congress of Mathematicians, Zurich, 1994*, pages 1502 – 1511. Basel: Birkaüser Verlag, 1995.

[82] J. A. C. Resing, R. E. DE Vries, G. Hooghiemstra, M. S. Keane, and G. J. Olsder. Asymptotic behavior of random discrete event systems. *Stoch. Proc. and Applications*, 36:195 – 216, 1990.

[83] I. V. Romanovskiĭ. Optimization of stationary control of a discrete deterministic process. *Kibernetika*, 3(2):66 – 78, 1967. English translation in Cybernetics and Systems Analysis 3(2), pp.52-62, 1967.

[84] G. Schullerus and V. Krebs. Input signal design for discrete event model based batch process diagnosis. In *4th IFAC Workshop on On-Line Fault Detection and Supervision in the Chemical Process Industries*, pages 69 – 74, Jejudo Island, Korea, June 2001.

[85] B. De Schutter and T. van den Boom. Model predictive control for max-min-plus systems. In R. Boel and G. Stremersch, editors, *Discrete Event Systems: Analysis and Control*, pages 201 – 208. Kluwer Academic Publ., 2000.

[86] B. De Schutter and T. van den Boom. Model predictive control for max-min-plus-scaling systems. In *Proceedings of the 2001 American Control Conference*, pages 319 – 324, Arlington, Virginia, June 2001.

[87] B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049 – 1056, 2001.

[88] B. De Schutter and T. van den Boom. Model predictive control for max-min-plus-scaling systems — efficient implementation. In *Proceedings of the 6th International Workshop on Discrete Event Systems (WODES'02)*, pages 343 – 348, Zaragoza, Spain, October 2002.

[89] B. De Schutter, T. van den Boom, and A. Hegyi. A model predictive control approach for recovery from delays in railway systems. *Transportation Research Record*, (1793):15 – 20, 2002.

[90] J.-W. Seo and T.-E. Lee. Steady-state analysis and scheduling of cyclic job shops with overtaking. *International Journal of Flexible Manufacturing Systems*, 14(4): 291–318, 2002.

[91] P. Spacek, A. El Moudni, S. Zerhouni, and M. Ferney. Max-plus algebra for discrete event systems - some links to structural controllability and structural observ-

ability. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 579 – 584, Monterey, CA, USA, August 1995.

[92] Subiono and J. van der Woude. Power algorithms for (max, +)- and bipartite (min, max, +)-systems. *Discrete Event Dynamic Systems*, 10:369 – 389, 2000.

[93] T. van den Boom and B. De Schutter. Model predictive control for perturbed max-plus-linear systems. *Systems and Control Letters*, 45:21 – 33, 2002.

[94] T. van den Boom and B. De Schutter. Modelling and control of discrete event systems using switching max-plus-linear systems. In *7^{th} IFAC Workshop on DISCRETE EVENT SYSTEMS (WODES'04)*, pages 115 – 120, Reims, France, September 2004.

[95] T. van den Boom, B. De Schutter, G. Schullerus, and V. Krebs. Adaptive model predictive control using max-plus-linear input-output models. In *Proceedings of the 2003 American Control Conference*, pages 933 – 938, Denver, Colorado, June 2003.

[96] J. van der Woude. A characterisation of the eigenvalue of a general (min, max, +)-system. *Discrete Event Dynamic Systems*, 11(3):203 – 210, 2001.

[97] J. van der Woude. A simplex-like method to compute the eigenvalue of an irreducible (max, +)-system. *Linear Algebra and its Applications*, 330:67 – 87, 2001.

[98] J. van der Woude and G. J. Olsder. On a proof of the general version of the spectral theorem in max-plus algebra. In *Proceedings of the 44th Conference on Decision and Control, and the European Control Conference 2005*, pages 7804 – 7809, Seville, Spain, December 2005.

[99] J. van der Woude and Subiono. Conditions for the structural existence of an eigenvalue of a bipartite (min, max, +)-system. *Theoretical Computer Science*, 293:13 – 24, 2003.

[100] N. Vlassis, R. Elhorst, and J. R. Kok. Anytime algorithms for multiagent decision making using coordinationgraphs. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 953 – 957, The Hague, The Netherlands, October 2004.

[101] C. Wende, Q. Xiangdong, and D. Shuhui. The eigen-problem and period analysis of the discrete-event system. *Systems Science and Mathematical Sciences*, 3(3): 200 – 230, 1990.

[102] G. Soto y Koelemeijer. The power and howard algorithm in the (max,+) semiring. In R. Boel and G. Stremersch, editors, *Discrete Event Systems: analysis and control*, pages 193 – 200. Kluwer Academic Publishers, 2000.

[103] N. E. Young, R. E. Tarjan, and J. B. Orlin. Faster parametric shortest path and minimum-balance algorithms. *Networks*, 21:205 – 221, 1991.

[104] Q. Zhu, W. Sheng, and N. Xi. Max-plus algebra model for on-line task scheduling of a reconfigurable manufacturing work-cell. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1245 – 1250, Sendai, Japan, 2004.

[105] U. Zimmermann. *Linear and Combinatorial Optimization in Ordered Algebraic Structures*, volume 10 of *Annals of Discrete Mathematics*. North-Holland, 1981.