2014

Modeling Bus Bunching with Petri Nets and Max-Plus Algebra

Harry Newcomb
Portland State University

Modeling Bus Bunching with Petri Nets and Max-Plus Algebra

by

Harry Newcomb

An undergraduate honors thesis submitted in partial fulllment
of the requirements for the degree of

Bachelor of Science

in

University Honors

and

Mathematics

Thesis Advisor
Prof. Gerardo Lafferriere

Portland State University
2014

Abstract

In this work, the possibilities of modeling bus bunching using Petri nets and max-plus algebra are investigated. The basic properties of max-plus algebra and Petri nets are introduced, and previous work modeling transportation networks with these tools is summarized. One previous model that incorporates a non-analytic feature is simplified to remove this feature while retaining the model's function, and it is proved that passenger interaction with the bus network cannot be modeled with autonomous timed event graphs with stop subnets.

Contents

1 Introduction

> Analytical studies of the bunching problem are few because the problem is difficult.
>
> Carlos F. Daganzo, "A headway-based approach to eliminate bus bunching: systematic analysis and comparison"

> I'm getting nowhere with my prototype. This has not in the least hindered the outpouring of my imagination. I regret only that my faith in the developmental power of misery has been cast so decisively in doubt.
>
> Paul Scheerbart, *The Perpetual Motion Machine*

Bus bunching is a phenomenon that should be familiar to anyone who has ever spent time on public transportation. As one bus fills up with passengers, it has to stop more often, allowing the bus behind it, because this bus has fewer passengers on board and fewer to pick up at each stop, to catch up. The result is a cascade of delay up and down the route. In the real world, there are many causes of bus bunching—traffic, breakdowns, all kinds of *force majeure*—but this paper focuses on delay created by passengers: the bunching caused by something internal and inherent in the running of the bus line.

The goal of this work is to investigate some possibilities of modeling passenger-caused delay in buses using the tools of timed Petri nets and max-plus algebra. Timed Petri nets are a modeling framework, particularly useful in situations where the *order* and *timing* of events is important. Max-plus algebra is a structure formed by taking the real numbers with the operations max (that is, the maximum function) and addition, and is useful for analyzing a certain kind of timed Petri net.

An example

Suppose we have a simple bus line with two stops, S_1 and S_2, as in Figure 1.1. It takes 6 minutes for a bus to drive from S_1 to S_2, and 4 minutes to drive from S_2 to S_1. At S_1, a bus will dwell for 1 minute, and for 0.5 minutes at S_2 to wait for people to arrive or disembark. But, while a bus isn't at a stop, passengers accrue there, and it takes them δ_1 minutes at stop 1 and δ_2 minutes at stop 2 to board the

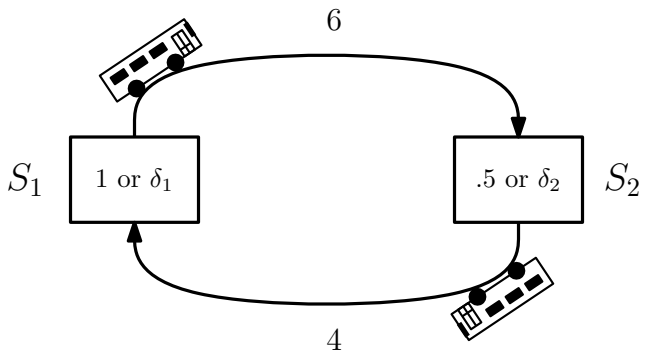


Figure 1.1

bus. If there are enough passengers at a stop, it will take longer for them to board than the scheduled dwell time, and the bus will be forced to wait at a stop for δ_1 or δ_2 minutes.

Let x_1 and x_2 indicate the time of departures from S_1 and S_2 respectively, where $x_1(k)$ is the kth departure of a bus from S_1, and $x_2(k)$ the kth from S_2. We can combine these into a vector, $\mathbf{x}(k)$,

$$\mathbf{x}(k) = \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix}$$

After a bus leaves S_2, it travels for 4 minutes to S_1, and then waits there either 1 minute or δ_1 minutes, whichever is greater, before it can depart S_1. At minimum, then, it takes $4 + \max\{1, \delta_1\}$ minutes after $x_2(k)$ for a bus to leave S_1. That is,

$$x_1(k+1) \geq x_2(k) + 4 + \max\{1, \delta_1\}.$$

So the timing of buses leaving S_1 depends on the timing of buses leaving S_2 beforehand. For our purposes, we can assume the buses are running as quickly as possible, which turns the above inequality into an equation:

$$x_1(k+1) = x_2(k) + 4 + \max\{1, \delta_1\}.$$

Similarly, for buses departing from S_2, we have

$$x_2(k+1) = x_1(k) + 6 + \max\{0.5, \delta_2\}.$$

We can express these equations together in vector form:

$$\mathbf{x}(k+1) = \begin{pmatrix} x_1(k+1) \\ x_2(k+1) \end{pmatrix} = \begin{pmatrix} x_2(k) + 4 + \max\{1, \delta_1\} \\ x_1(k) + 6 + \max\{0.5, \delta_2\} \end{pmatrix}.$$

So, using only addition and the maximum function, and given an initial condition $\mathbf{x}(0)$ that determines when a bus leaves each stop for the first time, we can determine

when a bus leaves a stop for the kth time. If there are buses leaving both stops for the first time at $t = 0$ minutes as in Figure 1.1, in other words if

$$\mathbf{x}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

and if the passenger boarding time never exceeds the dwell time, then the future firing times are

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 5 \\ 6.5 \end{pmatrix}, \begin{pmatrix} 11.5 \\ 11.5 \end{pmatrix}, \begin{pmatrix} 16.5 \\ 18 \end{pmatrix}, \ldots$$

This is approximately the kind of analysis of a bus system we can do with Petri nets and max-plus algebra. The trick will be those δ's—how can we construct a Petri net model that generates delays in the right way while still being analytically tractable? The ideal result would have been a Petri net model that is sophisticated enough to capture the passenger aspect, but simple enough that we could still analyze it in max-plus, but this does not appear to be possible within the narrow limits that were set.

In Chapter 2, the basic formalism of max-plus algebra and Petri nets are outlined, along with the method for translating a certain kind of Petri net into the language of max-plus. In Chapter 3, past attempts to model transportation with max-plus algebra and Petri nets are examined, In Chapter 4, an altered version of a model from the previous chapter is presented, and it is proved that one type of model is not adequate.

2 Max-Plus Algebra and Petri Nets

2.1 Max-plus basics

Max-plus algebra is a mathematical structure, the result of replacing addition and multiplication over the real numbers with, respectively, the maximum function and addition. More accurately, the operations are over the set $\mathbb{R}_{\max} := \mathbb{R} \cup \{-\infty\}$. The max function is symbolized $\oplus$ and addition is symbolized $\otimes$, while negative infinity is denoted ϵ and 0 is denoted e. That is, for all $a, b \in \mathbb{R}_{\max}$,

1. $a \oplus b = \max\{a, b\}$
2. $a \otimes b = a + b$
3. $a \oplus \epsilon = \epsilon \oplus a = a$
4. $a \otimes e = e \otimes a = a$.

This structure is a semiring, and thus has the following algebraic properties:

1. *Commutativity of $\oplus$ and $\otimes$.* For all $a, b \in \mathbb{R}_{\max}$,
$$a \oplus b = b \oplus a$$
and
$$a \otimes b = b \otimes a.$$
2. *Associativity of $\oplus$ and $\otimes$.* For all $a, b, c \in \mathbb{R}_{\max}$,
$$a \oplus (b \oplus c) = (a \oplus b) \oplus c$$
and
$$a \otimes (b \otimes c) = (a \otimes b) \otimes c.$$
3. *Existence of zero element.* For all $a \in \mathbb{R}_{\max}$,
$$a \oplus \epsilon = \epsilon \oplus a = a$$
4. *Existence of unit element.* For all $a \in \mathbb{R}_{\max}$,
$$a \otimes e = e \otimes a = a$$
5. *Distributivity of $\otimes$ over $\oplus$.* For all $a, b, c \in \mathbb{R}_{\max}$,
$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$
and
$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c).$$

6. *Zero element is absorbing.* For all $a \in \mathbb{R}_{\max}$,

$$a \otimes \epsilon = \epsilon \otimes a = \epsilon$$

Note that the only difference between a ring and a semiring is that a semiring does not require additive inverses. Max-plus algebra does not have them, since equations like $\max\{5, x\} = \epsilon$ have no solution for x in $\mathbb{R}_{\max}$.

By convention, to simplify notation, $\otimes$ has priority over $\oplus$. For instance:

$$a \oplus b \otimes c = a \oplus (b \otimes c).$$

For $a_1, a_2, \ldots, a_n \in \mathbb{R}_{\max}$, the sum of the a_i's is denoted $\bigoplus_{i=1}^{n} a_i$. That is,

$$\begin{aligned} \bigoplus_{i=1}^{n} a_i &= \max\{r_1, \ldots r_n\} \\ &= \max_{1 \le i \le n} \{r_i\}. \end{aligned}$$

Examples The following simple examples illustrate the features of max-plus algebra:

$$\begin{aligned} &4 \oplus \pi = \max\{4, \pi\} = 4, \\ &3 \otimes 2 = 3 + 2 = 5, \\ &3 \oplus \epsilon = \max\{3, \epsilon\} = 3 \\ &3 \otimes \epsilon = 3 - \infty = -\infty = \epsilon \\ &3 \otimes 4 \oplus 10 \otimes -5 = \max\{3 + 4, 10 - 5\} = \max\{7, 5\} = 7 \end{aligned}$$

2.2 Matrices over max-plus

The set of $n \times m$ matrices with entries in $\mathbb{R}_{\max}$ is denoted $\mathbb{R}_{\max}^{n \times m}$. Given $A \in \mathbb{R}_{\max}^{n \times m}$, the element of A in row i and column j is denoted $[A]_{ij}$ or a_{ij} for $1 \le i \le n, 1 \le j \le m$.

The sum of matrices $A, B \in \mathbb{R}_{\max}^{n \times m}$, denoted $A \oplus B$, is defined by

$$\begin{aligned} [A \oplus B]_{ij} &= a_{ij} \oplus b_{ij} \\ &= \max\{a_{ij}, b_{ij}\}. \end{aligned}$$

The product of matrices $A \in \mathbb{R}_{\max}^{n \times l}$ and $B \in \mathbb{R}_{\max}^{l \times m}$, denoted $A \otimes B$, is defined by

$$\begin{aligned} [A \otimes B]_{ij} &= \bigoplus_{j=1}^{l} a_{ij} \otimes b_{jk} \\ &= \max_{1 \le j \le l} \{a_{ij} + b_{jk}\}, \end{aligned}$$

The $n \times m$ matrix with all entries equal to ϵ is denoted $\mathcal{E}(n,m)$, or simply $\mathcal{E}$, when its size is implicit. The matrix denoted $E(n,m)$ is defined by

$$[E(n,m)]_{ij} = \begin{cases} e & \text{if } i = j, \\ \epsilon & \text{otherwise.} \end{cases}$$

As with $\mathcal{E}$, $E(n,m)$ may be written E when its dimensions are clear from the context.

For a matrix $A \in \mathbb{R}_{\max}^{n \times n}$ and $k \in \mathbb{Z}^+$, the kth power of A, denoted $A^{\otimes k}$, is defined by

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

The matrix $A^{\otimes 0}$ is defined to be $E(n,n)$.

Examples The following are examples of matrix operations in $\mathbb{R}_{\max}$:

$$\begin{aligned} \begin{pmatrix} 6 & 2 \\ e & \epsilon \end{pmatrix} \oplus \begin{pmatrix} 5 & 3 \\ 4 & e \end{pmatrix} &= \begin{pmatrix} 6 \oplus 5 & 2 \oplus 3 \\ e \oplus 4 & \epsilon \oplus e \end{pmatrix} \\ &= \begin{pmatrix} 6 & 3 \\ 4 & e \end{pmatrix} \end{aligned}$$

and

$$\begin{aligned} \begin{pmatrix} 6 & 2 \\ e & \epsilon \end{pmatrix} \otimes \begin{pmatrix} 5 & 3 \\ 4 & e \end{pmatrix} &= \begin{pmatrix} 6 \otimes 5 \oplus 2 \otimes 4 & 6 \otimes 3 \oplus 2 \otimes e \\ e \otimes 5 \oplus \epsilon \otimes 4 & 3 \otimes 3 \oplus \epsilon \otimes e \end{pmatrix} \\ &= \begin{pmatrix} 11 \oplus 6 & 9 \oplus 2 \\ 5 \oplus \epsilon & 6 \oplus \epsilon \end{pmatrix} \\ &= \begin{pmatrix} 11 & 9 \\ 5 & 6 \end{pmatrix} \end{aligned}$$

2.3 Petri nets

Petri nets are a common tool for modeling systems where events can only occur when certain conditions are met, and a certain class of Petri nets, called timed event graphs, have a particularly useful analytical interpretation in max-plus algebra.

Formally, a Petri net is a bipartite digraph; that is, a directed graph whose finite set of nodes can be partitioned into two disjoint subsets, $\mathcal{P}$ and $\mathcal{Q}$. Elements of $\mathcal{P}$, denoted p_i, $i = 1, 2, \ldots, |\mathcal{P}|$, are called *places*, and elements of $\mathcal{Q}$, denoted q_i, $i = 1, 2, \ldots, |\mathcal{Q}|$, are called *transitions*. The set of arcs, $\mathcal{D}$, is a subset of $(\mathcal{P} \times \mathcal{Q}) \cup (\mathcal{Q} \times \mathcal{P})$. That is, arcs can go from places to transitions and from transitions to places, but not from places to places or transitions to transitions. Given an arc $(p_i, q_j) \in \mathcal{D}$, we say that p_i is an upstream place for q_j, and q_j is a downstream transition for p_i. Analogous terminology applies for $(q_i, p_j) \in \mathcal{D}$.

Typically, when Petri nets are used to model a system, places represent conditions and transitions represent events (although this will not always be adhered to, even within this thesis). To represent the fulfillment of conditions, any number of

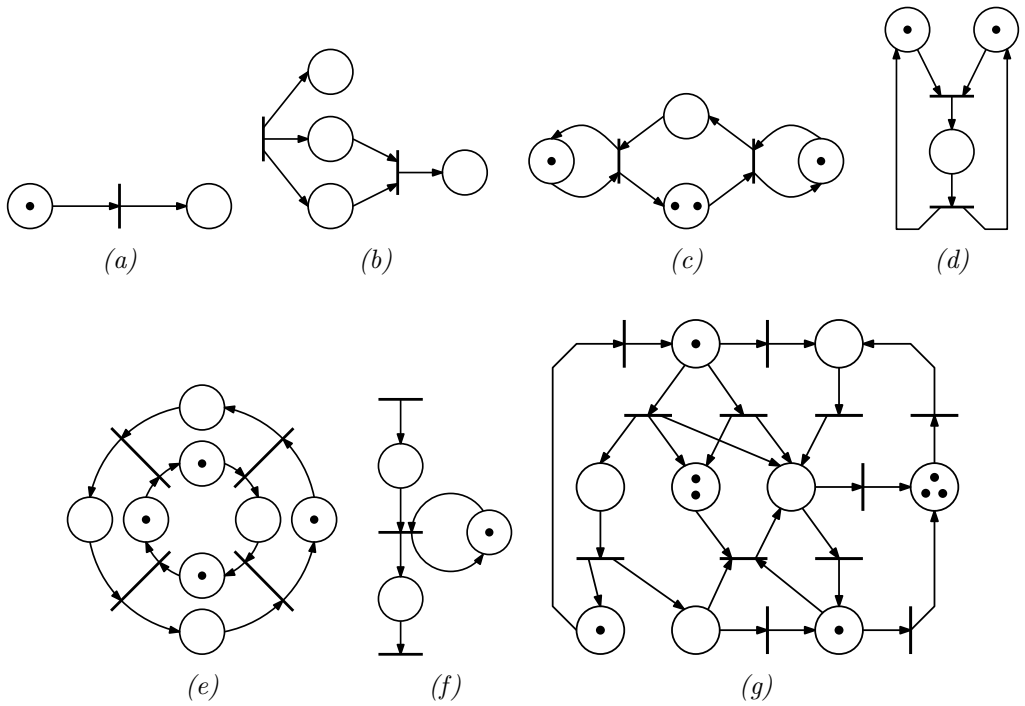


Figure 2.1: Various Petri nets. Figures (c), (d) and (e) are autonomous timed event graphs. Figure (f) is a nonautonomous timed event graph.

tokens may be allocated to each place in a Petri net. When every place upstream from a transition has a token in it, the transition is said to be *enabled*, and the transition can *fire*—when all the conditions for an event are met, the event takes place. When a transition fires, one token is removed from each place upstream of the transition, and one token is added to each place downstream of it. The distribution of tokens through the Petri net is called a *marking*. The *initial marking* of a Petri net, how tokens are assigned initially, is specified by a function $\mathcal{M}_0 : \mathcal{P} \to \{0, 1, 2, 3 \ldots\}$, where $\mathcal{M}_0(p_i)$ is the initial number of tokens at place p_i.

In diagrams of Petri nets, places are indicated by circles, transitions by straight line segments, arcs by arrowed lines, and tokens by dots within place circles. See Figure 2.1 for examples of Petri net diagrams.

To each place p_i in a Petri net we may also assign a *holding time*, t_i, which forces a token to stay at that place for a certain amount of time, preventing a downstream transition from firing until all the holding times of its upstream places have passed, even if the transition is enabled. This is why "enabled" is distinct from "firing". Note that firing is instantaneous. Also, each token in a place has its own timer. That is, if there are two tokens in a place, and one of them arrived in the place before the other, it will be able to fire a transition before the other. The holding

times associated with each place are indicated by a vector $\mathcal{T}$, where

$$\mathcal{T} = \begin{pmatrix} t_1 \\ \vdots \\ t_{|\mathcal{P}|} \end{pmatrix}.$$

A Petri net with a holding time associated with each place is called a *timed Petri net*. For convenience, places in Petri net diagrams may be labeled with each place's holding time instead of their p_i identification.

A Petri net is called an *event graph* if every transition has exactly one upstream and one downstream place. An event graph with holding times is called a *timed event graph*. Timed event graphs have a nice characterization in max-plus algebra, which will be the topic of the next section. In fact, timed event graphs are the only kind of Petri net that play nice with max-plus.

It is possible for a transition in a Petri net to have no upstream or no downstream places. In the former case, the transition may fire at times given by a function $u(k)$. A Petri net with no such transitions, that is a Petri net where every transition has at least one upstream and at least one downstream place, is called *autonomous*. A Petri net that is not autonomous is said to be *nonautonomous*.

Timed event graphs are often given as a 5-tuple, $(\mathcal{P}, \mathcal{Q}, \mathcal{D}, \mathcal{M}_0, \mathcal{T})$, which specifies the structure of the Petri net, but is not enough to determine the behavior of the system—the future firing times of its transitions—entirely. For that one needs an initial state vector. For a Petri net, the kth state vector

$$\mathbf{x}(k) = \begin{pmatrix} x_1(k) \\ \vdots \\ x_{|\mathcal{Q}|}(k) \end{pmatrix}$$

specifies the time of the kth firing of each transition. That is, transition q_i fires for the kth time at $x_i(k)$. Thus, the initial state vector $\mathbf{x}(0)$ gives the time that each transition fires for the first time. Together, $(\mathcal{P}, \mathcal{Q}, \mathcal{D}, \mathcal{M}_0, \mathcal{T})$ and $\mathbf{x}(0)$ determine entirely the behavior of a timed event graph for $t \geq 0$.

An example

Figure 2.2 shows a timed Petri net with the following specifications:

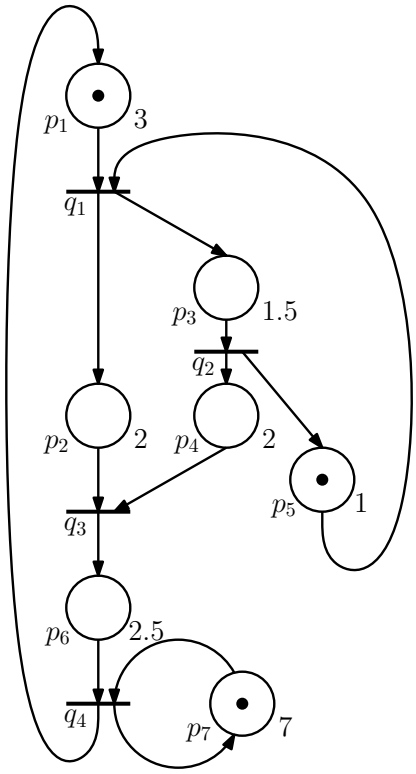


Figure 2.2

1. $\mathcal{P} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$
2. $\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$
3. $\mathcal{D} = \{(p_1, q_1), (p_2, q_3), (p_3, q_2), (p_4, q_3), (p_5, q_1), (p_6, q_4), (p_7, q_4), (q_1, p_2), (q_1, p_3), (q_2, p_4), (q_2, p_5), (q_3, p_6), (q_4, p_1), (q_4, p_7)\}$
4. $\mathcal{M}_0 = \begin{cases} 1 & \text{for } p_1, p_4, p_5, p_7 \\ 0 & \text{for } p_2, p_3, p_6 \end{cases}$
5. $\mathcal{T} = (3, 2, 1.5, 2, 1, 2.5, 7)^\top$

If we also specify the following initial state vector,

6. $\mathbf{x}(0) = (\epsilon, \epsilon, \epsilon, 0)^\top$

where the transtion q_4 fires at $t = 0$,[1] the timed event graph in Figure 2.2 will behave in the following way.

- At $t = 3$, transition q_1 will fire, removing tokens from p_1 and p_5 and adding tokens to p_2 and p_3. This will enable q_2.
- At $t = 4.5$, transition q_2 will be able to fire, and tokens will be removed from p_3 and added to p_4 and p_5. Transition q_3 will then be enabled.

[1]The ϵ's in the initial state vector mean that transitions q_1, q_2, and q_3 have never fired.

- After p_4's holding time passes, q_3 will fire at $t = 6.5$.
- At $t = 9$, q_4 will fire, returning a token to p_1. The token at p_7 will be instantaneously removed and replaced.

2.4 Autonomous timed event graphs and max-plus

As mentioned above, timed event graphs can be analyzed with max-plus algebra such that the firing times of transitions can be generated by a recurrence equation. The details of the derivation of these equations are provided in [1].

Given a timed event graph, $(\mathcal{P}, \mathcal{Q}, \mathcal{D}, \mathcal{M}_0, \mathcal{T})$, with maximum initial marking with respect to all places M, matrices A_0, A_1, ... A_M of size $|\mathcal{Q}|\times|\mathcal{Q}|$ are constructed as follows. For transitions q_j and q_i, $[A_m]_{ij}$ is taken to be the maximum of the holding times for places between q_j and q_i with m tokens initially. That is, if there are n places, $p_1, \ldots, p_n$, each with m tokens initially, and holding times $t_1, \ldots, t_n$ between q_j and q_i, then

$$[A_m]_{ij} = \begin{cases} \max\{t_1, \ldots, t_n\} & \text{if } \mathcal{M}_0(p_{q_j q_i}) = m \\ \epsilon & \text{otherwise} \end{cases}$$

Next we generate a matrix

$$A_0^* = \bigoplus_{i=0}^{|\mathcal{Q}|-1} A_0^i$$

a new state vector,

$$\tilde{x}(k) = \begin{pmatrix} x^\top(k) \\ x^\top(k-1) \\ \vdots \\ x^\top(k-M+1) \end{pmatrix}^\top$$

and the $(|\mathcal{Q}|\times M) \times (|\mathcal{Q}|\times M)$ matrix

$$\tilde{A} = \begin{pmatrix} A_0^* \otimes A_1 & A_0^* \otimes A_2 & \cdots & A_0^* \otimes A_{M-1} & A_0^* \otimes A_M \\ E & \mathcal{E} & \cdots & \mathcal{E} & \mathcal{E} \\ \mathcal{E} & E & \cdots & \mathcal{E} & \mathcal{E} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathcal{E} & \mathcal{E} & \cdots & E & \mathcal{E} \end{pmatrix}.$$

Now we can predict the behavior of the timed event graph with this, the *standard autonomous equation*:

$$\tilde{x}(k) = \tilde{A} \otimes \tilde{x}(k-1).$$

Since Figure 2.2 is an autonomous timed event graph, we can use it as an example. This maximum number of initial tokens for all places is one, so we have

matrices A_0 and A_1:

$$A_0 = \begin{pmatrix} \epsilon & \epsilon & \epsilon & \epsilon \\ 1.5 & \epsilon & \epsilon & \epsilon \\ 2 & 2 & \epsilon & \epsilon \\ \epsilon & \epsilon & 2.5 & \epsilon \end{pmatrix},$$

$$A_1 = \begin{pmatrix} \epsilon & 1 & \epsilon & 3 \\ \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 7 \end{pmatrix},$$

and

$$A_0^* = \begin{pmatrix} e & \epsilon & \epsilon & \epsilon \\ 1.5 & e & \epsilon & \epsilon \\ 3.5 & 2 & e & \epsilon \\ 6 & 4.5 & 2.5 & e \end{pmatrix}.$$

Since $M = 1$, $|\mathcal{Q}| \times M = |\mathcal{Q}|$, so $\tilde{A}$ is simply

$$\tilde{A} = \begin{pmatrix} \epsilon & 1 & \epsilon & 3 \\ \epsilon & 2.5 & \epsilon & 4.5 \\ \epsilon & 4.5 & \epsilon & 6.5 \\ \epsilon & 6.5 & \epsilon & 9 \end{pmatrix}.$$

Further, $k - M + 1 = k$, which means

$$\tilde{x}(k) = (\mathbf{x}^\top(k))^\top = \mathbf{x}(k).$$

We can now use the standard autonomous equation to determine all future firing times for transitions in Figure 2.2. For instance,

$$\mathbf{x}(1) = \tilde{A} \otimes \mathbf{x}(0) = \begin{pmatrix} \epsilon & 1 & \epsilon & 3 \\ \epsilon & 2.5 & \epsilon & 4.5 \\ \epsilon & 4.5 & \epsilon & 6.5 \\ \epsilon & 6.5 & \epsilon & 9 \end{pmatrix} \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 3 \\ 4.5 \\ 6.5 \\ 9 \end{pmatrix}$$

Note that this agrees with the firing times described in the walkthrough in Section 2.3.

3 Max-Plus and Transportation

3.1 Previous max-plus/Petri net models

The exemplar of modeling transportation systems with Petri nets and max-plus algebra is trains. One well-worked-out example can be found in [1], where such a model is applied to the Dutch railway system. In this model, transitions represent stations, a transition firing represents a train leaving from a stations, a token in a place represents a train traveling or dwelling at a station, and the corresponding holding time for that place is sum of the travel time and dwell time. Figure 3.2b shows a sample with three connecting train lines.

The focus in this model is on modeling the timing of trains departing from stops and the synchronization of connections between train lines. There is no consideration of passengers, and no need: trains serving connecting stations are synchronized, whereas buses that serve connecting stops are not. Also, the dwell time for passengers to catch connecting trains is incorporated into the scheduling, so there is little opportunity for passengers to cause delay. Buses typically do not have this dwell time at a given stop. Though this train model is not totally appropriate for modeling bus lines, it usually forms a basis for them.

For instance, [2] models a single bus line using a simple Petri net circuit similar to the one in Figure 3.1b. This model is a timed event graph, and is therefore subject to the algebraic analytical tools of max-plus; this approach is useful for evaluating the feasibility of timetables and scheduling for a given network, but ignores totally the passenger element and assumes the system runs smoothly at all times.

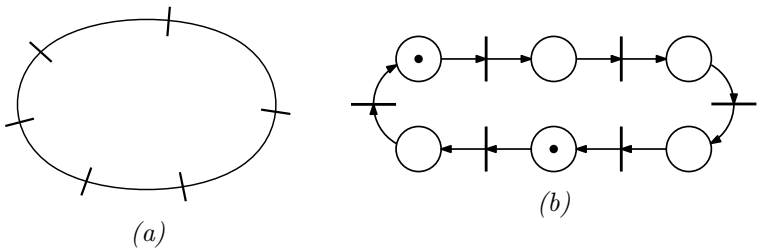


Figure 3.1: A basic line (a) and its Petri net interpretation (b). Tokens represent buses or trains traveling counterclockwise around the line, and transitions represent stops, firing when a bus or train departs or arrives. Most attempts to model bus networks with Petri nets and max-plus algebra build off this simple model, either by linking together multiple lines or by expanding stop transitions into subnets.

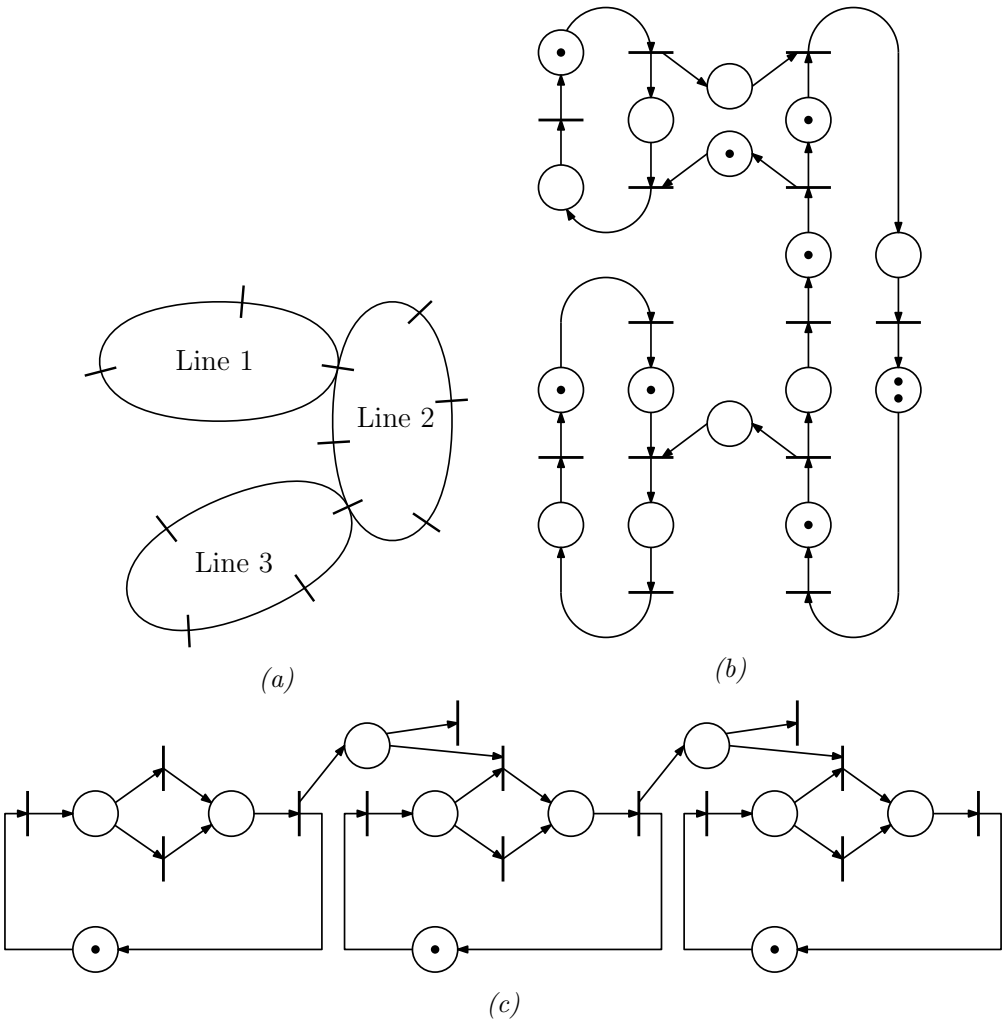


Figure 3.2: A generalized diagram of a multiline transportation network, (a), where vehicles travel clockwise around each line, and two different Petri net interpretations.

Another model, one that nods at the presence of passengers, can be found in [3]. This article studies n bus lines with connecting stops between adjacent lines where passengers can transfer from one line to the next. Figure 3.2c shows a three line version. The lines are not synchronized, so buses can move freely around them independent of bus movements in other lines. Note that the Petri net is not a timed event graph. Some of the places in the net have more than one downstream transition. This creates structural conflicts that the authors solve by incorporating a routing policy that determines which transition fires at what times when both transitions are enabled. The authors *do* use max-plus algebra to analyze this system, but the techniques of section 2.4 are not available.

None of the models heretofore mentioned allow for the possibility of passenger activity causing delay in the bus route.

3.2 Modeling passenger interaction

Examining the literature on modeling passenger interaction with bus routes, [4] concludes:

> Generally the existing models look something like this: the departure time of a bus from a stop is given by
>
> (departure time from previous stop) + (travel time to stop) + (time spent waiting at stop)
>
> It is the *(time waiting at stop)* term that depends heavily on passenger activity.

Concordantly, the authors develop a Petri net model where each stop in a bus line is expanded into a Petri subnet designed to capture the following behaviors:

1. After a bus leaves a stop, passengers begin accumulating at a stop.
2. When a bus arrives, passengers stop accumulating and begin boarding, which takes more or less time depending on how many passengers are there.

Figure 3.3 shows a diagram of their model with two stops. The net breaks down as follows.

- Transitions a_1 and a_2 are called *arrival transitions*, and represent a bus arriving at a respective stop.
- Transitions d_1 and d_2 are *departure transitions*, and represent a bus departing from a stop.
- The arcs going from p_1 and p_2 with circles at the end instead of arrows are called *inhibitor arcs*. Instead of allowing a transition to fire when a token is in place, they *prevent* it from firing.
- The places marked $t_{i,j}$, where i is the stop and j identifies the place, along with the transitions between the $t_{i,j}$ places, make up the *passenger arrival net*. As the Petri net runs, the token at $t_{1,1}$ in Figure will move rightward, which represents the arrival of passengers at the stop. The holding times of these places correspond to the rate at which passengers arrive.
- The places marked $b_{i,j}$, with the transitions between them, make up the *passenger boarding net*. After a bus arrives, the token from the passenger arrival net will move downward to the boarding net, where its leftward progress represents passengers boarding the bus. As above, the holding times reflect how quickly passengers board the bus.

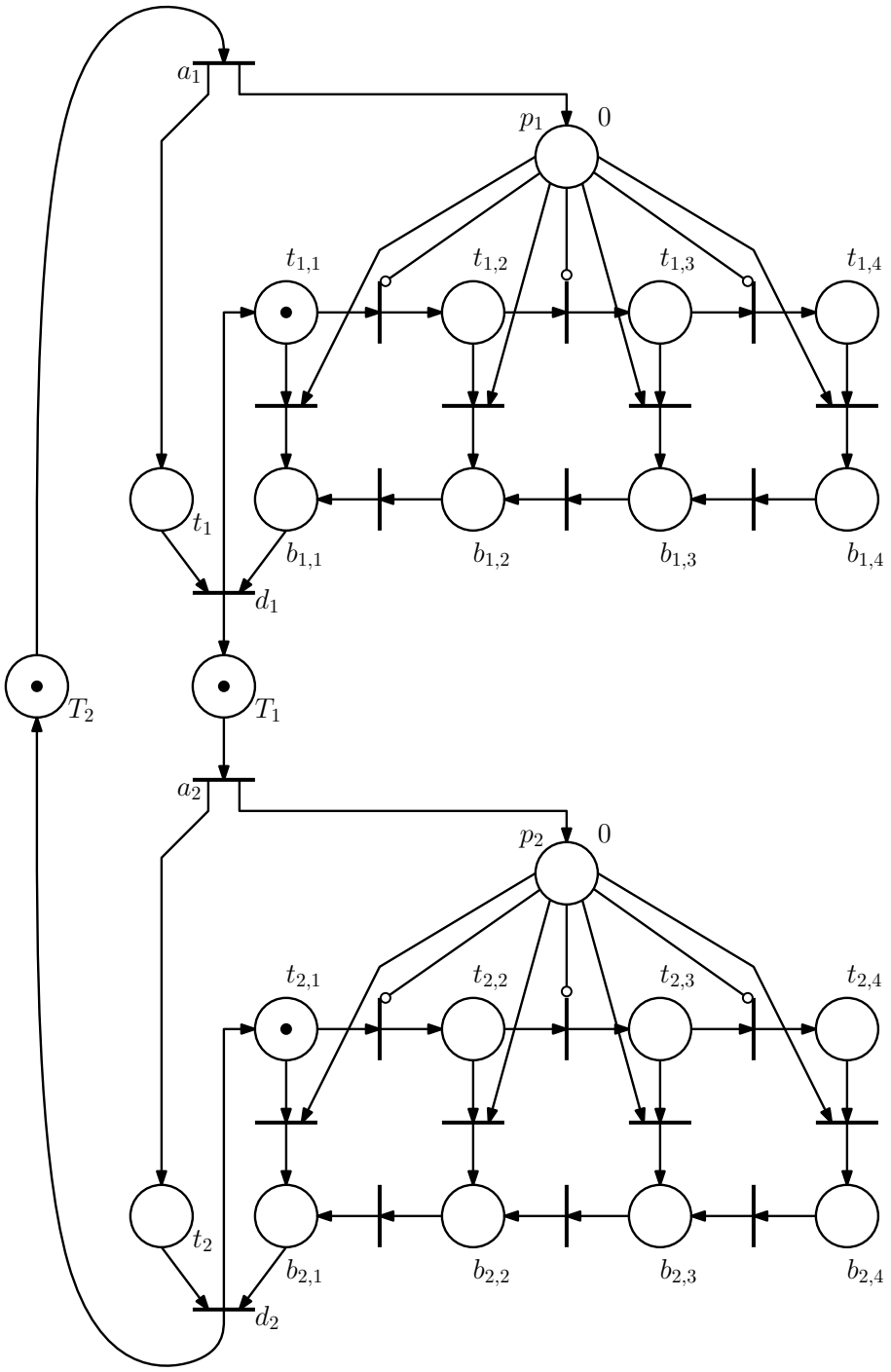


Figure 3.3: A Petri net model that incorporates passenger-caused delay. For the sake of clarity, some places have been relabeled from the original diagram in [4].

- The places t_1 and t_2 are *buffer places*, which serve to ensure that buses are not allowed to leave the stop early.

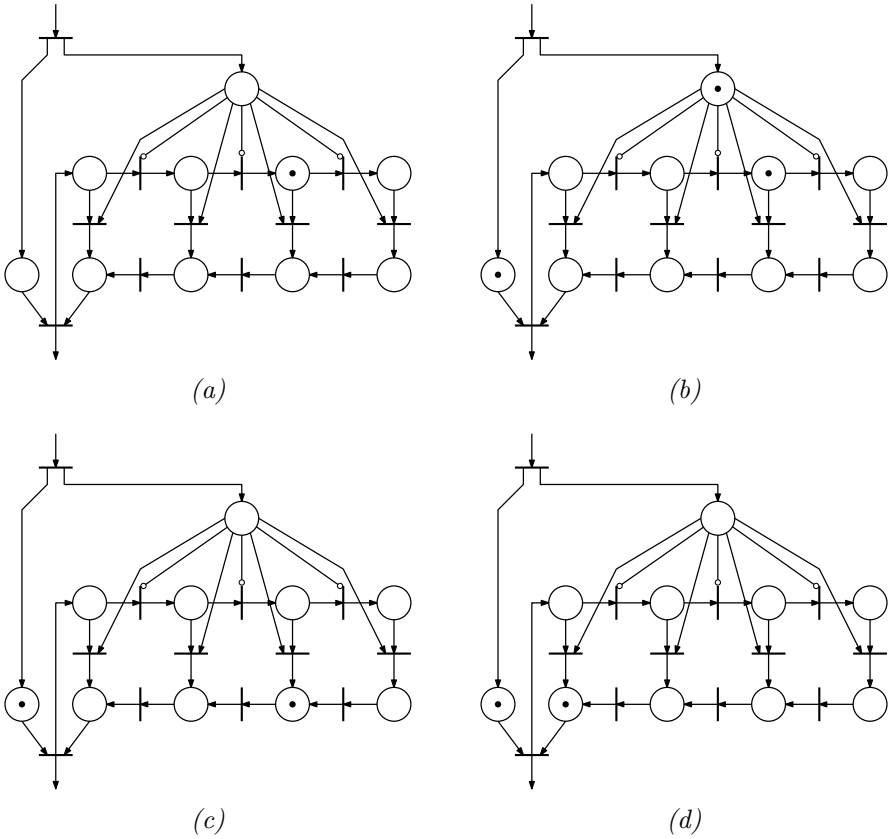


Figure 3.4: The evolution of the stop subnet in Figure 3.3 over time.

- The places marked T_1 and T_2, when occupied by tokens, represent respectively a bus traveling from stop one to stop two with travel time T_1, and a bus traveling from stop two to stop one, with travel time T_2.

This model runs as follows.

1. After d_1 fires and a bus leaves a stop, the stop subnet will be as it is in Figure 3.3.

2. As time passes and no bus arrives at a_1, the token which began at $t_{1,1}$ will begin to move rightward, Figure 3.4a.

3. When a bus arrives, a token placed in p_1 prevents the token representing passengers arriving from continuing to the right, Figure 3.4b.

4. Wherever the passenger token is, it will be sent down to the boarding subnet, and the token at p_1 will be removed, Figure 3.4c.

5. The passenger token will move leftward, until it arrives at $b_{1,1}$, Figure 3.4d.

6. When this happens, d_1 will be able to fire, removing the token from $b_{1,1}$ and adding a token to $t_{1,1}$, thus restoring the stop net to its configuration in Figure 3.3. A token is also added to T_1, as the bus it represents begins to travel to the next stop.

Note that the token at $t_{i,1}$ does not represent a passenger. Rather, its position in the subnet indicates how many passengers are currently at the stop. The further to the right it moves, the more passengers there are. This needn't be interpreted in a limited way. I.e., when a transition fires and the token moves right, this doesn't necessarily equate to the arrival of a single passenger. It could be a fraction of a passenger, or multiple passengers, depending on the specifics of the stop that is being modeled.

Also, though Figure 3.3 only goes up to $t_{i,4}$ and $b_{i,4}$, the arrival and boarding subnets can be extended arbitrarily to allow for more passengers, or to more finely track arrivals. Holding times can also be adjusted individually to account, for instance, for changes in the arrival rate of passengers.

Interestingly, reducing the number of arrival and boarding places in each stop to 1, as in Figure 3.5, eliminates the inhibitor arcs and creates a timed event graph; but it also eliminates the delay created by passengers. As we'll see in section 4.2, this is no coincidence.

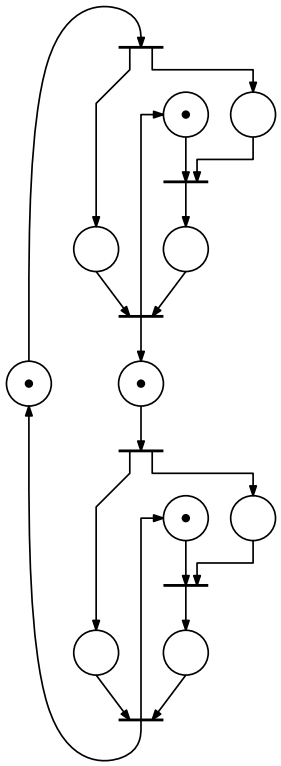


Figure 3.5

4 Attempts to model bus bunching in max-plus

4.1 A model without inhibitor arcs

There is a constant tension between creating a model that more precisely replicates a phenomenon, and one that is subject to easy analysis. The model from [4] takes us two steps from the simplicity of timed event graphs—the first in using inhibitor arcs and the second in incorporating places with more than one upstream or downstream transition—and therefore from the tools of max-plus algebra. Though a later result will show it is not possible to model passenger delay within a stop when limited to autonomous timed event graphs, it is possible to restructure the model used in [4] so that it does not include inhibitor arcs, with only a minimal change in function. A diagram of this new model with two stops is shown in Figure 4.1.

The model is mostly the same as in Figure 3.3, and the same labeling is used; however, a new place has been added to each stop.

- The places r_1 and r_2 with holding times of 0 replace the inhibitor arcs in Figure 3.3. The procedure for reconstructing stops without inhibitor arcs is as follows. Add a place r_i, and an arc from r_i to a_i. Replace each inhibitor arc from p_i to a transition with both an arc from the transition to r_i and an arc from r_i to the transition. In Figure 4.1, these arcs are represented with double arrows. Also, for every arc from p_i to a transition, add an arc from that transition to r_i.

This altered model runs as follows:

1. After d_1 fires, that is after a bus leaves stop 1, there are tokens at $t_{1,1}$ and at r_1.
2. After $t_{1,1}$ units of time pass, the token there moves right. The token at r_1 disappears and is immediately replaced when this happens, allowing the passenger token to continue moving right.
3. When a_1 fires, the token is removed from r_1 and one is placed in p_1.

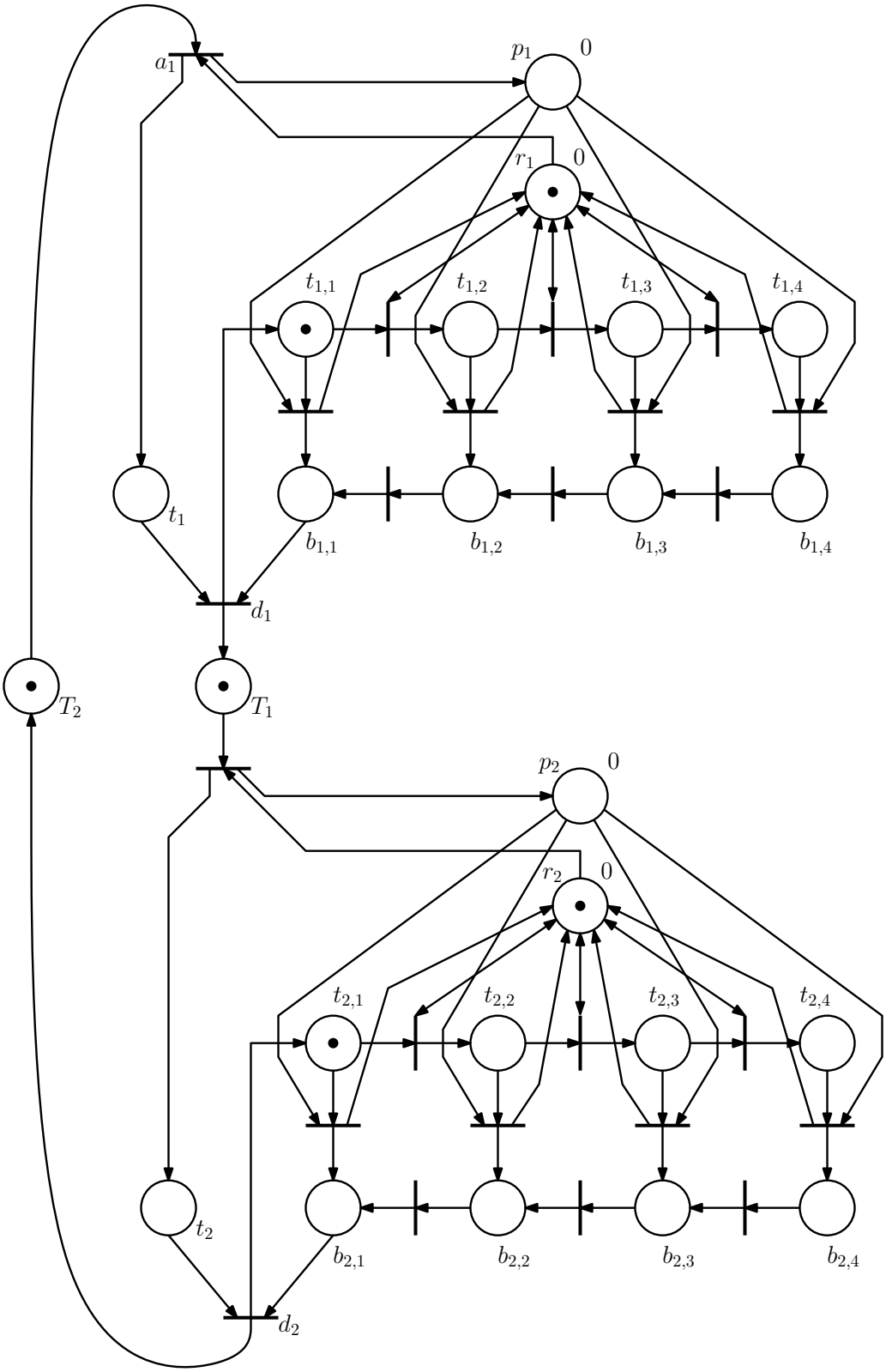


Figure 4.1: A version of the model in Figure 3.3 that has been altered to eliminate inhibitor arcs.

4. With r_1 empty, the token in the passenger arrival subnet can no longer move rightward, but can move down to the passenger loading subnet. When this happens the token at p_1 is removed and a token is added to r_1. The passenger loading subnet runs precisely as in the model with inhibitor arcs.

5. After the passenger token reaches $b_{1,1}$, the transition d_1 is allowed to fire, which removes the tokens at t_1 and $b_{1,1}$, and replaces the token at $t_{1,1}$, resetting the stop so that passengers can begin accumulating and the next bus can arrive.

A notable difference in the performance of this model is that a bus token cannot enter the subnet until the token at r_1 is replaced, since its absence prevents a_1 from firing.

4.2 Stop subnets in autonomous timed event graphs

Though it would be desirable to model passenger delay in an autonomous timed event graph (so that the tools of max-plus are available) and though it makes sense that this behavior should be modeled by creating stop subnets that generate delay based on when tokens representing buses pass through them (since this kind of passenger interaction takes place at stops)—in other words, it would be desirable to model bunching by stringing together stops of the kind generalized in Figure 4.2—the demonstration below will show that this is not possible. The above strictures are too limiting.

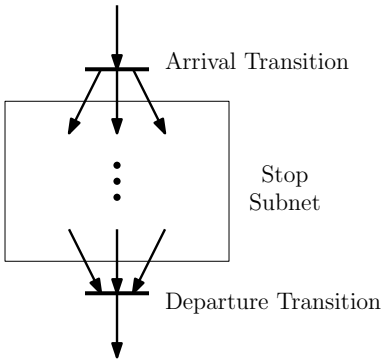


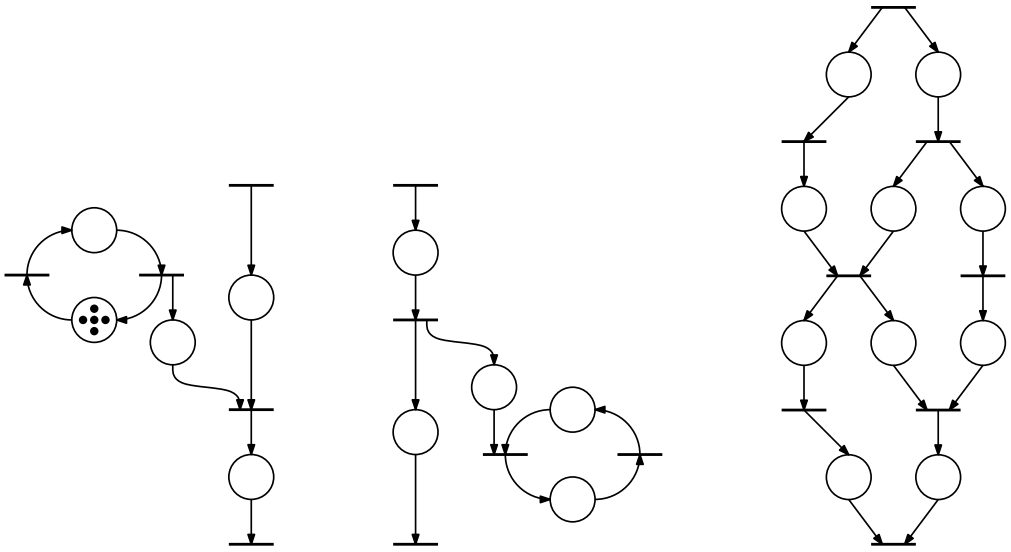
Figure 4.2

It will be useful here to have some notion of a path through a Petri net. A *path*, intuitively, is a sequence of places, transitions, and arcs through a Petri net, beginning and ending at a place or transition. For instance, the sequence

$$q_1 \to p_3 \to q_2 \to p_5 \to q_1 \to p_2 \to q_3$$

is a path through the Petri net in Figure 2.2. Note that this path loops back around and goes through q_1 twice.

Let's consider what must be true of the inside of a subnet. Since tokens outside the stop subnet represent buses, they must be able to "pass through" the stop



(a) *(b)* *(c)*

Figure 4.3

subnet. In the actual functioning of the subnet, the bus token may be split up repeatedly, but only one token can come in and only one can come out. Thus, there must be at least one path from the arrival transition to the departure transition. Further, the bus token must be allowed through. That is, the internals of the stop subnet must preclude situations where buses can never leave.

Within a stop subnet, there can be four kinds of paths:

1. Paths that begin in a circuit in the subnet and end at the departure transition, Figure 4.3a.
2. Paths that begin at the arrival transition and end with a circuit, Figure 4.3b.
3. Paths that begin at the arrival transition and end at the departure transition with no repeating places or transitions, like every path featured in Figure 4.3c.
4. Paths that begin at the arrival transition and end at the departure transition, *with* repeating places and transitions, Figure 4.4.

We can ignore types 1 and 2. Type 1 requires tokens to allow the stop to work, and since the initial marking can only be finite, the stop must eventually stop allowing buses to pass through it. Type 2 we can ignore because it does not affect the timing bus tokens passing through the stop subnet whatsoever.

This leaves types 3 and 4. If a stop subnet contains only paths of type 3, then the amount of time it takes for a bus token to pass from arrival to departure transitions is constant, being simply the maximum of the sum of the holding times for all the paths. The time it takes does not change depending on when a bus last passed through the stop, so no delay is possible.

If, however, a stop contains type 4 paths, then the timing does change, but in the wrong way. Rather than more time between buses leading to more time spent at the stop, more time means less time spent at the stop. Consider Figure 4.4. Figure 4.4a

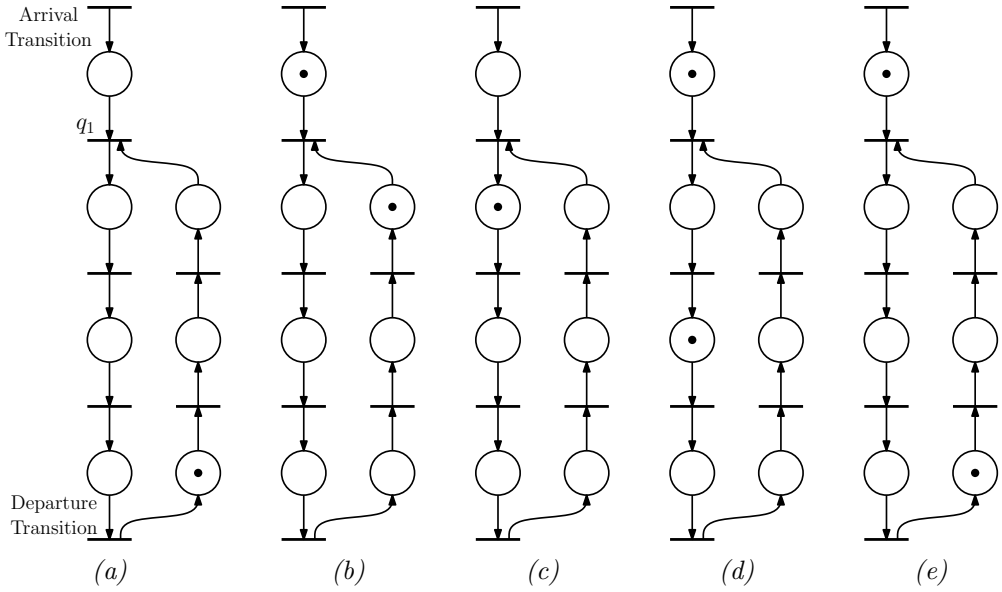


Figure 4.4

shows such a stop just after a bus has departed. Note both that there is a type 3 path going straight from the arrival transition to the departure transition, and that a path from downstream to upstream on it creates the type 4 path. Note also that there is a token in the subnet. This is required; if it were absent, the subnet would not work, as buses would not be able to pass through. If enough time passes, when another bus comes, this upstream path will have "reset", as in Figure 4.4b and the new bus token will be able to pass through transition q_1 with no delay, leading to Figure 4.4c. But then, if another bus arrives at the stop soon after, the new bus will be held up at q_1, forced to wait for the upstream path to "reset", as in Figures 4.4d and 4.4e. That is, passing through the stop will take *more time* for a bus that arrives *sooner* after another bus. This is exactly the opposite of what we want!

Obviously the Petri net in Figure 4.4 is very simple, but the same result occurs with more complex stop subnets containing type 4 paths. The path that goes back upstream holds up bus tokens that arrive before the upstream path resets. Rather than more time between bus tokens arriving meaning more delay at a stop, it means less. There is no way to construct a stop subnet that is an autonomous timed event graph where more time between bus tokens means more time spent at the stop. Thus when trying to model passenger effects with Petri nets, we can decisively rule out such models. However, we cannot rule out models that are nonautonomous, models where stops are not isolated, or models that do not feature stop subnets at all.

5 Conclusion

In this thesis, Petri nets and max-plus algebra were presented as a way of trying to model the phenomenon of bus bunching. One kind of model was ruled out, but there remains the possibility of modeling bunching with Petri nets that are nonautonomous or are not timed event graphs, even if they would be less useful. Another model was simplified to remove untractable elements, in the hope that it may be more easily analyzed, though no analysis was performed on it in this paper.

Envoi

"On 12 July of the year 1910 after introducing a new factor, I succeeded in flawlessly solving the problem. Alas, I can say nothing about it without invalidating its registration at the patent offices of various governments. But I did arrive at a satisfying conclusion."

Bibliography

[1] B. Heidergott, G. J. Olsder, and J. van der Woude. *Max Plus at Work*. Princeton University Press, 2006.

[2] Z. R. Königsberg. A generalized eigenmode algorithm for reducible regular matrices over the max-plus algebra with applications to the metro-bus transport system in mexico city. *Nonlinear Analysis: Hybrid Systems*, 2(4):1205–1216, 2008.

[3] A. Nait-Sidi-Moh, M.-A. Manier, and A. E. Moudni. Spectral analysis for performance evaluation in a bus network. *European Journal of Operational Research*, 193:289–302, 2009.

[4] T. Shelly and G. Lafferriere. Analysis of bus modeling via max-plus algebra and simulations: A report. 2010.

[5] T. L. Shelly. Max-plus algebra and generalized eigenmodes of reducible regular matrices. 2010.