Supervisory Controller Design to Enforce Desired Properties in Systems Modeled by Timed-arc Petri Nets

ALTUĞ İFTAR
Department of Electrical and Electronics Engineering
Eskişehir Technical University
26555 Eskişehir, TURKEY

Abstract: - Discrete-event systems which are modelled by timed-arc Petri nets are considered. A supervisory controller design approach for such systems for enforcing some desired properties is proposed. For the representation of the state of the system, the proposed design approach uses stretching. The designed controller enforces $\hat{L}$-boundedness (for any given bound vector $\hat{L}$) and reversibility simultaneously, whenever it is possible to design such a controller. Furthermore, Δ-liveness is also enforced, where Δ is the largest possible subset of the set of transitions for which it is possible to design such a controller. Furthermore, the designed controller is maximally permissive in the sense that no transitions are disabled unnecessarily and the reachability set of the controlled system is the largest possible set in which boundedness and reversibility can be enforced simultaneously. To demonstrate the proposed approach, an example controller design is also presented for an automated manufacturing system.

Key-Words: - Discrete-event Systems, Timed-arc Petri Nets, Supervisory Controller Design, Boundedness, Reversibility, Liveness, Manufacturing Systems, Automation

Received: April 28, 2023. Revised: February 26, 2024. Accepted: April 9, 2024. Published: May 29, 2024.

1 Introduction

Many discrete-event systems (DES) can be modelled using Petri nets, [1], [2]. Early models of Petri nets did not include the notion of time. However, later it became apparent that this notion is needed to appropriately model many DES, and timed Petri nets (TPNs) were introduced to describe the evolution of these systems throughout time, [3], [4], [5], [6], [7], [8], [9]. A TPN, besides the basic elements of an untimed Petri net, also includes certain time-delays. In a Petri net time-delays can be included in the transitions, [10], [11], [12], or in the places, [13]. In a Petri net model, transitions and places respectively represent the events and the resources. Therefore, the time needed for an event to occur may be represented by a time-delay in the corresponding transition and the time needed for a resource to become available may be represented by a time-delay in the corresponding place. Thus, when both events and resources may require certain times, one may need a TPN model where both the transitions and the places are timed, [14]. However, as discussed in [15], even this formalism may not be sufficient. To overcome this problem, rather than transitions and places, time-delays can be included in the arcs, [16]. A Petri net in which arcs are timed is called a timed-arc Petri net (TAPN), [15].

The representation of the state of any dynamic system is necessary in order to simulate it or to design a controller for it. The marking vector is sufficient to represent the state of an untimed Petri net. However, this is not sufficient for a TPN, [16]. To overcome this difficulty, the method of stretching was first introduced for Petri nets in which the transitions are timed, [17], and then for Petri nets in which the places are timed, [18]. Petri nets in which both the transitions and places are timed are then considered in [14]. Finally, the method of stretching was introduced for TAPNs in [15].

Supervisory control is needed for DES to avoid undesirable behaviour, such as deadlock, [19], or to enforce desirable behaviour, such as boundedness, reversibility, and/or liveness, [20]. Controller

design to avoid deadlock for various types of TPNs were considered in [10], [13], [15], and to enforce some desirable properties were considered in [21], [22], [23]. In the present work, we consider supervisory controller design in TAPNs in order to enforce boundedness, reversibility, and liveness. As in previous works, we assume that a suitable time unit can be found which divides (at least approximately) all the process delays. This unit is then taken as the unit of time and all the time-delays are thus become an integer. Furthermore, the time variable, denoted by t, is also taken as integer and the initial time is assumed to be $t = 0$.

We present the necessary preliminaries in Section 2. The proposed supervisory controller design approach is presented in Section 3. An example of an automated manufacturing system and controller design for it are presented in Section 4. Finally, some concluding remarks are included in Section 5.

Throughout the paper, $\mathbf{Z}$ denotes the set of non-negative, and $\mathbf{Z}^+$ denotes the set of positive integers. For two vectors with the same dimension, x and y, $x \geq y$ ($x \leq y$) indicates that each element of x is greater (less) than or equal to the corresponding element of y. For a vector x, x^T denotes its transpose. The number of elements of a set Ω is denoted by $|\Omega|$ and its j^{th} element ($j = 1, \ldots, |\Omega|$) is denoted by $[\Omega]_j$.

2 Preliminaries

2.1 Timed-Arc Petri Nets

As presented in [15], A TAPN is a tuple $H = (\Pi, \Theta, A, B, S_0, \mathcal{D}_N, \mathcal{D}_O)$, where Π denotes the set of places, Θ denotes the set of transitions, $A : \Pi \times \Theta \to \mathbf{Z}$ is the matrix which indicates the weights of the arcs from places to transitions, $B : \Pi \times \Theta \to \mathbf{Z}$ is the matrix which indicates the weights of arcs from transitions to places (as usual, a zero entry in A or B means there is no arc between the corresponding place and transition), $S_0 : \Pi \to \mathbf{Z}$ indicates the initial number of tokens at each place. $\mathcal{D}_N : \mathcal{A}_N \to \mathbf{Z}$ indicates the time-delays of the input arcs, where $\mathcal{A}_N := \{(\pi, \theta) \mid A(\pi, \theta) > 0\}$ is the set of input arcs, and $\mathcal{D}_O : \mathcal{A}_O \to \mathbf{Z}^+$ indicates the time-delays of the output arcs, where $\mathcal{A}_O := \{(\theta, \pi) \mid B(\pi, \theta) > 0\}$ is the set of output arcs. Note that, the time-delay of an input arc may be zero or positive, however, the time-delay of an output arc must be positive. This is because, in a Petri net places usually correspond to resources and transitions usually correspond to events. While a resource may be readily available for any event, the execution of any event always takes some time.

A TAPN can be shown graphically as in Fig. 1. In all the figures, circles correspond to places, bars correspond to transitions, and the arrows correspond to arcs. "$w = \ldots$" next to an arc specifies the weight of that arc and "$\tau = \ldots$" specifies the time-delay of that arc. To simplify graphics, however, a unity weight is not explicitly shown. Similarly, a unit time-delay for an output arc and a zero time-delay for an input arc are also not explicitly shown. Finally, the tokens present in a place at the initial time are shown by dots inside the corresponding circle.

Therefore, for the TAPN shown in Fig. 1, we have $\Pi = \{\pi_1, \pi_2, \pi_3\}$, $\Theta = \{\theta_1, \theta_2, \theta_3\}$,

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$S_0 = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T,$$

$\mathcal{A}_N = \{(\pi_1, \theta_1), (\pi_1, \theta_2), (\pi_2, \theta_3), (\pi_3, \theta_3)\}$, $\mathcal{A}_O = \{(\theta_1, \pi_2), (\theta_2, \pi_3), (\theta_3, \pi_1)\}$, $\mathcal{D}_N = \{0, 2, 3, 2\}$, and $\mathcal{D}_O = \{2, 1, 3\}$.

2.2 Arc Stretched Petri Nets

To simplify the representation of the state of a TAPN, the method of arc stretching was first introduced in [15]. In this method, a so-called arc stretched Petri net (ASPN) is defined for any given TAPN. To construct the ASPN, let us first decompose $\mathcal{A}_N$ and $\mathcal{A}_O$ into, respectively, $\mathcal{A}_N^1$ and $\mathcal{A}_N^2$ and $\mathcal{A}_O^1$ and $\mathcal{A}_O^2$, where $\mathcal{A}_N^1 := \{a \in \mathcal{A}_N \mid \mathcal{D}_N(a) = 0\}$, $\mathcal{A}_N^2 := \{a \in \mathcal{A}_N \mid \mathcal{D}_N(a) \geq 1\}$, $\mathcal{A}_O^1 := \{a \in \mathcal{A}_O \mid \mathcal{D}_O(a) = 1\}$, $\mathcal{A}_O^2 := \{a \in \mathcal{A}_O \mid \mathcal{D}_O(a) \geq 2\}$. Then, the ASPN is constructed as follows:

I) for any $a = (\pi, \theta) \in \mathcal{A}_N^2$,
   i) $\mathcal{D}_N(a)$ new places π_1^a, π_2^a, …, $\pi_{\mathcal{D}_N(a)}^a$, and $\mathcal{D}_N(a)$ new transitions, θ_1^a, θ_2^a, …, $\theta_{\mathcal{D}_N(a)}^a$, are defined.
   ii) arc a, which goes from π to θ, now goes from π to θ_1^a; the time-delay of a becomes zero; its weight remains the same.
   iii) $\mathcal{D}_N(a)$ new input arcs, from π_1^a to θ_2^a, …, from $\pi_{\mathcal{D}_N(a)-1}^a$ to $\theta_{\mathcal{D}_N(a)}^a$, and from

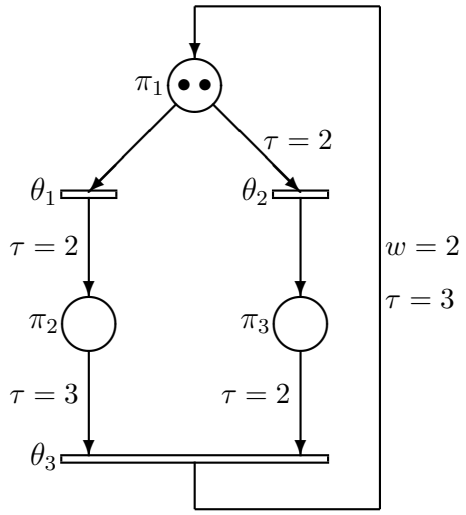

Fig. 1. Example TAPN.

$\pi^a_{\mathcal{D}_N(a)}$ to θ, are defined each having unit weight and zero time-delay.

iv) $\mathcal{D}_N(a)$ new output arcs, from θ^a_1 to π^a_1, ..., from $\theta^a_{\mathcal{D}_N(a)-1}$ to $\pi^a_{\mathcal{D}_N(a)-1}$, and from $\theta^a_{\mathcal{D}_N(a)}$ to $\pi^a_{\mathcal{D}_N(a)}$, are defined each having unit weight and unit time-delay.

II) for any $a = (\theta, \pi) \in \mathcal{A}^2_O$,

i) $\mathcal{D}_O(a) - 1$ new places π^a_1, π^a_2, ..., $\pi^a_{\mathcal{D}_O(a)-1}$, and $\mathcal{D}_O(a)-1$ new transitions, θ^a_1, θ^a_2, ..., $\theta^a_{\mathcal{D}_O(a)-1}$, are defined.

ii) arc a, which goes from θ to π, now goes from $\theta^a_{\mathcal{D}_O(a)-1}$ to π; the time-delay of a becomes unity; its weight remains the same.

iii) $\mathcal{D}_O(a)-1$ new input arcs, from π^a_1 to θ^a_1, from π^a_2 to θ^a_2, ..., and from $\pi^a_{\mathcal{D}_O(a)-1}$ to $\theta^a_{\mathcal{D}_O(a)-1}$, are defined each having unit weight and zero time-delay.

iv) $\mathcal{D}_O(a)-1$ new output arcs, from θ to π^a_1, from θ^a_1 to π^a_2, ..., and from $\theta^a_{\mathcal{D}_O(a)-2}$ to $\pi^a_{\mathcal{D}_O(a)-1}$, are defined each having unit weight and unit time-delay.

Usig the above procedure, for the TAPN shown in Fig. 1, the ASPN shown in Fig. 2 is constructed. In the ASPN, firing of θ^a_1, for any $a = (\pi, \theta) \in \mathcal{A}^2_N$, is equivalent to the firing of θ. Furthermore, θ^a_2, ..., $\theta^a_{\mathcal{D}_N(a)}$, and θ fires immediately as it becomes enabled. Moreover, for any $a = (\theta, \pi) \in \mathcal{A}^2_O$, θ^a_1, θ^a_2, ..., $\theta^a_{\mathcal{D}_O(a)-1}$ also fires immediately as it becomes enabled. We let $\tilde{\Theta}$ denote the set of those transitions which are to fire immediately as they

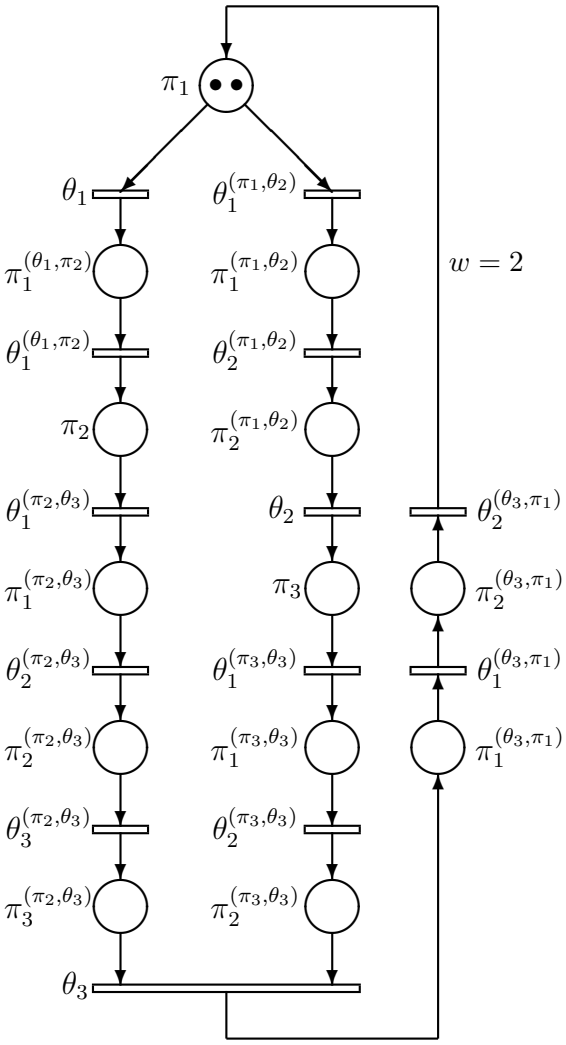

Fig. 2. ASPN for the Example TAPN.

become enabled. The number of tokens initially present in a place in the original Petri net is not changed, but the initial number of tokens in any one of the newly introduced places is defined as zero.

The ASPN is a tuple $\hat{H} = (\hat{\Pi}, \hat{\Theta}, \tilde{\Theta}, \hat{A}, \hat{B}, \hat{S}_0)$, where $\hat{\Pi}$ consist of all the places in Π and the places which are introduced by the above procedure; $\hat{\Theta}$, consist of all the transitions in Θ and the transitions which are introduced by the above procedure; $\tilde{\Theta} \subset \hat{\Theta}$ is the set of transitions which are to fire immediately as they become enabled. The elements of the input matrix, $\hat{A}$,

are given as

$$\hat{A}(\pi,\theta)=\begin{cases} A(\pi,\theta), & \text{if } \pi\in\Pi \\ 1, & \text{if } \pi=\pi_l^a \text{ and } \theta=\theta_{l+1}^a \text{ for some } a\in\mathcal{A}_N^2 \text{ with } \mathcal{D}_N(a)\geq 2 \text{ and } l=1,\ldots,\mathcal{D}_N(a)-1 \\ 1, & \text{if } \pi=\pi_{\mathcal{D}_N(a)}^a \text{ and } \theta=\tilde{\theta} \text{ for some } a=(\tilde{\pi},\tilde{\theta})\in\mathcal{A}_N^2 \\ 1, & \text{if } \pi=\pi_l^a \text{ and } \theta=\theta_l^a \text{ for some } a\in\mathcal{A}_O^2 \text{ and } l=1,\ldots,\mathcal{D}_O(a)-1 \\ 0, & \text{otherwise} \end{cases}$$

and the elements of the output matrix, $\hat{B}$, are given as

$$\hat{B}(\pi,\theta)=\begin{cases} B(\pi,\theta), & \text{if } (\theta,\pi)\in\mathcal{A}_O^1 \\ B(\pi,\tilde{\theta}), & \text{if } \pi\in\Pi \text{ and } \theta=\theta_{\mathcal{D}_O(a)-1}^a \text{ for some } a=(\tilde{\theta},\pi)\in\mathcal{A}_O^2 \\ 1, & \text{if } \pi=\pi_1^a \text{ and } \theta=\tilde{\theta} \text{ for some } a=(\tilde{\theta},\tilde{\pi})\in\mathcal{A}_O^2 \\ 1, & \text{if } \pi=\pi_{l+1}^a \text{ and } \theta=\theta_l^a \text{ for some } a\in\mathcal{A}_O^2 \text{ with } \mathcal{D}_O(a)\geq 3 \text{ and } l=1,\ldots,\mathcal{D}_O(a)-2 \\ 1, & \text{if } \pi=\pi_l^a \text{ and } \theta=\theta_l^a \text{ for some } a\in\mathcal{A}_N^2 \text{ and } l=1,\ldots,\mathcal{D}_N(a) \\ 0, & \text{otherwise} \end{cases}$$

Finally, the initial marking vector, $\hat{S}_0$, is given as

$$\hat{S}_0(\pi)=\begin{cases} S_0(\pi), & \text{if } \pi\in\Pi \\ 0, & \text{otherwise} \end{cases}$$

Note that, since all the input arcs of ASPN have zero time-delay and all the output arcs have unit time-delay, there is no need for any explicit information about the time-delays in $\hat{H}$.

In general, the number of places, transitions, and arcs of the ASPN is more than the number of places, transitions, and arcs of the original TAPN. However, the state of the ASPN at any time $t\in\mathbf{Z}$ is solely and uniquely given by the marking vector $\hat{S}(t)$ at time t. Also, the state of the original TAPN can be defined using the state of the ASPN. Furthermore, starting with $\hat{S}(0)=\hat{S}_0$, the state of the ASPN evolves according to the equation

$$\hat{S}(t+1)=\hat{S}(t)+\sum_{\theta\in f(t)}\left[\hat{B}(\theta)-\hat{A}(\theta)\right] \qquad (1)$$

where $f(t)$ denotes the set of transitions that fire at time t and $\hat{A}(\theta)$ and $\hat{B}(\theta)$ denote the columns of respectively $\hat{A}$ and $\hat{B}$, that correspond to the transition θ. The set of transitions $f(t)$, however, can fire at time t, only if $f(t)\in\mathcal{E}(\hat{S}(t))$, where

$$\mathcal{E}(\sigma):=\left\{f\subset\hat{\Theta}\mid\sigma\geq\sum_{\theta\in f}\hat{A}(\theta)\right\}. \qquad (2)$$

is the set of sets of simultaneously enabled transitions at state σ. Furthermore, the set of enabled transitions at state σ is given as:

$$\epsilon(\sigma):=\left\{\theta\in\hat{\Theta}\mid\sigma\geq\hat{A}(\theta)\right\}. \qquad (3)$$

2.3 Boundedness, Reversibility, and Liveness

The set of all reachable states. $\Sigma(H)$, of a TAPN H from its initial state S_0 is called as its reachability set. As mentioned above, it is rather complicated to describe the state of a TAPN, which makes the description of its reachability set also complicated. However, the state of the ASPN $\hat{H}$, can simply be described by its marking vector. Thus, the reachability set, $\Sigma(\hat{H})$, of the ASPN $\hat{H}$ can easily be described as the set of all the markings $\hat{S}$, such that there exists a sequence of enabled transitions which lead from the initial marking $\hat{S}_0$ to $\hat{S}$. Moreover, there is a one-to-one correspondence between the states of $\hat{H}$ and of H, thus, between $\Sigma(\hat{H})$ and $\Sigma(H)$, [15].

Most desirable properties for a system modelled by a TAPN are boundedness, reversibility, and liveness. For a bound vector $L:\Pi\to\mathbf{Z}$, a TAPN H is said to be L-bounded if $S(\pi)\leq L(\pi)$, $\forall\pi\in\Pi$, for all markings S corresponding to every state in $\Sigma(H)$. H is said to be bounded if it is L-bounded for some bounded L. H is said to be reversible if for every state $S\in\Sigma(H)$, there exists a sequence of enabled transitions which lead from S to the initial state. $\theta\in\Theta$ is said to be live if for every state $S\in\Sigma(H)$, there exists a sequence of enabled transitions which lead from S to $S'\in\Sigma(H)$ such that θ is enabled at S'. For $\Delta\subset\Theta$, H is said to be Δ-live if every $\theta\in\Delta$ is live. Finally, H is said to be live if it is Θ-live.

3 Supervisory Controller Design

A supervisory controller design approach is presented in this section in order to enforce boundedness, reversibility and liveness in a system modelled by a TAPN. Our approach is to first design a

controller for the ASPN, since it is much easier to represent the state of the TAPN using its ASPN. Once a controller which enforces boundedness, reversibility and liveness in the ASPN $\hat{H}$ is obtained, this controller also enforces boundedness, reversibility and liveness in the original TAPN H, since, due to the one-to-one correspondence between the states of the ASPN $\hat{H}$ and of the original TAPN H,

- H is bounded if and only if $\hat{H}$ is bounded.
- H is reversible if and only if $\hat{H}$ is reversible.
- H is live if and only if $\hat{H}$ is live.

In order to design a controller for the ASPN, we first determine its bounded reachability set, Σ_B, for a given bound vector $\hat{L} : \hat{\Pi} \to \mathbf{Z}$. Σ_B is the largest subset of $\Sigma(\hat{H})$ such that any $\hat{S} \in \Sigma_B$ is bounded by $\hat{L}$ (i.e., $\hat{S}(\pi) \leq \hat{L}(\pi)$, $\forall \pi \in \hat{\Pi}$) and can be reached from $\hat{S}_0$ by passing through only states which are bounded by $\hat{L}$. We note that, for any Petri net, Σ_B always has finitely many elements, even though $\Sigma(\hat{H})$ may have infinitely many elements for an unbounded Petri net, [20]. Algorithm 1, which is borrowed from [23], constructs Σ_B. The inputs to this algorithm are the ASPN definition $\hat{H}$ and the bound vector $\hat{L}$. The function $\tilde{\mathcal{E}}(\sigma)$, used in Algorithm 1, returns the set of sets of simultaneously enabled transitions at state σ, excluding those sets which do not include any enabled transition which must fire immediately as it becomes enabled (thus any $\theta \in \tilde{\Theta}$ always fires immediately as it becomes enabled). Thus:

$$\tilde{\mathcal{E}}(\sigma) := \mathcal{E}(\sigma) \setminus \Big\{ f \in \mathcal{E}(\sigma) \mid \theta \notin f \text{ for some } \theta \in \epsilon(\sigma) \cap \tilde{\Theta} \Big\} \ , \qquad (4)$$

where $\mathcal{E}(\cdot)$ is given by (2) and $\epsilon(\cdot)$ is given by (3). Furthermore, the function

$$\nu(\sigma, f) := \sigma + \sum_{\theta \in f} \Big[\hat{B}(\theta) - \hat{A}(\theta) \Big] \qquad (5)$$

returns the next state, as given by (1), when f fires at σ. If $\hat{S}_0(\pi) > \hat{L}(\pi)$, for some $\pi \in \hat{\Pi}$, i.e., if $\hat{H}$ is not $\hat{L}$-bounded at the initial time, Algorithm 1 returns an empty set for Σ_B. In this case, no controller can enforce $\hat{L}$-boundedness.

Following the construction of Σ_B (assuming $\Sigma_B \neq \emptyset$), Algorithm 2, which is also borrowed from [23], constructs the bounded reversible reachability set, Σ_S. Σ_S is the largest subset of Σ_B such that for any $\hat{S} \in \Sigma_S$ either $\hat{S} = \hat{S}_0$ or $\hat{S}$ can be reached from $\hat{S}_0$ and $\hat{S}_0$ can be reached from $\hat{S}$ without passing through any markings outside Σ_S. The inputs to Algorithm 2 are the ASPN definition $\hat{H}$ and the bounded reachability set Σ_B, as constructed by Algorithm 1. Besides Σ_S, this algorithm also returns the set of all live transitions, $\hat{\Delta}$. Here, functions $\tilde{\mathcal{E}}$ and ν are as in Algorithm 1 and the function $\epsilon(\sigma)$, which is given by (3), returns the set of enabled transitions at state σ.

To enforce $\hat{L}$-boundedness and reversibility simultaneously for the ASPN, a controller must keep the state of the ASPN within the bounded reversible reachability set, Σ_S. Such a controller can be described as:

$$\gamma(\sigma, f) := \begin{cases} 1 \ , & \text{if } \nu(\sigma, f) \in \Sigma_S \\ 0 \ , & \text{otherwise} \end{cases} \ , \qquad (6)$$

Algorithm 1 : Algorithm to construct Σ_B

```
Inputs: Ĥ = (Π̂, Θ̂, Θ̃, Â, B̂, Ŝ0) and L̂
Output: ΣB
if Ŝ0 ≤ L̂ then
  ΣB = Σ1 = {Ŝ0}
  loop
    Σ2 = ∅
    for j = 1 to |Σ1| do
      σ = [Σ1]j
      Φ = Ẽ(σ)
      for k = 1 to |Φ| do
        φ = [Φ]k, σ̂ = ν(σ, φ)
        if σ̂ ∉ (Σ2 ∪ ΣB) and σ̂ ≤ L̂ then
          Σ2 ← Σ2 ∪ {σ̂}
        end if
      end for
    end for
    if Σ2 == ∅ then
      exit loop
    end if
    ΣB ← ΣB ∪ Σ2
    Σ1 = Σ2
  end loop
else
  ΣB = ∅
end if
return ΣB
```

Algorithm 2 : Algorithm to determine Σ_S and $\hat{\Delta}$

Inputs: $\hat{H} = (\hat{\Pi}, \hat{\Theta}, \tilde{\Theta}, \hat{A}, \hat{B}, \hat{S}_0)$ and Σ_B
Outputs: Σ_S and $\hat{\Delta}$
$\Sigma_X = \Sigma_B$, $\hat{\Delta} = \emptyset$
Start: $\Sigma' = \Sigma_X \setminus \{\hat{S}_0\}$, $\Sigma_S = \{\hat{S}_0\}$
loop
  $\Sigma = \Sigma'$, $Flg = 0$
  for $j = 1$ to $|\Sigma|$ do
    $\bar{\sigma} = [\Sigma]_j$, $\Phi = \tilde{\mathcal{E}}(\bar{\sigma})$
    for $k = 1$ to $|\Phi|$ do
      $\phi = [\Phi]_k$, $\hat{\sigma} = \nu(\bar{\sigma}, \phi)$
      if $\hat{\sigma} \in \Sigma_S$ then
        $\Sigma_S \leftarrow \Sigma_S \cup \{\bar{\sigma}\}$, $\Sigma' \leftarrow \Sigma' \setminus \{\bar{\sigma}\}$,
        $Flg = 1$, go to Break
      end if
    end for
    Break: continue
  end for
  if $Flg == 0$ then
    exit loop
  end if
end loop
if $\Sigma_S == \{\hat{S}_0\}$ then
  go to Finish
end if
$\Sigma_R = \emptyset$, $\Sigma_a = \{\hat{S}_0\}$
loop
  $\Sigma_b = \emptyset$, $\Sigma_R \leftarrow \Sigma_R \cup \Sigma_a$
  for $j = 1$ to $|\Sigma_a|$ do
    $\bar{\sigma} = [\Sigma_a]_j$, $\Phi = \tilde{\mathcal{E}}(\bar{\sigma})$
    for $k = 1$ to $|\Phi|$ do
      $\phi = [\Phi]_k$, $\hat{\sigma} = \nu(\bar{\sigma}, \phi)$
      if $\hat{\sigma} \in \Sigma_S$ and $\hat{\sigma} \notin \Sigma_R \cup \Sigma_b$ then
        $\Sigma_b \leftarrow \Sigma_b \cup \{\hat{\sigma}\}$
      end if
    end for
  end for
  if $\Sigma_b == \emptyset$ then
    exit loop
  end if
  $\Sigma_a = \Sigma_b$
end loop
if $\Sigma_S \neq \Sigma_R$ then
  $\Sigma_X = \Sigma_R$, go to Start
end if
for $j = 1$ to $|\Sigma_S|$ do
  $\bar{\sigma} = [\Sigma_S]_j$, $\Delta_1 = \epsilon(\bar{\sigma})$
  for $k = 1$ to $|\Delta_1|$ do
    $\theta = [\Delta_1]_k$
    if $\nu(\bar{\sigma}, \{\theta\}) \in \Sigma_S$ then
      $\hat{\Delta} \leftarrow \hat{\Delta} \cup \{\theta\}$
    end if
  end for
end for
Finish: return Σ_S , $\hat{\Delta}$

for any $\sigma \in \Sigma_S$ and for any $f \in \tilde{\mathcal{E}}(\sigma)$. Here, the controller (6) disables the firing of f at state σ if $\gamma(\sigma, f) = 0$ and does not disable the firing of f at state σ if $\gamma(\sigma, f) = 1$. This controller also enforces $\hat{\Delta}$-liveness, where $\hat{\Delta}$ is as returned by Algorithm 2. Thus, if $\hat{\Delta} = \hat{\Theta}$, then controller (6) enforces $\hat{L}$-boundedness, reversibility, and liveness simultaneously. Otherwise, $\hat{\Delta}$ is the largest subset of $\hat{\Theta}$ such that a controller exists which enforces $\hat{L}$-boundedness, reversibility, and $\hat{\Delta}$-liveness simultaneously.

4 Example

We consider an automated manufacturing system example which is a modified version of the example presented in [14]. The system consists of three machines and three stores. Machines 1 and 2 obtain parts put on one of the two transporters at store 1, work on them and send them to stores 2 and 3, respectively. Machine 3 obtain one part from store 2 and one part from store 3, produce the final product and return the transporters to store 1. Machine 1 can receive a part from store 1 with no time-delay. It can deliver its product to store 2 in two time units. Machine 2 can receive a part from store 1 within two time units and can deliver its product to store 3 in one time unit. Machine 3 needs 3 time units to receive a product from store 2 and 2 time units to receive a product from store 3. It can deliver the end product and return both transporters to store 1 in 3 time units.

A TAPN model of this system is shown in Fig. 1, where π_1, π_2, and π_3, represent stores 1, 2, and 3, respectively, and θ_1, θ_2, and θ_3, represent processing by the machines 1, 2, and 3, respectively. The ASPN for this TAPN is shown in Fig. 2.

Our aim is to design a controller to enforce $\hat{L}$-boundedness for $\hat{L}(\pi) = 2$, $\forall \pi \in \hat{\Pi}$, reversibility, and liveness simultaneously. Algorithm 1 returns the bounded reachability set, Σ_B, which consists of 62 states (including the initial state $\hat{S}_0 = \sigma_0$). Algorithm 2 then eliminates 24 of these states, since it is not possible to reach to $\hat{S}_0$ from any one of these states. The remaining 38 states, which form the bounded reversible reachability set Σ_S, are shown in Table 1, where the ordering of the places is as follows: π_1, $\pi_1^{(\theta_1,\pi_2)}$,

Table 1. Bounded reversible reachability set, Σ_S.

$\sigma_0 =$	2	0	0	0	0	0	0	0	0	0	0	0	0
$\sigma_1 =$	1	1	0	0	0	0	0	0	0	0	0	0	0
$\sigma_2 =$	1	0	0	0	0	0	1	0	0	0	0	0	0
$\sigma_3 =$	0	1	0	0	0	0	1	0	0	0	0	0	0
$\sigma_4 =$	1	0	1	0	0	0	0	0	0	0	0	0	0
$\sigma_6 =$	0	0	1	0	0	0	1	0	0	0	0	0	0
$\sigma_7 =$	1	0	0	0	0	0	0	1	0	0	0	0	0
$\sigma_8 =$	0	1	0	0	0	0	0	1	0	0	0	0	0
$\sigma_{10} =$	0	0	1	0	0	0	0	1	0	0	0	0	0
$\sigma_{11} =$	1	0	0	1	0	0	0	0	0	0	0	0	0
$\sigma_{13} =$	0	0	0	1	0	0	1	0	0	0	0	0	0
$\sigma_{16} =$	0	0	0	1	0	0	0	1	0	0	0	0	0
$\sigma_{17} =$	1	0	0	0	0	0	0	0	1	0	0	0	0
$\sigma_{18} =$	0	1	0	0	0	0	0	0	1	0	0	0	0
$\sigma_{20} =$	0	0	1	0	0	0	0	0	1	0	0	0	0
$\sigma_{22} =$	0	0	0	1	0	0	0	0	1	0	0	0	0
$\sigma_{23} =$	1	0	0	0	1	0	0	0	0	0	0	0	0
$\sigma_{25} =$	0	0	0	0	1	0	1	0	0	0	0	0	0
$\sigma_{27} =$	0	0	0	0	1	0	0	1	0	0	0	0	0
$\sigma_{29} =$	0	0	0	0	1	0	0	0	1	0	0	0	0
$\sigma_{30} =$	1	0	0	0	0	0	0	0	0	1	0	0	0
$\sigma_{31} =$	0	1	0	0	0	0	0	0	0	1	0	0	0
$\sigma_{33} =$	0	0	1	0	0	0	0	0	0	1	0	0	0
$\sigma_{35} =$	0	0	0	1	0	0	0	0	0	1	0	0	0
$\sigma_{38} =$	0	0	0	0	1	0	0	0	0	1	0	0	0
$\sigma_{39} =$	1	0	0	0	0	1	0	0	0	0	0	0	0
$\sigma_{41} =$	0	0	0	0	0	1	1	0	0	0	0	0	0
$\sigma_{43} =$	0	0	0	0	0	1	0	1	0	0	0	0	0
$\sigma_{45} =$	0	0	0	0	0	1	0	0	1	0	0	0	0
$\sigma_{47} =$	0	0	0	0	0	1	0	0	0	1	0	0	0
$\sigma_{48} =$	1	0	0	0	0	0	0	0	0	0	1	0	0
$\sigma_{49} =$	0	1	0	0	0	0	0	0	0	0	1	0	0
$\sigma_{51} =$	0	0	1	0	0	0	0	0	0	0	1	0	0
$\sigma_{53} =$	0	0	0	1	0	0	0	0	0	0	1	0	0
$\sigma_{55} =$	0	0	0	0	1	0	0	0	0	0	1	0	0
$\sigma_{57} =$	0	0	0	0	0	1	0	0	0	0	1	0	0
$\sigma_{60} =$	0	0	0	0	0	0	0	0	0	0	0	1	0
$\sigma_{61} =$	0	0	0	0	0	0	0	0	0	0	0	0	1

π_2, $\pi_1^{(\pi_2,\theta_3)}$, $\pi_2^{(\pi_2,\theta_3)}$, $\pi_3^{(\pi_2,\theta_3)}$, $\pi_1^{(\pi_1,\theta_2)}$, $\pi_2^{(\pi_1,\theta_2)}$, π_3, $\pi_1^{(\pi_3,\theta_3)}$, $\pi_2^{(\pi_3,\theta_3)}$, $\pi_1^{(\theta_3,\pi_1)}$, $\pi_2^{(\theta_3,\pi_1)}$. Algorithm 2 also returns $\hat{\Delta} = \hat{\Theta}$, which means that as long as the states are restricted to Σ_S, all the transitions are live.

The controller described by (6) then disables θ_1 at states σ_1, σ_4, σ_{11}, σ_{23}, and σ_{39} and disables $\theta_1^{(\pi_1,\theta_2)}$ at states σ_2, σ_7, σ_{17}, σ_{30}, and σ_{48}, since firing θ_1 at states σ_1, σ_4, σ_{11}, σ_{23}, or σ_{39} or firing $\theta_1^{(\pi_1,\theta_2)}$ at states σ_2, σ_7, σ_{17}, σ_{30}, or σ_{48} leads to a state not in Σ_S. Since firing θ_1 in the ASPN is equivalent to firing θ_1 in the original TAPN and firing $\theta_1^{(\pi_1,\theta_2)}$ in the ASPN is equivalent to firing θ_2 in the original TAPN, a controller which disables θ_1 at states σ_1, σ_4, σ_{11}, σ_{23}, and σ_{39} and disables θ_2 at states σ_2, σ_7, σ_{17}, σ_{30}, and σ_{48}, enforces boundedness, reversibility and liveness simultaneously in the original TAPN. This controller also avoids deadlock, since deadlock can not occur in a live Petri net, [24].

5 Conclusions

Supervisory controller design to enforce some desired properties, namely boundedness, reversibility, and liveness in DES modelled by TAPNs has been considered. Since it is very complicated to represent the state of a TAPN, the approach of stretching has been used to obtain the ASPN of the given TAPN. Since it is much easier to represent the state of the ASPN and there is a one-to-one corresponce between the states of the two Petri nets, an approach to design a controller first for the ASPN and then obtain a controller for the original TAPN is proposed. Algorithms to design such a controller have been presented. These algorithms terminate in finite time. Algorithm 1's computational complexity is proportional to the size of the bounded reachability set, Σ_B. Algorithm 2's computational complexity is no more than the square of the size of Σ_B. Therefore, the computational complexity of the overall design approach is no more than the square of the size of Σ_B.

For a given bound vector $\hat{L}$, the designed controller, (6), enforces $\hat{L}$-boundedness and reversibility simultaneously, whenever it is possible to design such a controller. Furthermore, the designed controller also enforces Δ-liveness, where Δ is the largest possible such subset of Θ. Thus, boundedness, reversibility, and liveness are enforced simultaneously whenever possible. Furthermore, the designed controller is maximally permissive in the sense that no transitions are disabled unnecessarily and the reachability set of the controlled system is the largest possible set in which boundedness and reversibility can be enforced simultaneously.

We have used the so-called behavioral approach, [24], [25], in the controller design. Although behavioral approach is more tedious than

the so-called structural approach, [26], [27], it can guarantee the design of maximally permissive controllers; whereas the structural approach, in general, produces conservative controllers, [26].

DES are getting more complicated each day, [28], [29]. Some of such systems are so complicated that it may not be possible to design a centralized controller for them. Therefore, as a future study, the present approach can be extended to decentralized supervisory controller design, which would be possible along the lines of [30].

References

[1] M. Zhou and F. DiCesare, Petri Net Synthesis for Discrete Event Control of Manufacturing Systems. Norwell, MA: Kluwer Academic Publishers, 1993.

[2] J. Proth and X. Xie, Petri Nets: A Tool for Design and Management of Manufacturing Systems. West Sussex: John Wiley & Sons, 1996.

[3] F. D. J. Bowden, "A brief survey and synthesis of the roles of time in Petri nets," Mathematical and Computer Modelling, vol. 31, pp. 55–68, 2000.

[4] J. Wang, Timed Petri Nets: Theory and Application. Boston, MA: Kluwer Academic, 1998.

[5] W. M. Zuberek, "Timed Petri nets in modeling and analysis of cluster tools," IEEE Transactions on Robotics and Automation, vol. 17, pp. 562–575, 2001.

[6] L. Popova-Zeugmann, Time and Petri Nets. New York: Springer, 2013.

[7] L. Liang, F. Basile, and Z. Li, "A reduced computation of state space to enforce GMECs and deadlock-freeness on TPN systems," in IFAC PapersOnLine 53-4, pp. 166–172, 2020.

[8] S. Akshay, L. Helouet, and R. Phawade, "Combining free choice and time in Petri nets," Journal of Logical and Algebraic Methods in Programming, vol. 101, 2020.

[9] D. Lefebvre and C. Daoui, "Control design for bounded partially controlled TPNs using timed extended reachability graphs and MDP," IEEE Transactions on Systems, Man, and Cybernetics, vol. 50, no. 6, pp. 2273–2283, 2020.

[10] A. Aybar, et.al., "Deadlock avoidance controller design for timed Petri nets using stretching," IEEE Systems Journal, vol. 2, pp. 178–188, 2008.

[11] G. J. Tsinarakis, N. C. Tsourveloudis, and K. P. Valavanis, "Modeling, analysis, synthesis, and performance evaluation of multioperational production systems with hybrid timed Petri nets," IEEE Trans. on Automation Science and Engineering, vol. 3, no. 1, pp. 29–46, 2006.

[12] M. Vatani and A. Doustmohammadi, "Decomposition of first-order hybrid Petri nets for hierarchical control of manufacturing systems," Journal of Manufacturing Systems, vol. 35, pp. 206–214, 2015.

[13] A. Aybar, et.al., "Supervisory controller design for timed-place Petri nets," Kybernetika, vol. 48, pp. 1114–1135, 2012.

[14] A. İftar, "Supervisory control of manufacturing systems modeled by timed Petri nets," in Proceedings of the 12th IFAC Workshop on Intelligent Manufacturing Systems, (Austin, TX, U.S.A.), pp. 128–132, Dec. 2016.

[15] A. Aybar, et.al., "Supervisory control of discrete-event systems modeled by timed-arc Petri nets," in Proceedings of the European Control Conference, (Saint Petersburg, Russia), pp. 656–661, May 2020.

[16] C. Lakos and L. Petrucci, "Modular state space exploration for timed Petri nets," International Journal on Software Tools for Technology Transfer, vol. 9, pp. 393–411, 2007.

[17] A. Aybar, et.al., "Supervisory controller design for timed Petri nets," in Proceedings of the IEEE International Conference on System of Systems Engineering, (Los Angeles, CA, U.S.A.), pp. 59–64, Apr. 2006.

[18] A. Aybar, et.al., "Representation of the state of timed-place Petri nets using stretching," in Proceedings of the 4th IFAC Workshop on Discrete-Event System Design, (Playa de Gandia, Spain), pp. 79–84, Oct. 2009.

[19] Z. W. Li, M. C. Zhou, and N. Q. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," IEEE Transactions on Systems, Man, and Cybernetics–Part C, vol. 38, pp. 173–188, 2008.

[20] A. Aybar, et.al., "Centralized and decentralized supervisory controller design to enforce boundedness, liveness, and reversibility in Petri nets," International Journal of Control, vol. 78, pp. 537–553, 2005.

[21] A. Aybar, et.al., "Supervisory controller design to enforce some basic properties in timed-transition Petri nets using stretching," Nonlinear Analysis: Hybrid Systems, vol. 6, pp. 712–729, 2012.

[22] A. Aybar, et.al., "Supervisory controller design to enforce basic properties in timed-place Petri nets," in Preprints of the 6th IFAC Conference on Management and Control of Production and Logistics, (Fortaleza, Brazil), pp. 486–492, Sept. 2013.

[23] A. İftar, "Supervisory controller design to enforce boundedness, reversibility and liveness in systems modeled by timed Petri nets," WSEAS Transactions on Systems, vol. 22, pp. 745–751, 2023.

[24] R. S. Sreenivas, "On the existence of supervisory policies that enforce liveness in discrete-event dynamic systems modelled by controlled Petri nets," IEEE Transactions on Automatic Control, vol. 42, pp. 928–945, 1997.

[25] H. Boucheneb, A. Alger, and G. Berthelot, "Towards a simplified building of time Petri nets reachability graph," in Proceedings of the 5th International Workshop on Petri Nets and Performance Models, (Reims, France), pp. 46–55, May 1993.

[26] M. V. Iordache and P. J. Antsaklis, "Design of $\mathcal{T}$-liveness enforcing supervisors in Petri nets," IEEE Transactions on Automatic Control, vol. 48, pp. 1962–1974, 2003.

[27] Z. W. Li and M. C. Zhou, "Elemetary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," IEEE Transactions

on Systems, Man, and Cybernetics–Part A, vol. 34, pp. 38–51, 2004.

[28] E. J. Davison, A. G. Aghdam, and D. E. Miller, Decentralized Control of Large-Scale Systems. New York: Springer, 2020.

[29] M. Kordestani, A. A. Safavi, and M. Saif, "Recent survey of large-scale systems: Architectures, controller strategies, and industrial applications," IEEE Systems Journal, vol. 15, pp. 5440–5453, Dec. 2021.

[30] A. Aybar, et.al., "Overlapping decompositions and expansions of Petri nets," IEEE Transactions on Automatic Control, vol. 47, pp. 511–515, 2002.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding

This work has been supported by the Scientific Research Projects Commission of Eskişehir Technical University.

Conflicts of Interest

The author has no conflicts of interest to declare that are relevant to the content of this article.