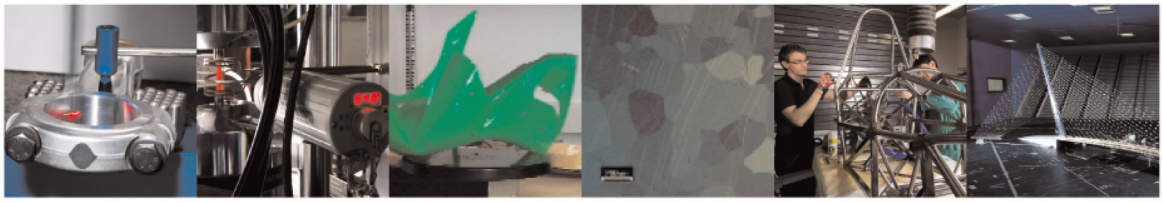

Synthesis of free-labeled Petri nets with inhibitor arcs: an exact and efficient approach

Cheng, Wei;Zhu, Lulai;Matta, Andrea

This is a post-peer-review, pre-copyedit version of W. Cheng, L. Zhu and A. Matta, "Synthesis of Free-Labeled Petri Nets With Inhibitor Arcs: An Exact and Efficient Approach," in IEEE Transactions on Automation Science and Engineering, vol. 23, pp. 10730-10749, 2026, doi: 10.1109/TASE.2026.3695285

Synthesis of free-labeled Petri nets with inhibitor arcs: an exact and efficient approach

Wei Cheng, *Student Member, IEEE,* Lulai Zhu, *Member, IEEE,* Andrea Matta, *Senior Member, IEEE,*

***Abstract*—This paper investigates the synthesis of free-labeled Petri nets with inhibitor arcs from prefix-closed language samples. Inhibitor arcs are crucial for capturing priority logic, such as blocking behavior, in discrete-event systems, and are therefore essential for faithful Petri net models. We formulate the synthesis problem as an integer linear program that seeks a minimum Petri net that reproduces the language samples. To improve scalability, we exploit the structure of the ILP and develop a decomposition strategy that enables exact solution through branch-and-price. The proposed approach is validated on three case studies: the classical producer–consumer system, a parallel production line, and a lab-scale LEGO manufacturing plant. The results demonstrate improved robustness and efficiency over a state-of-the-art algorithm on larger instances. This provides a practical foundation for supervisor control and digital twin development in discrete-event systems.**

***Note to Practitioners*—This paper is motivated by the practical challenge of building trustworthy digital twins for complex manufacturing and automation systems. A digital twin is an active virtual replica of a physical plant, used to simulate system behaviors, monitor operations, and optimize performance. However, to build digital twins, engineers currently face a dilemma. Manually constructing traditional simulation models is highly time-consuming and error-prone, whereas training data-driven neural networks often yield black-box models that are difficult to interpret and trust. To bridge this gap, we propose an automated tool that directly converts routine event logs into transparent, flowchart-like models (Petri nets). These models explicitly capture how processes flow, share limited resources, and prioritize tasks, including complex interactions like system blocking. Consequently, practitioners can rapidly turn historical data into a working, understandable digital twin. This enables engineers to visually pinpoint the root causes of bottlenecks, analyze deadlocks, and safely evaluate new dispatching policies before physical deployment. One current limitation is its strict data requirement: the tool assumes the log captures every possible sequence of events up to a certain length. This completeness is challenging in practice due to rare behaviors or unrecorded events. Ongoing work focuses on constructing suitable inputs from real-world logs and characterizing exactly how much data is needed to reliably generate these digital twins.**

***Index Terms*—Petri net synthesis, Inhibitor arc, Branch-and-price**

I. Introduction

Petri nets are a modeling formalism particularly useful for system verification [1], fault diagnosis [2], [3], and process management [4], [5]. *Petri net synthesis* is the automated process of constructing a Petri net that precisely reflects given system behavior, such as a state space or a language [6], [7]. It underpins automated process model discovery in process management [4], [5] as well as the generation of *digital twin* models (i.e., executable virtual replicas that mirror physical system dynamics) for *discrete-event systems (DES)*, such as manufacturing systems [8], [9], [10]. Similar research has been carried out separately in the fields of both computer science and automation science under different topics, including process mining [4], [5] and system identification [11], [12]. In spite of substantial progress made so far, numerous gaps still remain open in this research area.

One of the research gaps lies in the lack of general approaches to the synthesis of Petri nets with *inhibitor arcs*, commonly referred to as *inhibitor nets* for short. Inhibitor arcs raise the expressive power of Petri nets to Turing completeness and are indispensable for modeling priority rules in unbounded systems [13]. As for bounded systems such as production lines, inhibitor arcs are not strictly required but offer a straightforward way to represent blocking behavior and other execution logic [14]. They have also been effectively exploited in DES to yield compact supervisor control [15], [16]. Consequently, the ability to *synthesize* inhibitor nets from data is crucial for automatically deriving models that are both expressive and compact. Despite this importance, the synthesis of inhibitor nets has received only limited attention so far.

Another research gap lies in the computational complexity of Petri net synthesis, which is NP-complete in general [17]. To cope with this, some works [18], [19] restrict attention to 1-bounded (safe) nets, for which polynomial-time synthesis algorithms exist. However, such 1-bounded nets are often inadequate to capture detailed system behavior, for instance buffer capacities in production lines, which in turn affects the fidelity of subsequent work-in-process control. Other researchers have formulated the synthesis of general Petri nets as an *integer linear program (ILP)* problem [11], [12] and then solved it with off-the-shelf optimization solvers; while exact, these approaches do not exploit the problem structure and therefore have limited scalability. This motivates the development of exact synthesis techniques that exploit the inherent structure of the ILP to improve the efficiency and scalability of Petri net synthesis.

This paper addresses the above two gaps by developing

Manuscript received December 15, 2025. This paper is an extended version of the paper presented at the IEEE 20th International Conference on Automation Science and Engineering (CASE) held on August 28 – September 1, 2024 in Bari, Italy. The research underlying this paper was supported by the European Union's Horizon Europe research and innovation program under grant agreement No. 101092021 (AUTO-TWIN), and China Scholarship Council (CSC).

W. Cheng, L. Zhu, and A. Matta are with Department of Mechanical Engineering, Politecnico di Milano, Via Privata Giuseppe La Masa, 1, Milan, 20156, Italy (e-mail: wei.cheng@polimi.it, lulai.zhu@polimi.it, andrea.matta@polimi.it).

an exact and efficient approach for synthesizing free-labeled Petri nets with inhibitor arcs from prefix-closed language samples. We formulate the synthesis problem as an ILP that searches for a minimum inhibitor net consistent with the given language sample, and we exploit its structure to derive a decomposition strategy that can be solved exactly via branch-and-price (B&P) with improved computational efficiency. The proposed approach is validated on three case studies: the classical producer–consumer system [20], [21], which is unbounded and highlights the indispensable role of inhibitor arcs in modeling priority; a parallel production line and a lab-scale LEGO manufacturing plant, which together demonstrate the scalability and applicability to complex bounded systems. Beyond synthesis itself, the resulting Petri nets provide a foundation for digital twins of DES, supporting model-based monitoring and control. For instance, a synthesized net can run in parallel with a physical production line, with token flows tracking parts and buffer occupancies to enable early detection of blocking and bottlenecks.

The rest of the paper is organized as follows. Section II reviews related work and positions this study. Section III introduces the Petri net formalism and the synthesis problem under study. Section IV presents the proposed ILP formulation and the B&P algorithm. Section V reports experimental results. Section VI concludes the paper and outlines future research directions.

II. State of the art and motivation

A. Related work

In the computer science community, Ehrenfeucht and Rozenberg [6], [7] introduced the first approach to Petri net synthesis, which transforms elementary transition systems into safe Petri nets based on the *theory of regions (TR)*. Since then, several extensions have been made to this approach. Research in [22], [23], and [24] removed the 1-bounded restriction on places. Another extension takes into account inhibitor arcs in the resulting net [25], [26]. Lorenz *et al*. [27] further addressed the synthesis of k-bounded inhibitor nets. From the input perspective, alternative algorithms have been proposed for synthesizing 1-bounded, k-bounded, and unbounded Petri nets from languages [18], [28], [29]. More recently, Peng and Szczerbicka [30] improved TR-based synthesis by speeding up minimal-region computation, enabling more scalable discovery of k-bounded nets from event logs. Along a different input direction, Bergenthum and Kovář [31] proposed Petri net regions to synthesize nets from a set of labeled Petri nets.

TR-based Petri net synthesis techniques typically assume a complete and noise-free specification of system behavior, which is often unrealistic. To address this limitation, van der Aalst *et al*. [4] proposed the so-called α-miner to derive from event logs workflow nets that represent business processes. A business process is a flow of activities performed to acquire a specific outcome, and a workflow net is a special 1-bounded Petri net developed to capture such an activity flow. This pioneering work laid the foundation for the field of process mining, which has achieved widespread success in process management. A review of recent advances in process mining is provided by Augusto *et al*. [5].

Independently, the automation science community has made significant contributions in parallel. Meda-Campaña and López-Mellado [32] considered transition-marking sequences as input for the synthesis of interpreted Petri nets. Building on this, Estrada-Vargas *et al*. [33], [34] identified safe Petri nets using input/output signal sequences that reflect the behavior of manufacturing systems. A method to discover safe Petri nets from event sequences by computing T-invariants was investigated in [19], and Pomares-Angelino and López-Mellado [35] further extended it to incorporate silent transitions via a repairing approach based on structural patterns. López-Mellado and Álvarez-Pérez [36] enhanced identified cyclic nets by scheduling T-components to reduce surplus behavior. Moreover, Montes-Partida and López-Mellado [37] addressed identification and assessment of timed Petri nets from event sequences.

Many approaches leverage the optimization framework to search for a minimum Petri net. Giua and Seatzu [11] formulated the synthesis of free-labeled Petri nets from prefix-closed languages as an ILP. This formulation was later extended in [38] to identify λ-labeled Petri nets. Additionally, Cabasino *et al*. [39] proposed an ILP formulation for discovering unbounded Petri nets from automata, ensuring that the coverability graph of the resulting net is isomorphic to the given automaton. A novel ILP formulation was later introduced by Dotoli *et al*. [12] to identify λ-labeled Petri nets from observed transition-marking sequences, which, compared to the methods present in [11], [38], does not require a complete language as input. The use of ILP techniques for synthesizing nondeterministic nets from finite automata was explored in [40]. Recently, Cheng *et al*. [41] developed a new ILP for the synthesis of free-labeled Petri nets with inhibitor arcs. It is shown in [42], [43] that timed Petri nets and stochastic Petri nets may also be discovered by incorporating temporal information into ILP. Notably, Denno *et al*. [44] applied genetic programming to construct augmented queueing Petri nets based on event logs and states of smart manufacturing systems.

B. Position of this study

Table I compares our approach with representative studies that are most relevant to our problem setting. We focus on general Petri net synthesis rather than specialized discovery tasks (e.g., workflow net discovery for business processes), and we select works with clearly specified input types (e.g., transition systems, prefix-closed language samples, or event sequences/logs). The comparison emphasizes technical aspects that directly affect our setting, including boundedness, inhibitor arc semantics (when applicable), the presence of self-loops, and whether an optimization formulation and a dedicated algorithm are provided. To motivate these dimensions and to clarify our positioning, we briefly discuss why inhibitor arcs matter in unbounded systems, why self-loops are relevant in bounded settings, and the rationale for utilizing the ILP.

First, in unbounded systems, classical place/transition (P/T) nets cannot condition the firing of transitions on whether an unbounded place has exactly a given number of tokens (in

TABLE I
COMPARISON OF RELATED PETRI NET SYNTHESIS STUDIES.

Reference	Input data type & feature	Synthesized Petri net feature			Proposed approach	
		boundedness	inhibitor arc semantics	self-loops	formulation	specific algorithm
Busi and Pinna, 1997 [25]	transition system	1-bounded	a-posteriori	no	none	TR
Pietkiewicz-Koutny, 1999 [26]	transition system	1-bounded	a-priori	yes	none	TR
Lorenz *et al*., 2007 [27]	prefix-closed language samples	k-bounded	a-priori	yes	none	TR
Peng and Szczerbicka, 2025 [30]	event logs	k-bounded	none	no	none	TR
Bergenthum and Kovář, 2025 [31]	labeled Petri nets	k-bounded	none	no	none	TR
Pomares-Angelino and López-Mellado, 2022 [35]	event sequences	1-bounded	none	no	none	structural pattern repair
López-Mellado and Álvarez-Pérez, 2025 [36]	event sequences	1-bounded	none	no	none	T-components approach
Montes-Partida and López-Mellado, 2025 [37]	event sequences	1-bounded	none	no	none	T-invariants approach
Cabasino *et al*., 2006 [39]	coverability graph	unbounded	none	yes	ILP	none
Cabasino *et al*., 2007 [38]	prefix-closed language samples	k-bounded	none	yes	ILP	none
Dotoli *et al*., 2008 [12]	transition & marking	k-bounded	none	no	ILP	none
Zhu *et al*., 2024 [40]	finite automata	k-bounded	none	yes	ILP	none
Denno *et al*., 2018 [44]	event logs & states	k-bounded	a-posteriori	no	none	genetic programming
This study	prefix-closed language samples	unbounded	a-posteriori	yes	ILP	B&P

particular, whether it is empty). The capability to guard a transition by such a place-is-empty condition is usually referred to as *zero testing*. The introduction of inhibitor arcs allows P/T nets to perform zero testing on unbounded places, thereby raising their expressive power to that of Turing machines [13]. This zero-testing capability also enables compact encodings of priority logic: a lower-priority transition can be disabled whenever a higher-priority transition remains fireable, which is crucial for representing advanced DES behaviors such as blocking [14].

Second, in the bounded case, inhibitor arcs are not strictly required to model priority logic, since they can be eliminated via complementary-place transformations [45]. However, such transformations typically increase structural complexity and introduce self-loops, which hinder linear-algebraic analysis and reduce interpretability [13]. This motivates synthesizing inhibitor arcs directly when they yield simpler and more interpretable nets. Moreover, existing inhibitor net synthesis methods often rely on restrictive boundedness assumptions [25], [26] or adopt *a-priori* semantics [27], which differs from the widely used *a-posteriori* semantics in DES modeling and generalized stochastic Petri nets [46], [47]. In addition, the approach in [44] constructs inhibitor nets with a-posteriori semantics from a restricted library of manufacturing-oriented subnets, limiting generality. For studies that do not consider inhibitor arcs, inhibitor arc semantics is not applicable and is therefore marked as "none" in Table I.

Finally, regarding the synthesis methodology, ILP provides a principled way to synthesize Petri nets by translating the behavioral requirements induced by the input data into constraints. Meanwhile, an objective function can encode a desired notion of minimality, such as penalizing net complexity [11], [38], [12], [43], [42]. However, prior work typically relies on off-the-shelf solvers and does not exploit the formulation structure to improve scalability [39], [38], [12], [40]. These observations position the gap addressed in this paper: an ILP for synthesizing free-labeled inhibitor nets from prefix-closed language samples under a-posteriori semantics, solved exactly via B&P. In our approach, inhibitor arcs are introduced only when they are truly needed: either because the target priority behavior in an unbounded setting cannot be represented without them, or because they yield a simpler structure in bounded settings. Otherwise, the synthesized net naturally reduces to a plain P/T net.

III. PRELIMINARIES

This section presents the Petri net formalism considered in this work and the synthesis problem of interest. For more details about Petri net formalisms, we point the reader to [13], [45], and [47].

A. Petri net formalism

Definition 1. A place/transition (P/T) net is a directed bipartite graph defined by a quadruple $PN = (P, T, \mathbf{I}, \mathbf{O})$. P and T are finite and disjoint sets, where P is the set of places and T is the set of transitions. Let $m = |P|$ and $n = |T|$. $\mathbf{I}, \mathbf{O} \in \mathbb{N}^{m \times n}$ are the input and output matrices, respectively.[1] The element $\mathbf{I}(p,t)$ (resp. $\mathbf{O}(p,t)$) specifies the weight assigned to the input arc from the place p to the transition t (resp. the output arc from the transition t to the place p).

Definition 2. The P/T net augmented with *inhibitor* arcs, known as an inhibitor net, is defined by a pair $IN = (PN, \mathbf{H})$. $\mathbf{H} \in \mathbb{N}^{m \times n}$ is the inhibitor matrix, where the element $\mathbf{H}(p,t)$ specifies the weight assigned to the inhibitor arc from the place p to the transition t.

Any arc with a zero weight, regardless of whether it is an input, output or inhibitor arc, has no effect in the inhibitor net and is considered to be absent. Therefore, the sets of input and output transitions of the place p are given by ${}^\bullet p = \{\, t \in$

[1] $\mathbb{N}$ refers to the set of natural numbers including zero.

$T : \mathbf{O}(p,t) > 0\}$ and $p^\bullet = \{t \in T : \mathbf{I}(p,t) > 0\}$; the sets of input, output, and inhibitor places of the transition t can be written as ${}^\bullet t = \{p \in P : \mathbf{I}(p,t) > 0\}$, $t^\bullet = \{p \in P : \mathbf{O}(p,t) > 0\}$, and ${}^\circ t = \{p \in P : \mathbf{H}(p,t) > 0\}$.

The state of the inhibitor net is represented by a vector $\mathbf{M} : P \to \mathbb{N}^{m\times 1}$ whose entry $\mathbf{M}(p)$ specifies the number of tokens in the place p. $\mathbf{M}$ is termed the marking. The net together with an initial marking $\mathbf{M}_0$ constitutes an inhibitor net system defined by a pair $INS = (IN, \mathbf{M}_0)$.

We denote by p_i and t_j respectively the place and the transition corresponding to the i-th row and the j-th column of the input matrix $\mathbf{I}$ (as well as the output matrix $\mathbf{O}$ and the inhibitor matrix $\mathbf{H}$). t_j is enabled at a marking $\mathbf{M}$, shortened to $\mathbf{M}[t_j\rangle$, if and only if (iff) both of the following conditions are satisfied:

$$\forall p \in {}^\bullet t_j : \ \mathbf{M}(p) \geq \mathbf{I}(p,t_j), \tag{1}$$

$$\forall p \in {}^\circ t_j : \ \mathbf{M}(p) < \mathbf{H}(p,t_j). \tag{2}$$

Provided that $\mathbf{M}[t_j\rangle$, t_j may fire and consequently evolve the inhibitor net from $\mathbf{M}$ to a new marking

$$\mathbf{M}' = \mathbf{M} + \mathbf{C}\,\boldsymbol{e}_j, \tag{3}$$

where $\mathbf{C} = (\mathbf{O} - \mathbf{I}) \in \mathbb{Z}^{m\times n}$ is called the incidence matrix, and $\boldsymbol{e}_j \in \{0,1\}^{n\times 1}$ is the j-th canonical basis of an n-dimensional vector space. This single-step evolution is expressed as $\mathbf{M}[t_j\rangle\mathbf{M}'$ for short.

A firing sequence σ is a finite sequence of transitions $t_{(1)}\cdots t_{(j)}\cdots t_{(k)}$ such that $\mathbf{M}_{(0)}[t_{(1)}\rangle\mathbf{M}_{(1)}\cdots[t_{(j)}\rangle\mathbf{M}_{(j)}\cdots[t_{(k)}\rangle\mathbf{M}_{(k)}$.[2] The start marking $\mathbf{M}_{(0)}$ and the end marking $\mathbf{M}_{(k)}$ satisfy the following relation:

$$\mathbf{M}_{(k)} = \mathbf{M}_{(0)} + \mathbf{C}\,\boldsymbol{\sigma}, \tag{4}$$

where $\boldsymbol{\sigma} \in \mathbb{N}^{n\times 1}$ is the firing count vector of σ. The j-th entry of $\boldsymbol{\sigma}$ adds up how many times the transition t_j occurs in σ. We write $\mathbf{M}_{(0)}[\sigma\rangle\mathbf{M}_{(k)}$ to represent such a multi-step evolution and simply $\mathbf{M}_{(0)}[\sigma\rangle$ to indicate that σ is enabled at $\mathbf{M}_{(0)}$.

The reachability set $R(INS)$ of the inhibitor net system is the set of all markings reachable from the initial marking $\mathbf{M}_0$. The net system is said to be v-bounded if $\mathbf{M}(p) \leq v$ holds for every marking $\mathbf{M} \in R(INS)$ and every place $p \in P$.

A nonzero solution $\boldsymbol{\phi} \in \mathbb{N}^{m\times 1}$ to the linear equation system $\mathbf{C}^\top\boldsymbol{\phi} = \mathbf{0}_{n\times 1}$ is a *P-invariant* of the inhibitor net system. It follows from (3) that for any marking $\mathbf{M} \in R(INS)$, the weighted sum $\mathbf{M}^\top\boldsymbol{\phi}$ of the number of tokens in each place remains constant. A nonzero solution $\boldsymbol{\tau} \in \mathbb{N}^{n\times 1}$ to the linear equation system $\mathbf{C}\boldsymbol{\tau} = \mathbf{0}_{m\times 1}$ is a *T-invariant* of the inhibitor net system. As implied by (4), the firing sequence τ evolves the net system from any marking $\mathbf{M} \in R(INS)$ to $\mathbf{M}$ itself provided that the firing count vector of τ equals $\boldsymbol{\tau}$ and $\mathbf{M}[\tau\rangle$.

A language is a formal way of describing the behavior of a DES. The DES event set E is viewed as an alphabet and $\mathcal{L} \subseteq E^*$ is the set of all words (sequences of events) generated by the system, also called the DES language. If an INS is used to model the DES, the system events are associated with transitions. This leads to the introduction of free-labeled INS, in which each transition in T corresponds to a distinct event in E. In this setting, we can identify events with transitions and work directly with languages over T.

Definition 3. The *language* of a free-labeled INS is the set of all firing sequences enabled at the initial marking $\mathbf{M}_0$:

$$\mathcal{L}(INS) = \{\sigma \in T^* : \mathbf{M}_0[\sigma\rangle\}, \tag{5}$$

where T^* denotes the set of all firing sequences over T, including the empty sequence ϵ. By convention, ϵ is always enabled at $\mathbf{M}_0$, hence $\epsilon \in \mathcal{L}(INS)$.

For two sequences $\sigma, \zeta \in T^*$, we say that ζ is a *prefix* of σ if there exists a sequence $\eta \in T^*$ such that $\sigma = \zeta\eta$. The language $\mathcal{L}(INS)$ is *prefix-closed*: the prefix of any sequence $\sigma \in \mathcal{L}(INS)$ is either another firing sequence or the empty sequence ϵ, and both belong to $\mathcal{L}(INS)$.

The *h-bounded sample* of $\mathcal{L}(INS)$, i.e., its truncation at a finite horizon $h \in \mathbb{N}$, is defined by

$$\mathcal{L}_{\leq h}(INS) = \{\sigma \in \mathcal{L}(INS) : |\sigma| \leq h\}. \tag{6}$$

where $|\sigma|$ is the length of σ. The family $(\mathcal{L}_{\leq h}(INS))_{h\in\mathbb{N}}$ of bounded language samples is monotone in the sense that $\mathcal{L}_{\leq h}(INS) \subseteq \mathcal{L}_{\leq h+1}(INS)$ and convergent in the sense that $\lim_{h\to\infty}\mathcal{L}_{\leq h}(INS) = \mathcal{L}(INS)$. If $\mathcal{L}(INS)$ is prefix-closed, then its h-bounded sample $\mathcal{L}_{\leq h}(INS)$ is prefix-closed as well.

We consider two free-labeled INSs to be *equivalent* iff their languages are exactly the same [13]. Equivalence may be defined alternatively in terms of the reachability set, but such notions of equivalence are beyond the scope of this work.

B. Synthesis problem

Petri net synthesis is the process of identifying a Petri net that precisely reflects the behavior of a DES, which in our case is manifested as a language. Given the language $\mathcal{L}$ of the target system, the resulting INS should ideally produce an identical language $\mathcal{L}(INS) = \mathcal{L}$. However, the language sample acquired from the target system is often a finite subset of $\mathcal{L}$. We therefore aim at the partial equality of $\mathcal{L}(INS)$ and $\mathcal{L}$ up to a finite horizon. To this end, the following assumptions are made.

Assumption 1. The language $\mathcal{L}$ of the DES under study is prefix-closed and formed under a *specific initial condition*. Concretely, every word in $\mathcal{L}$ arises from an independent execution that starts from the same initial state (e.g., an event trace recorded during a single run of a manufacturing system in a fixed initial work-in-process setup).

The above assumption complies with the properties of the considered Petri net language, which is defined as the prefix-closed set of firing sequences with respect to the $\mathbf{M}_0$. It is a common assumption in prior work on Petri net synthesis [11], [38], [27].

Assumption 2. The h-bounded sample $\mathcal{L}_{\leq h} \subseteq \mathcal{L}$ is observable from the target system for any finite horizon $h \in \mathbb{N}$.

[2] $t_{(j)}$ is the j-th transition in the firing sequence σ, which is not necessarily the j-th transition t_j in the set T.

Assumption 2 claims the language observability of the DES to be analyzed. As the horizon h grows, the language sample $\mathcal{L}_{\leq h}$ will eventually converge to the full language $\mathcal{L}$ of the system, which represents the complete system behavior. It is however worth pointing out that the complete behavior of a system can already be discovered from $\mathcal{L}_{\leq h}$ for a sufficiently large h even when $\mathcal{L}$ is infinite, as demonstrated in [11] and [38] as well as our experiments (see Section V). On the other hand, the larger h is, the more computation time is needed to solve for the inhibitor net system that can reproduce $\mathcal{L}_{\leq h}$. Therefore, h is a design parameter that trades off between the computation time to spend and the completeness of the system behavior to capture. To manage this trade-off, we incrementally increase the horizon h and re-solve the synthesis problem. We terminate when the objective value stabilizes, indicating a structurally simpler net is unlikely, which is consistent with empirical practices in prior studies [11], [38].

Assumption 3. The number $m = |P|$ of places in the INS to be synthesized is known.

This assumption is mainly for notation simplicity rather than a fundamental restriction. In practice, existing formulations can treat the place set as a decision variable: given an upper bound on the number of places, one can introduce selection variables and minimize $|P|$ to obtain a minimal net [38]. The proposed method in this work is compatible with such formulations, so Assumption 3 can be relaxed.

Problem 1. Given a prefix-closed, h-bounded language sample $\mathcal{L}_{\leq h} \subseteq \mathcal{L}$ and the number $m = |P|$ of places, find the minimum free-labeled inhibitor net system $INS = (P, T, \mathbf{I}, \mathbf{O}, \mathbf{H}, \mathbf{M}_0)$ such that its language sample $\mathcal{L}_{\leq h}(INS)$ matches the given sample $\mathcal{L}_{\leq h}$ within the horizon h.

IV. Methodology

We now elaborate the proposed methodology for solving Problem 1. An overview of the methodology is provided first, followed by a detailed description of each individual step.

A. Overview

Our goal is to obtain the minimum free-labeled INS whose language sample $\mathcal{L}_{\leq h}(INS)$ is the same as the observed language sample $\mathcal{L}_{\leq h}$ within the horizon h. Here, "minimum" means minimizing the sum of total arc weights and initial tokens for a given number of places. For this purpose, a set of positive examples and a set of negative examples are identified from $\mathcal{L}_{\leq h}$. The positive (resp. negative) examples specify transitions to be enabled (resp. disabled) by INS after the firing sequence from the initial marking. Then, we construct an ILP with the parameters of INS as variables. The objective function of the ILP measures the size of INS. Its constraints impose the enabling and disabling requirements as well as additional structural requirements on INS. Finally, we decompose the ILP into a master problem and multiple subproblems so that the exact solution can be found efficiently through the B&P algorithm.

B. Identification of positive and negative examples

The language sample $\mathcal{L}_{\leq h}$ includes all the paths of the DES from the initial state to those reachable in h steps. Let $\sigma \in \mathcal{L}_{\leq h-1}$ be an admissible sequence whose length $|\sigma| \leq h-1$. If appending a transition $t \in T$ to σ yields another admissible sequence $\sigma t \in \mathcal{L}_{\leq h}$, then t is enabled at the state reached by executing σ from the initial state; otherwise, t is disabled at that state. To understand how the DES evolves, we identify every pair $(\sigma, t) \in \mathcal{L}_{\leq h-1} \times T$ as either a *positive* or a *negative* example depending on whether $\sigma t \in \mathcal{L}_{\leq h}$ or not. The sets of identified positive and negative examples are then given by

$$\mathcal{E}^+ = \{\, (\sigma,\, t) \in \mathcal{L}_{\leq h-1} \times T :\ \sigma t \in \mathcal{L}_{\leq h} \,\} \tag{7}$$

$$\mathcal{E}^- = \{\, (\sigma,\, t) \in \mathcal{L}_{\leq h-1} \times T :\ \sigma t \notin \mathcal{L}_{\leq h} \,\} \tag{8}$$

where

$$\mathcal{L}_{\leq h-1} \ = \ \{\, \sigma \in \mathcal{L}_{\leq h} :\ |\sigma| \leq h-1 \,\}. \tag{9}$$

Example 1. Consider a language sample $\mathcal{L}_{\leq 3} = \{\epsilon,\, t_1,\, t_1t_2,\, t_1t_2t_1\}$ with the alphabet $T = \{t_1, t_2\}$ and the horizon $h = 3$. The language sample $\mathcal{L}_{\leq 2}$ with $h = 2$, the set $\mathcal{E}^+$ of positive examples, and the set $\mathcal{E}^-$ of negative examples are as follows:

$$\mathcal{L}_{\leq 2} = \{\, \epsilon,\, t_1,\, t_1t_2 \,\}$$
$$\mathcal{E}^+ = \{\, (\epsilon,\, t_1),\ (t_1,\, t_2),\ (t_1t_2,\, t_1) \,\}$$
$$\mathcal{E}^- = \{\, (\epsilon,\, t_2),\ (t_1,\, t_1),\ (t_1t_2,\, t_2) \,\}.$$

C. Construction of Integer Linear Program

For convenience, we make an arbitrary enumeration of the positive and negative example sets. Let $\xi_k^+ = \left(\sigma_k^+, t_{j_k^+}\right) \in \mathcal{E}^+$ be the k-th positive example. In the free-labeled INS context, this means that there exists a marking $\mathbf{M}_{\sigma_k^+}$ with $\mathbf{M}_0[\sigma_k^+\rangle\mathbf{M}_{\sigma_k^+}$ such that $\mathbf{M}_{\sigma_k^+}[t_{j_k^+}\rangle$. Conversely, let $\xi_l^- = \left(\sigma_l^-,\, t_{j_l^-}\right) \in \mathcal{E}^-$ be the l-th negative example. For the marking $\mathbf{M}_{\sigma_l^-}$ satisfying $\mathbf{M}_0[\sigma_l^-\rangle\mathbf{M}_{\sigma_l^-}$, transition $t_{j_l^-}$ is disabled.

Given the number m of places, we fix the place set as $P = \{p_1, \ldots, p_m\}$ with $|P| = m$.[3] The transition set $T = \{t_1, \ldots, t_n\}$ is formed of distinct symbols appearing in the language sample $\mathcal{L}_{\leq h}$. The unknown parameters to be determined are the input matrix $\mathbf{I}$, the output matrix $\mathbf{O}$, the inhibitor matrix $\mathbf{H}$, and the initial marking $\mathbf{M}_0$. We construct an ILP to compute these unknowns; the notation used in the formulation is summarized in Table II.

$$\text{(ILP)} \quad \min \quad f(\mathbf{I}, \mathbf{O}, \mathbf{H}, \mathbf{M}_0) \tag{10a}$$
$$\text{s.t.} \quad \mathcal{G}\left(\mathcal{E}^+, \mathcal{E}^-\right) \tag{10b}$$

where

$$f(\mathbf{I}, \mathbf{O}, \mathbf{H}, \mathbf{M}_0) = \mathbf{1}_{1\times m}(\mathbf{I} + \mathbf{O} + \mathbf{H})\,\mathbf{1}_{n\times 1} + \mathbf{1}_{1\times m}\mathbf{M}_0 \tag{11}$$

[3]The indices of places are set arbitrarily. Renumbering places will lead to an equivalent net with the same language, so only the number m of places matters for the synthesis.

TABLE II
NOTATION OF THE INTEGER LINEAR PROGRAM.

Parameters	Description
r	Cardinality of the positive example set $\mathcal{E}^+$.
s	Cardinality of the negative example set $\mathcal{E}^-$.
σ_k^+	Firing sequence of the k-th positive example, where $k \in \{1, \ldots, r\}$.
σ_l^-	Firing sequence of the l-th negative example, where $l \in \{1, \ldots, s\}$.
$t_{j_k^+}$	Transition enabled in the k-th positive example, where $j_k^+ \in \{1, \ldots, n\}$ is its index in T.
$t_{j_l^-}$	Transition disabled in the l-th negative example, where $j_l^- \in \{1, \ldots, n\}$ is its index in T.
ξ_k^+	The k-th positive example in $\mathcal{E}^+$, where $\xi_k^+ = \left(\sigma_k^+, t_{j_k^+}\right)$.
ξ_l^-	The l-th negative example in $\mathcal{E}^-$, where $\xi_l^- = \left(\sigma_l^-, t_{j_l^-}\right)$.
$\mathbf{X}$	A binary matrix, where $\mathbf{X}(p_i, t_j) = 1$ iff p_i is an inhibitor place of t_j, and $\mathbf{X}(p_i, t_j) = 0$ otherwise.
$\mathbf{Y}$	A binary matrix. For the ξ_l^-, $\mathbf{Y}\left(p_i, \xi_l^-\right) = 1$ iff p_i is an input place of $t_{j_l^-}$ and disables $t_{j_l^-}$ at $\mathbf{M}_{\sigma_l^-}$; otherwise $\mathbf{Y}\left(p_i, \xi_l^-\right) = 0$.
$\mathbf{Z}$	A binary matrix. For the ξ_l^-, $\mathbf{Z}\left(p_i, \xi_l^-\right) = 1$ iff p_i is an inhibitor place of $t_{j_l^-}$ and enables $t_{j_l^-}$ at $\mathbf{M}_{\sigma_l^-}$; otherwise $\mathbf{Z}\left(p_i, \xi_l^-\right) = 0$.
K	A sufficiently large positive constant used in the ILP.

$$
\mathcal{G}\left(\mathcal{E}^+, \mathcal{E}^-\right) =
\begin{cases}
\mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_k^+ - \mathbf{I}\boldsymbol{e}_{j_k^+} \geq \mathbf{0}_{m\times 1} \quad \forall\, \xi_k^+ \in \mathcal{E}^+ & \text{(12a)}\\
\mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_k^+ - \mathbf{H}\boldsymbol{e}_{j_k^+} \leq -\mathbf{1}_{m\times 1} \\
\qquad + K\left(\mathbf{1}_{m\times 1} - \mathbf{X}\left(:, t_{j_k^+}\right)\right) \quad \forall\, \xi_k^+ \in \mathcal{E}^+ & \text{(12b)}\\
\mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_k^+ - \mathbf{H}\boldsymbol{e}_{j_k^+} \geq -K\mathbf{X}\left(:, t_{j_k^+}\right) \quad \forall\, \xi_k^+ \in \mathcal{E}^+ & \text{(12c)}\\
\mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_l^- - \mathbf{I}\boldsymbol{e}_{j_l^-} \leq -\mathbf{1}_{m\times 1} \\
\qquad + K\left(\mathbf{1}_{m\times 1} - \mathbf{Y}\left(:, \xi_l^-\right)\right) \quad \forall\, \xi_l^- \in \mathcal{E}^- & \text{(12d)}\\
\mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_l^- - \mathbf{I}\boldsymbol{e}_{j_l^-} \geq -K\mathbf{Y}\left(:, \xi_l^-\right) \quad \forall\, \xi_l^- \in \mathcal{E}^- & \text{(12e)}\\
\mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_l^- - \mathbf{H}\boldsymbol{e}_{j_l^-} \geq -K\mathbf{Z}\left(:, \xi_l^-\right) \quad \forall\, \xi_l^- \in \mathcal{E}^- & \text{(12f)}\\
\mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_l^- - \mathbf{H}\boldsymbol{e}_{j_l^-} \leq -\mathbf{1}_{m\times 1} \\
\qquad + K\left(\mathbf{1}_{m\times 1} - \mathbf{Z}\left(:, \xi_l^-\right)\right) \quad \forall\, \xi_l^- \in \mathcal{E}^- & \text{(12g)}\\
\mathbf{1}_{1\times m}\left(\mathbf{X}\left(:, t_{j_l^-}\right) - \mathbf{Z}\left(:, \xi_l^-\right)\right) + \mathbf{1}_{1\times m}\mathbf{Y}\left(:, \xi_l^-\right) \\
\qquad \geq 1 \quad \forall\, \xi_l^- \in \mathcal{E}^- & \text{(12h)}\\
\mathbf{M}_0 \in \mathbb{N}^{m\times 1} & \text{(12i)}\\
\mathbf{I}, \mathbf{O}, \mathbf{H} \in \mathbb{N}^{m\times n} & \text{(12j)}\\
\mathbf{C} = \mathbf{O} - \mathbf{I} \in \mathbb{Z}^{m\times n} & \text{(12k)}\\
\mathbf{X} \in \{0,1\}^{m\times n} & \text{(12l)}\\
\mathbf{Y}, \mathbf{Z} \in \{0,1\}^{m\times s}. & \text{(12m)}
\end{cases}
$$

The size of the ILP (10) is reported in Table III for reference. In this ILP, the enabling and disabling implications induced by the positive and negative examples are encoded in the linear constraint set $\mathcal{G}\left(\mathcal{E}^+, \mathcal{E}^-\right)$ in (12). For a positive example, constraints (12a) enforce the input place enabling condition (see (29)), while (12b)–(12c) with $\mathbf{X}$ encode the inhibitor place enabling condition (see (30)). For a negative example, constraints (12d)–(12e) with $\mathbf{Y}$ encode the input place disabling condition (see (31)), and (12f)–(12g) with $\mathbf{Z}$ encode the inhibitor-place disabling condition (see (32)). Constraint (12h) then enforces that, for each negative example, at least one input or inhibitor place disables the transition, implementing the disjunction in (31)–(32). Any feasible solution of $\mathcal{G}\left(\mathcal{E}^+, \mathcal{E}^-\right)$ therefore yields a parameter set $(\mathbf{I}, \mathbf{O}, \mathbf{H}, \mathbf{M}_0)$ that satisfies all example-induced implications, as shown in Theorem 1. The objective $f(\mathbf{I}, \mathbf{O}, \mathbf{H}, \mathbf{M}_0)$ in (11) then selects one with minimum total arc weight and minimum initial marking.

TABLE III
SIZE OF THE INTEGER LINEAR PROGRAM.

Description	Size
number of integer variables (non-binary)	$m(3n+1)$
number of binary variables	$m(n+2s)$
number of constraints	$3mr + 4ms + s$

Theorem 1. *Let $\mathcal{E}^+$ and $\mathcal{E}^-$ be the set of positive examples and the set of negative examples identified from a prefix-closed, h-bounded language sample $\mathcal{L}_{\leq h}$, respectively. If a free-labeled $INS = (P, T, \mathbf{I}, \mathbf{O}, \mathbf{H}, \mathbf{M}_0)$ satisfies $\mathcal{G}(\mathcal{E}^+, \mathcal{E}^-)$, then its language sample $\mathcal{L}_{\leq h}(INS)$ matches $\mathcal{L}_{\leq h}$ within the horizon h.*

Proof: See Appendix A. ∎

It is also possible to augment the ILP with supplementary constraints that impose structural requirements on the resultant INS, as in previous work [11], [38], [12]. For instance, to avoid the presence of isolated places or transitions, one may specify the constraints (13a)–(13d). More precisely, (13a) and (13b) ensure that every place has at least one input and one output transition, while (13c) and (13d) ensure that each transition has at least one input and one output place. If a P-invariant $\boldsymbol{\phi}$ is prescribed (e.g., due to a conservation requirement), the constraints (13e) and (13f) come in handy. Likewise, the constraints (13g) and (13h) define a T-invariant $\boldsymbol{\tau}$.

$$
\begin{cases}
\mathbf{O}\,\mathbf{1}_{n\times 1} \geq \mathbf{1}_{m\times 1} & \text{(13a)}\\
\mathbf{I}\,\mathbf{1}_{n\times 1} \geq \mathbf{1}_{m\times 1} & \text{(13b)}\\
\mathbf{I}^\top \mathbf{1}_{m\times 1} \geq \mathbf{1}_{n\times 1} & \text{(13c)}\\
\mathbf{O}^\top \mathbf{1}_{m\times 1} \geq \mathbf{1}_{n\times 1} & \text{(13d)}\\
\mathbf{C}^\top \boldsymbol{\phi} = \mathbf{0}_{n\times 1} & \text{(13e)}\\
\boldsymbol{\phi} \in \mathbb{N}^{m\times 1} \setminus \{\mathbf{0}_{m\times 1}\} & \text{(13f)}\\
\mathbf{C}\,\boldsymbol{\tau} = \mathbf{0}_{m\times 1} & \text{(13g)}\\
\boldsymbol{\tau} \in \mathbb{N}^{n\times 1} \setminus \{\mathbf{0}_{n\times 1}\}. & \text{(13h)}
\end{cases}
$$

The constant K in the formulation (12) plays the role of a Big-M coefficient and must dominate all the markings visited along any sequence $\sigma \in \mathcal{L}_{\leq h}$. Thus, K should be large enough such that

$$
K \gg \max\left\{\begin{matrix} \max_i \mathbf{M}_0(p_i) + \max|\sigma| \cdot \max_{i,j} \mathbf{O}(p_i, t_j), \\ \max_i \mathbf{M}_\sigma(p_i) \end{matrix}\right\}, \tag{14}
$$

where $\mathbf{M}_\sigma = \mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}$ is the marking obtained by firing the sequence $\sigma \in \mathcal{L}_{\leq h}$ from the initial marking $\mathbf{M}_0$. Since $\mathbf{O}$, $\mathbf{C}$, and $\mathbf{M}_0$ are unknown, (14) cannot be used to determine K. In practice, K has to be chosen based on prior knowledge or conservative worst-case estimation. However, an excessively large choice of K would weaken the linear programming (LP) relaxation [48] and thus degrades the performance of the default *branch-and-cut (B&C)* algorithm used by modern optimization solvers (e.g., Gurobi [49]) to solve the ILP.

D. Solution of Integer Linear Program

To mitigate the impact of these Big-M coefficients, we analyze the structure of the ILP and observe that its feasible region exhibits a *single-bordered block-diagonal* form. This observation motivates a *Dantzig–Wolfe (DW)* decomposition of the ILP into a *master problem (MP)* and multiple *subproblems (SPs)* which can be solved exactly via the B&P algorithm. Constraints involving the Big-M coefficient K are allocated to the SPs, whereas the MP operates on the convex hulls of the feasible regions of the SPs. The DW decomposition yields a Lagrangian bound, which is remarkably tighter than the classical bound obtained through the direct relaxation of the original ILP [48]. Furthermore, the MP and SPs arising from the DW decomposition have a considerably lower dimension than the original ILP and are therefore much easier to solve. In the remainder of this subsection, we derive the DW decomposition as well as the resultant MP and SPs.

Dantzig–Wolfe decomposition: As shown in Appendix B, the original ILP (10) can be rewritten in the standard form (SF)

$$\begin{aligned} (\text{SF}) \quad \min \quad & \boldsymbol{\gamma}\boldsymbol{\alpha} && (15a)\\ \text{s.t.} \quad & \mathbf{A}\,\boldsymbol{\alpha} \leq \boldsymbol{\beta} && (15b)\\ & \boldsymbol{\alpha} \in \mathbb{N}, && (15c) \end{aligned}$$

where $\boldsymbol{\alpha}$ is the decision variable vector, $\boldsymbol{\gamma}$ is the objective coefficient vector, $\mathbf{A}$ is the constraint coefficient matrix, and $\boldsymbol{\beta}$ is the right-hand-side vector. From (40), we know that the constraint set (15b) of the SF has the following single-bordered block-diagonal structure:

$$\begin{cases} \mathbf{B}_1\boldsymbol{\alpha}_1 + \cdots + \mathbf{B}_i\boldsymbol{\alpha}_i + \cdots + \mathbf{B}_m\boldsymbol{\alpha}_m \leq \boldsymbol{\beta}_0 & (16a)\\ \mathbf{D}_1\boldsymbol{\alpha}_1 \leq \boldsymbol{\beta}_1 & (16b)\\ \qquad \ddots \qquad\quad \vdots & \\ \qquad\quad \mathbf{D}_i\boldsymbol{\alpha}_i \leq \boldsymbol{\beta}_i & (16c)\\ \qquad\qquad \ddots \quad \vdots & \\ \qquad\qquad\quad \mathbf{D}_m\boldsymbol{\alpha}_m \leq \boldsymbol{\beta}_m. & (16d) \end{cases}$$

The first constraint (16a) links all the block variables $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$, while each of the rest (16b)–(16d) entails only to a single block variable $\boldsymbol{\alpha}_i$. By its definition (37) , $\boldsymbol{\alpha}_i$ collects all the decision variables related to the place p_i, i.e., the entries of the i-th rows of the input matrix $\mathbf{I}$, the output matrix $\mathbf{O}$, and the inhibitor matrix $\mathbf{H}$ together with the i-th entry of the initial marking $\mathbf{M}_0$. The feasible region of the SF can be obtained from (16) as

$$\mathcal{S} = \left\{ (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m) \;\middle|\; \sum_{i=1}^{m} \mathbf{B}_i\boldsymbol{\alpha}_i \leq \boldsymbol{\beta}_0 \text{ and } \boldsymbol{\alpha}_i \in \mathcal{S}_i \right\}, \tag{17}$$

where $\mathcal{S}_i = \{\boldsymbol{\alpha}_i \in \mathbb{N} : \ \mathbf{D}_i\boldsymbol{\alpha}_i \leq \boldsymbol{\beta}_i\}$ is the block-i feasible region.

We decompose the SF by delegating the linking constraint (16a) to the MP and distributing the remaining constraints (16b)–(16d) to m SPs, each of which involves a single block variable $\boldsymbol{\alpha}_i$. Thus, solving the i-th subproblem SP_i yields a feasible $\boldsymbol{\alpha}_i$ that depicts a *subnet* centered around the place p_i, including transitions connected with p_i, the weights of the connecting arcs, and the initial marking of p_i. The MP helps to combine m such subnets into a complete inhibitor net system, enforcing the linking constraint (16a) and minimizing the global objective (15a).

Master problem: The MP is defined by

$$\begin{aligned} (\text{MP}) \quad \min \quad & \sum_{i=1}^{m} \boldsymbol{\gamma}_i\boldsymbol{\alpha}_i && (18a)\\ \text{s.t.} \quad & \sum_{i=1}^{m} \mathbf{B}_i\boldsymbol{\alpha}_i \leq \boldsymbol{\beta}_0 && (18b)\\ & \boldsymbol{\alpha}_i \in \mathcal{S}_i \quad \forall i \in \{1, \ldots, m\}. && (18c) \end{aligned}$$

It follows from (18c) that the search space $\prod_{i=1}^{m} \mathcal{S}_i$ of the MP contains exponentially many integer points. Instead of solving the MP directly, we apply the *column generation (CG)* technique [48], which finds the solution to the MP over the convex hulls of the SPs' feasible regions by iteratively solving a restricted master problem (RMP). In the context of CG, a column refers to the vector of coefficients associated with a decision variable. The CG loop starts from a few columns and adds a new column in each iteration until the solution to the RMP converges to the Lagrangian bound of the MP.

By Minkowski's theorem (Theorem 4.8 in [50]), the convex hull of $\mathcal{S}_i$ (i.e., the feasible region of the i-th subproblem SP_i) can be represented as

$$\begin{aligned} \text{conv}\,(\mathcal{S}_i) = \Bigg\{ \boldsymbol{\alpha}_i \in \mathbb{R} : \boldsymbol{\alpha}_i = & \sum_{q=1}^{Q} \lambda_{i,q}\boldsymbol{\nu}_{i,q} + \sum_{w=1}^{W} \mu_{i,w}\boldsymbol{\rho}_{i,w}, \\ & \sum_{q=1}^{Q} \lambda_{i,q} = 1, \lambda_{i,q} \in \mathbb{R}_+, \mu_{i,w} \in \mathbb{R}_+ \Bigg\}, \end{aligned} \tag{19}$$

where $\lambda_{i,q}$ and $\mu_{i,w}$ are non-negative coefficients, while $\boldsymbol{\nu}_{i,q}$ and $\boldsymbol{\rho}_{i,w}$ are the extreme points and rays of $\text{conv}(S_i)$. Applying (19) to replace $\boldsymbol{\alpha}_i$ in the MP (18) yields the RMP (20). The decision variables of the RMP are the non-negative coefficients $\lambda_{i,q}$ and $\mu_{i,w}$. The columns of the RMP are composed of the objective coefficients $\boldsymbol{\gamma}_i$ and the constraint coefficients $\mathbf{B}_i$ of the MP as well as the extreme points $\boldsymbol{\nu}_{i,q}$ and rays $\boldsymbol{\rho}_{i,w}$ of $\text{conv}(S_i)$. Here, π is the dual variable of the linking constraint (20b), and π'_i is the dual variable of the convexity constraint (20c) on $\text{conv}(S_i)$. They are used to compute the local objectives of the SPs and guide the dynamic addition of columns in the CG loop.

$$\text{(RMP)} \quad \min \quad \sum_{i=1}^{m}\sum_{q=1}^{Q}\left(\boldsymbol{\gamma}_i\boldsymbol{\nu}_{i,q}\right)\lambda_{i,q} + \sum_{i=1}^{m}\sum_{w=1}^{W}\left(\boldsymbol{\gamma}_i\boldsymbol{\rho}_{i,w}\right)\mu_{i,w} \tag{20a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{q=1}^{Q}\left(\mathbf{B}_i\boldsymbol{\nu}_{i,q}\right)\lambda_{i,q} + \sum_{i=1}^{m}\sum_{w=1}^{W}\left(\mathbf{B}_i\boldsymbol{\rho}_{i,w}\right)\mu_{i,w} \le \boldsymbol{\beta}_0 \quad [\pi] \tag{20b}$$

$$\sum_{q=1}^{Q}\lambda_{i,q} = 1 \qquad [\pi_i'] \quad \forall i \in \{1,\dots,m\} \tag{20c}$$

$$\lambda_{i,q} \in \mathbb{R}_+, \mu_{i,w} \in \mathbb{R}_+ \quad \forall i \in \{1,\dots,m\}\,. \tag{20d}$$

Subproblems: The i-th subproblem SP_i is defined by

$$(\text{SP}_i) \quad \min \quad \theta_i\left(\boldsymbol{\alpha}_i\right) \tag{21a}$$

$$\text{s.t.} \quad \mathbf{D}_i\boldsymbol{\alpha}_i \le \boldsymbol{\beta}_i \tag{21b}$$

$$\boldsymbol{\alpha}_i \in \mathbb{N}, \tag{21c}$$

where $\theta_i\left(\boldsymbol{\alpha}_i\right) = \left(\boldsymbol{\gamma}_i - \pi\mathbf{B}_i\right)\boldsymbol{\alpha}_i - \pi_i'$ is the reduced cost of the block variable $\boldsymbol{\alpha}_i$, given the current RMP duals π and π_i'. Although SP_i aims to find the optimal $\boldsymbol{\alpha}_i$ that minimizes $\theta_i\left(\boldsymbol{\alpha}_i\right)$, it only needs to be solved exactly in the final CG iteration to certify the optimality of the RMP. In intermediate CG iterations, a feasible $\boldsymbol{\alpha}_i$ with a negative $\theta_i\left(\boldsymbol{\alpha}_i\right)$ is sufficient. If such a $\boldsymbol{\alpha}_i$ exists, a new column is constructed from that $\boldsymbol{\alpha}_i$ and added to the RMP in the next CG iteration. Otherwise if no SPs can contribute a negative reduced cost, the current solution to the RMP is optimal. Branching is then applied to enforce the integrality of the found solution.

E. Computational complexity

We discuss the complexity of the proposed ILP in terms of the numbers of variables and constraints. Given a language sample $\mathcal{L}_{\le h}$ up to horizon h over the alphabet T, where $|T| = n$, the ILP is constructed from the identified positive and negative example pairs $\mathcal{E}^+, \mathcal{E}^- \subseteq \mathcal{L}_{\le h-1} \times T$, with $r = |\mathcal{E}^+|$ and $s = |\mathcal{E}^-|$. Since $\mathcal{E}^+$ and $\mathcal{E}^-$ partition $\mathcal{L}_{\le h-1} \times T$, we have

$$r + s = |\mathcal{L}_{\le h-1}| \cdot |T|. \tag{22}$$

In the worst case, the observed sample is the full language universe up to horizon h, i.e., $\mathcal{L}_{\le h} = T^{\le h}$. Then

$$r + s = |T^{\le h-1}| \cdot n = \sum_{\ell=1}^{h} n^\ell = \frac{n^{h+1} - n}{n-1}, \qquad (n \ge 2). \tag{23}$$

This shows that the explicit input size (the number of identified examples) grows exponentially with h. For completeness, when $n = 1$, we have $r + s = h$.

As summarized in Table III, the ILP in Sec. IV-C contains $m(3n+1)$ general integer variables and $m(n+2s)$ binary variables, with $3mr + 4ms + s$ linear constraints. Under the above worst-case input, the total numbers of variables and constraints is therefore $\mathcal{O}(mn^h)$. Hence, assembling the ILP is linear in the explicit ILP size, while solving the resulting ILP exactly is NP-hard in general [48].

V. Numerical experiments and results

This section evaluates the proposed approach for inhibitor net synthesis on three case studies: the classical producer–consumer system, a parallel production line, and a laboratory LEGO plant. We first introduce the three case studies, then describe the experimental design, and finally discuss the results.

A. Case studies

Producer–consumer system: In the first case study, we use the producer–consumer system to demonstrate the applicability of our approach to unbounded systems. As shown in Fig. 1, this system, originally presented in [20], [21] to demonstrate the limitation of P/T nets in conducting zero testing, comprises two producers (P1, P2), two consumers (C1, C2), two buffers (B1, B2), and a shared channel. Producer P1 supplies only consumer C1, and producer P2 supplies only consumer C2. Unconsumed items are stored in dedicated buffers: B1 for (P1, C1) pairs and B2 for (P2, C2) pairs. The shared channel conveys items from the buffers to the consumers according to a priority rule. (P1, C1) pairs have precedence over (P2, C2) pairs in using the channel. Specifically, no items can be transmitted from buffer B2 to consumer C2 unless buffer B1 is empty. When the capacity of buffer B1 is unlimited, the place representing the state of buffer B1 is unbounded. Because classical P/T nets do not support zero testing on an unbounded place, the aforementioned priority rule cannot be modeled within a P/T net [13].

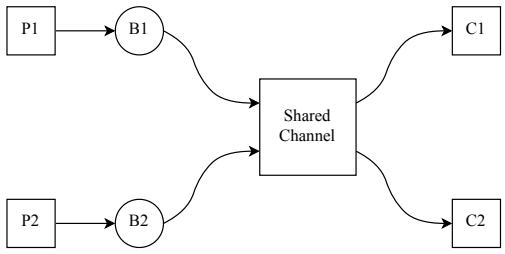

Fig. 1. Case study 1: the producer–consumer system.

Fig. 2 shows the inhibitor-net model of the producer–consumer system. In this model, places p_1, p_3, and p_5 encode the states of producer P1, consumer C1, and buffer B1, while p_2, p_4, and p_6 encode the states of producer P2, consumer C2, and buffer B2. Transitions t_1 and t_2 represent the production of

an item by producers P1 and P2, respectively, and transitions t_3 and t_4 represent the consumption of an item by consumer C1 and C2, respectively. The inhibitor arc from place p_5 to transition t_4 captures the priority rule by disabling t_4 whenever p_5 contains at least one token.

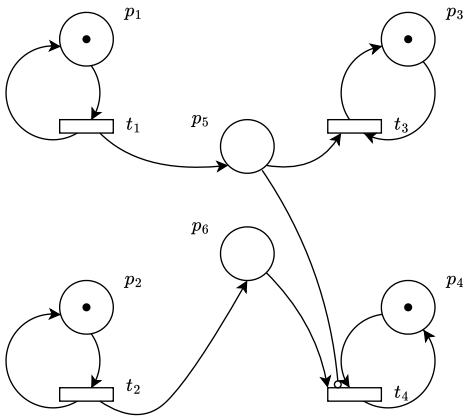

Fig. 2. Inhibitor net model of the producer–consumer system.

Parallel production line: In the second case study, we use a parallel production line to demonstrate the scalability and applicability of our approach on bounded systems with complex control logic. As shown in Fig. 3, the line consists of four stations (S1–S4) for discrete-part manufacturing. Operating in parallel, stations S2 and S3 both feature a finite two-slot buffer and assemble two raw parts from the source station S1 into one finished part, which is then unloaded by the sink station S4. Part flows in this system are governed by highly concurrent and interlocking control policies, including priority-based routing (PBR, where S2 has higher priority than S3), blocking after service (BAS, where parts are held at their current station if downstream buffers are full), and random-order unblocking (ROU, where blocked parallel stations compete randomly for downstream availability).

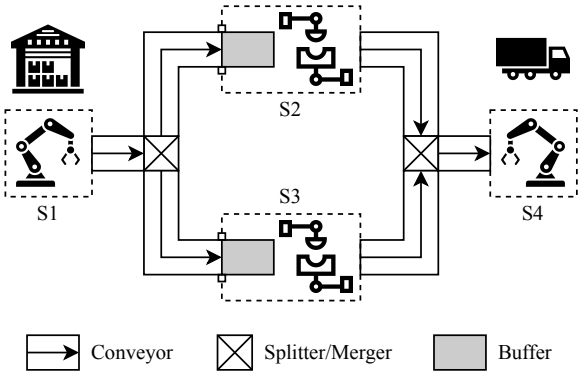

Fig. 3. Case study 2: a parallel production line.

To further assess the practical relevance of the synthesis algorithm, we consider two distinct configurations for this case study. The first is a *deadlock-free* configuration under normal operation. Since deadlocks are a major practical concern in manufacturing systems, the second is a more challenging *deadlock-prone* configuration, obtained through a realistic fault-injection scenario. Specifically, we simulate a sudden communication loss or catastrophic breakdown at the sink station S4 by adding a fault transition. Under the BAS policy, this single-point failure traps the system and creates a deadlock across the entire production line. The detailed structures of the ground-truth inhibitor nets and their equivalent P/T nets are provided in Appendix C.

Lab-scale LEGO manufacturing plant: In the third case study, we use a lab-scale LEGO manufacturing plant to demonstrate the capability of the proposed algorithm to synthesize a complex inhibitor-net model from a prefix-closed language of a physical system. The LEGO plant is a closed-loop manufacturing system built with LEGO bricks and controllers [51]. As illustrated in Fig. 4, the system comprises five stations: three (S1, S2, and S5) are arranged in series, while two (S3 and S4) operate in parallel to provide flexibility and balance the workload. Each station is equipped with a preceding buffer, which temporarily holds parts to reduce production delays, and a processing machine. A conveyor system connects all stations and enables the continuous flow of parts throughout the plant. As in Case Study 2, part flows in the LEGO plant are governed by the BAS policy. With respect to routing, however, the LEGO plant differs from the production-line case: instead of PBR, it employs purely nondeterministic routing on its parallel branches. After parts are processed at station S2, a gate (G) dispatches them to either S3 or S4 according to a random competition policy. Likewise, when parts from S3 and S4 merge into S5, the system follows the ROU policy. The detailed structure of the ground-truth Petri net for this system is provided in Appendix C.

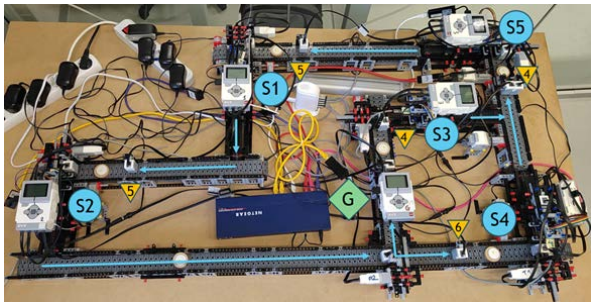

Fig. 4. Case study 3: layout of the lab-scale LEGO manufacturing plant.

B. Experimental design

We designed two experiments to evaluate the proposed synthesis approach from complementary perspectives. The first experiment assesses the effectiveness of the proposed ILP formulation in modeling the synthesis problem, while the second evaluates the efficiency of the proposed exact solution for solving that formulation.

Effectiveness experiment: This experiment evaluates the capability of the proposed ILP to construct an inhibitor-net model for unbounded systems with zero testing. As a baseline, we use the ILP formulation introduced in [11], which also takes a language sample as input but synthesizes a plain P/T net. To ensure a fair comparison, both ILP formulations are

applied to the first case study, the producer–consumer system, using the same input language sample and solved with Gurobi. Effectiveness is assessed in terms of the compactness of the synthesized net, its structural isomorphism to the ground-truth inhibitor net, and its language consistency with the input sample up to the specified horizon h.

As noted in Section II, alternative approaches for inhibitor-net synthesis either rely on restrictive assumptions (e.g., 1-boundedness [25], [26] or *a priori* concurrency [27]) or use template-based constructions tailored to specific classes of systems [44]. Since these approaches are not generally applicable to our setting, we do not consider them comparable baselines.

Efficiency experiment: This experiment evaluates the efficiency of the proposed decomposition-based approach for solving the proposed ILP exactly. Our goal is to compare the proposed problem-specific DW reformulation with a strong direct-solution baseline for the proposed ILP. To this end, we use the default B&C algorithm in Gurobi as the baseline. The proposed ILP is applied to all three case studies and solved using both approaches. For a fair comparison, we align key algorithmic settings, such as the thread count and time limit, across all experiments. The main performance indicator is the CPU time required to solve each instance exactly.

Experimental setup: Table IV summarizes the configurations of the numerical experiments, including the language-sample horizons (h), the numbers of positive and negative examples (*pos* and *neg*), the large constant K, and the number of places m for all instances.

For the first two case studies, the prefix-closed language sample $\mathcal{L}_{\leq h}$ is generated from the ground-truth Petri nets shown in Fig. 2, Fig. 7(a), and Fig. 7(c) at various horizons h. Following the incremental procedure described in Assumption 2, we experimentally determine the minimal horizon at which the synthesized net stabilizes: $h = 4$ for the producer–consumer system, $h = 10$ for the deadlock-free production line, and $h = 11$ for the deadlock-prone production line. To examine algorithmic scalability as problem size grows, we additionally evaluate these systems at larger horizons.

For the third case study, however, generating a monolithic language sample from the complete highly concurrent net in Fig. 8 is computationally intractable, because the size of $\mathcal{L}_{\leq h}$ grows exponentially with h (see Section IV-E). To address this issue, we exploit the structural modularity of the physical system together with language closure properties, such as the language concatenation theorem (Theorem 6.2 in [13]). Specifically, we manually decompose the complete net into nine localized subnets corresponding to the machine and buffer modules of each station. We then extract the language sample of each subnet at its respective minimal stabilizing horizon, synthesize the subnet independently, and finally reconstruct the complete Petri net by manual structural composition.

To illustrate this system-level decomposition-and-composition strategy, Fig. 9 shows the subnet associated with station S1 and its interfaces with the adjacent S5 and S2 stations. As shown, S1 is partitioned into the S1 buffer (black bounding box) and the S1 machine (red bounding box). To enable consistent recomposition, overlapping boundary elements between adjacent subnets are deliberately retained. For example, the shared transition T_1_S1_LOAD is explicitly included in both the S1 buffer and S1 machine subnets. Likewise, interface places and transitions associated with the upstream S5 blocks are preserved within the S1 buffer subnet to ensure a seamless connection with the S5 machine. This composition step is purely interface-based: shared boundary places and transitions are merged according to label consistency, without altering the internally synthesized subnet structures. By preserving these overlapping interfaces, the independently synthesized subnets can be naturally merged into a complete net.

TABLE IV
CONFIGURATION OF THE NUMERICAL EXPERIMENTS.

Case study		Language sample			Parameters	
		h	*pos*	*neg*	K	m
Producer–consumer system		4 7 9	72 1800 15784	28 668 5628	10	6
Parallel production line	Deadlock-free	10 11 12	194 390 812	654 1170 2316	10	12
	Deadlock-prone	11 12	563 1207	1831 3869	10	12
LEGO plant	S1 buffer	12	1290	2590	10	5
	S1 machine	11	72	376	10	6
	S2 buffer	12	2632	4728	10	5
	S2 machine	10	56	272	10	5
	S3 and S4 buffer	6	1291	1909	10	7
	S3 machine	10	22	92	10	5
	S4 machine	10	22	92	10	5
	S5 buffer	6	844	1865	10	8
	S5 machine	11	72	376	10	6

To avoid degenerate minimal solutions (e.g., isolated places or transitions) and to ensure a meaningful algorithmic comparison, we impose structural constraints in all experiments. For the producer–consumer system, we enforce the basic structural constraints (13a)–(13d). For the parallel production line, we additionally enforce (13e)–(13g) to encode known physical properties. In particular, to capture the BAS policy, we impose a P-invariant $\boldsymbol{\phi} = \mathbf{1}_{m\times 1}$ via (13e) for both configurations. To capture the routing branches, we enforce specific T-invariants via (13g): for the deadlock-free configuration (8 transitions; Fig. 7(a)), we use $\boldsymbol{\tau}_1 = (2,0,0,0,2,1,1,1)^\top$ and $\boldsymbol{\tau}_2 = (2,2,1,1,0,0,0,1)^\top$; for the deadlock-prone configuration, which includes the additional fault transition t_9 (Fig. 7(c)), these invariants become $\boldsymbol{\tau}_3 = (2,0,0,0,2,1,1,1,0)^\top$ and $\boldsymbol{\tau}_4 = (2,2,1,1,0,0,0,1,0)^\top$. Finally, for the nine LEGO subnets, we enforce (13a)–(13e), including a conservative P-invariant $\boldsymbol{\phi} = \mathbf{1}_{m\times 1}$ for each subnet.

Regarding optimization settings, the DW-reformulated ILP is solved exactly via B&P. In our experiments, this B&P algorithm is implemented in GCG [52], with CPLEX [53] as the underlying solver for both the master and subproblems.

In the implementation, the problem-specific ingredient is the decomposition itself, namely, the identification of the block structure and the construction of the corresponding master and subproblems, whereas generic components such as branching, node processing, and column management are handled by GCG. Although the GCG B&P framework natively supports parallel subproblem solving, both the B&P (GCG) and the baseline B&C (Gurobi) are restricted to sequential execution (one thread) in all experiments. This setting isolates algorithmic efficiency from hardware-level parallelization effects. In addition, Gurobi is evaluated with its default presolve both enabled and disabled to assess the role of presolve explicitly. A uniform time limit of 3,600 s is imposed on every instance.

To assess robustness, each experiment is repeated ten times using random seeds 1–10, thereby accounting for performance variability induced by the internal heuristics of the B&P and B&C frameworks. All experiments are conducted on a laptop running Ubuntu 24.04, equipped with an Intel® Core™ i7-12700H processor (2.30 GHz, 6 physical/12 logical cores, E-cores disabled) and 64 GB of RAM.

C. Experimental results

The results of the two numerical experiments introduced in Section V-B are presented below. Since the two experiments serve different purposes, different performance metrics are reported accordingly. For the effectiveness experiment, we focus on the synthesized nets and report the objective value and solving time. For the efficiency experiment, we additionally report optimality and feasibility statistics as well as the optimality gap under time limits.

Metrics: The following metrics are used throughout this section:

- *obj. val*: objective value of the optimal solution. For the proposed ILP, this equals the total number of input arcs, output arcs, inhibitor arcs, and the initial marking (number of tokens). For the baseline ILP [11], inhibitor arcs are absent; hence, only input arcs, output arcs, and the initial marking are counted.
- *avg. t*: average solving time (s) over 10 replications, reported together with its 95% confidence interval. If a run terminates at the imposed time limit (3,600 s) without proving optimality, the time limit is recorded as its solving time.
- *no. opt*: number of runs in which optimality was proven.
- *no. feas*: number of runs that returned a feasible incumbent (including runs that were proven optimal).
- *avg. g*: average optimality gap (in percentage) over runs that return a feasible incumbent but do not prove optimality [48]:

$$\text{gap} = \frac{|\text{best bound} - \text{incumbent objective}|}{|\text{incumbent objective}|} \times 100\%, \tag{24}$$

where "best bound" denotes the final dual bound reported by the algorithm and "incumbent objective" denotes the best feasible objective value found. If no feasible solution is obtained, *gap* is not defined and is reported as "–".

Effectiveness experiment results: Table V reports the effectiveness results on the producer–consumer system. Since feasibility of either ILP formulation guarantees consistency with the input language sample up to the corresponding horizon h, the discussion below focuses on structural compactness and correctness.

The proposed ILP consistently attains a lower objective value than the baseline ILP [11], indicating that it synthesizes significantly more compact nets (recall that the objective minimizes the number of arcs and tokens). As the horizon h increases, the objective value of the baseline ILP grows markedly, from 25 to 40, reflecting its structural inability to represent the priority behavior compactly within a P/T net. To fit the enlarged language sample, the baseline formulation is forced to introduce additional arcs and tokens. In contrast, the objective value of the proposed ILP stabilizes at 17 across all tested horizons. Moreover, the proposed formulation returns solutions that are structurally isomorphic to the ground-truth inhibitor net in Fig. 2, where priority is enforced directly by an inhibitor arc (see Fig. 6).

This structural gain comes with an expected computational cost. The average solving time increases with h for both formulations because the ILP size grows with the language sample. At larger horizons, the proposed ILP requires more time than the baseline (e.g., 659.54 s versus 6.76 s at $h = 9$), which is consistent with the added combinatorial difficulty induced by explicitly modeling inhibitor arcs. Nevertheless, the additional computational effort yields a structurally correct and interpretable inhibitor net model, whereas the baseline terminates quickly but produces a more complicated P/T net structure that only mimics the observed priority behavior.

TABLE V
COMPARISON OF THE BASELINE ILP AND THE PROPOSED ILP (AVERAGE SOLVING TIMES OVER 10 REPLICATIONS, 95% CONFIDENCE INTERVALS).

Case study	h	The baseline ILP		The proposed ILP	
		obj. val	avg. t (s)	obj. val	avg. t (s)
Producer–consumer	4	25	< 1.0	17	< 1.0
	7	34	2.06 ± 0.48	17	14.90 ± 3.66
	9	40	6.76 ± 2.95	17	659.54 ± 252.70

Both ILPs can admit multiple optimal solutions, i.e., distinct net topologies that yield the same objective value. For illustration, an optimal solution of the baseline ILP at $h = 7$ is shown in Fig. 5. Compared with the ground-truth net in Fig. 2, it is structurally cluttered and relies on an excessive number of tokens and arcs to reproduce the priority behavior indirectly. In contrast, the proposed ILP returns a net that is fully isomorphic to the ground truth. As shown in Fig. 6, the transition sets coincide, and the place set $\{p_1, p_2, p_3, p_4, p_5, p_6\}$ in Fig. 2 maps to $\{p_2, p_4, p_1, p_5, p_3, p_6\}$ in the synthesized net.

Efficiency experiment results: Table VI reports the efficiency results of our proposed decomposition-based approach and the baseline B&C (with and without presolve) on the three case studies. For the producer–consumer system, structural correctness has already been established in the effectiveness experiment and is therefore not revisited here. For the other two case studies, the synthesized nets for the parallel pro-

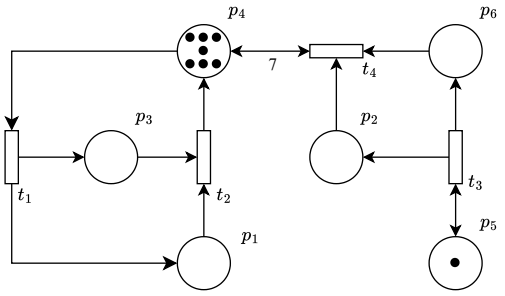

Fig. 5. An optimal solution of the baseline ILP for the producer–consumer system at $h = 7$.

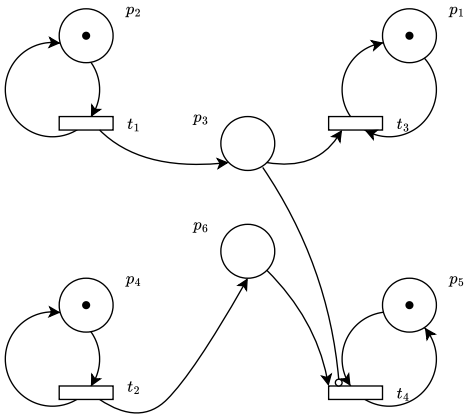

Fig. 6. An optimal solution of the proposed ILP for the producer–consumer system at $h = 7$.

duction line are verified to be structurally isomorphic to the corresponding ground-truth nets in Fig. 7(a) and Fig. 7(c), respectively. And the reconstructed complete net, obtained after manually composing the synthesized LEGO subnets, is structurally isomorphic to the ground-truth net in Fig. 8. Taken together, Tables IV and VI show that computational difficulty is determined not only by sample size, but also by concurrency, blocking structure, and subnet coupling.

This pattern is partially visible in the producer–consumer system. For the smaller instances at $h = 4$ and $h = 7$, B&C with presolve solves all runs to optimality and outperforms our decomposition-based approach in average time (0.19 s versus 2.19 s at $h = 4$, and 14.90 s versus 25.57 s at $h = 7$). This is expected because these instances are still small enough that direct solution of the full ILP is highly effective, whereas our approach still incurs the overhead of decomposition and reformulation. At $h = 9$, however, the language sample grows to $15{,}784$ positive and $5{,}628$ negative examples, and the balance reverses: our approach proves optimality in all 10 runs with an average time of 349.09 s, while B&C with presolve requires 659.54 s. Without presolve, B&C becomes much less stable, proving optimality only once and reaching an average gap of 10.93%. This shows that once the full ILP becomes sufficiently large, our approach can offset the overhead of decomposition and outperform direct B&C.

The advantage of our approach is even more pronounced on the parallel production line, where it proves optimality in all 10 runs for both the deadlock-free and deadlock-prone configurations. For the deadlock-free instances, B&P solves the three horizons in 179.97 s, 274.07 s, and 471.87 s on average, whereas B&C with presolve reaches the $3{,}600$ s time limit in all cases and still leaves nonzero optimality gaps between 8.78% and 24.41%. Disabling presolve weakens B&C further. More importantly, the deadlock-prone configuration is significantly harder than the deadlock-free one: at $h = 11$ and $h = 12$, the average solving time of our approach increases to 1478.16 s and 2117.62 s, respectively, and the performance of B&C deteriorates sharply, with only 8 and 5 feasible runs under presolve and only 2 and 0 feasible runs without presolve. This substantial slowdown cannot be attributed to sample growth alone. Compared with the deadlock-free configuration, the deadlock-prone variant introduces additional blocking dependencies and a more intricate boundary between admissible and inadmissible traces, thereby making the exact synthesis problem intrinsically harder.

The LEGO plant provides a complementary perspective. For the machine modules (S1, S2, S3, S4, and S5 machines), the language samples are very small. For example, the S3 and S4 machines each have only 22 positive and 92 negative examples, and B&C, especially with presolve, is consistently faster, often solving the instances within a few seconds. For the simpler buffer modules, such as the S1 and S2 buffers, B&C with presolve also remains highly competitive and even outperforms our approach, whereas disabling presolve leads to much poorer performance. The situation changes, however, for the more strongly coupled S3 and S4 buffer and S5 buffer modules. Although their sample sizes are not the largest among the nine LEGO plant components in Table IV, our approach proves optimality in all 10 runs, whereas B&C with presolve proves optimality in only 6 and 2 runs, respectively, and B&C without presolve rarely finds a feasible solution. This indicates that sample size alone does not determine computational difficulty; local synchronization patterns and structural coupling within the subnet are equally important.

To better understand why B&C performs well on some instances yet deteriorates rapidly on harder ones, we measured the reduction achieved by presolve before the B&C search. In the producer–consumer case at $h = 7$, presolve reduces the proposed ILP to 31.8% of the constraints and 74.6% of the variables in 0.4 s, which explains why B&C with presolve remains competitive on this instance. In contrast, for the deadlock-prone parallel production line at $h = 12$, presolve reduces the ILP to 15.9% of the constraints and 16.1% of the variables in 6.7 s, yet B&C still cannot close the remaining optimality gap within the time limit. This shows that presolve can remove algebraic redundancy, but it cannot fully overcome the intrinsic combinatorial difficulty caused by concurrency, synchronization, and deadlock-related interactions. Our approach remains superior on these harder instances because it exploits the inherent structure through DW decomposition, and the resulting reformulation typically yields stronger Lagrangian bounds and a more scalable search process.

TABLE VI
COMPARISON OF B&P AND B&C (AVERAGE SOLVING TIMES OVER 10 REPLICATIONS, 95% CONFIDENCE INTERVALS).

Case study		h	Decomposition-based approach		B&C with presolve (Gurobi)				B&C without presolve (Gurobi)			
			no. opt	avg. t (s)	no. opt	no. feas	avg. t (s)	avg. g (%)	no. opt	no. feas	avg. t (s)	avg. g (%)
Producer–consumer		4	10	2.19 ± 0.83	10	10	0.19 ± 0.09	0	10	10	1.07 ± 0.18	0
		7	10	25.57 ± 6.05	10	10	14.90 ± 3.66	0	10	10	80.40 ± 25.08	0
		9	10	349.09 ± 160.01	10	10	659.54 ± 252.70	0	1	10	3423.08 ± 559.47	10.93
Parallel production line	Deadlock-free	10	10	179.97 ± 91.87	0	10	3600	8.78	0	10	3600	23.37
		11	10	274.07 ± 46.51	0	9	3600	23.66	0	4	3600	38.13
		12	10	471.87 ± 137.70	0	10	3600	24.41	0	1	3600	37.25
	Deadlock-prone	11	10	1478.16 ± 353.10	0	8	3600	19.38	0	2	3600	48.21
		12	10	2117.62 ± 437.05	0	5	3600	34.29	0	0	3600	–
LEGO plant	S1 buffer	12	10	122.18 ± 19.47	10	10	14.38 ± 6.77	0	0	10	3600	16.60
	S1 machine	11	10	52.87 ± 12.96	10	10	4.49 ± 1.29	0	10	10	6.19 ± 1.52	0
	S2 buffer	12	10	247.56 ± 22.44	10	10	61.06 ± 34.95	0	0	6	3600	42.76
	S2 machine	10	10	34.58 ± 10.46	10	10	1.86 ± 1.02	0	10	10	2.51 ± 1.03	0
	S3 and S4 buffer	6	10	753.18 ± 512.30	6	7	2481.93 ± 1069.07	5.28	0	1	3600	41.67
	S3 machine	10	10	5.57 ± 1.70	10	10	0.27 ± 0.10	0	10	10	0.26 ± 0.16	0
	S4 machine	10	10	5.68 ± 1.72	10	10	0.27 ± 0.10	0	10	10	0.27 ± 0.16	0
	S5 buffer	6	10	830.77 ± 262.98	2	9	3312.19 ± 672.05	36.14	0	3	3600	46.09
	S5 machine	11	10	53.24 ± 13.13	10	10	4.41 ± 1.28	0	10	10	6.11 ± 1.50	0

VI. CONCLUSION

This paper presented an exact and efficient approach for synthesizing free-labeled Petri nets with inhibitor arcs from prefix-closed language samples. We formulated the synthesis problem as an ILP that searches for a minimum inhibitor net consistent with the given sample. We exploited the single-bordered block-diagonal structure of the ILP to derive a Dantzig–Wolfe decomposition solved by the branch-and-price (B&P) algorithm. The proposed method was validated on three case studies: an unbounded producer–consumer system, a parallel production line, and a lab-scale LEGO manufacturing plant. The results showed that the proposed ILP can recover the ground-truth inhibitor net of the producer–consumer system. They also showed that the proposed decomposition-based approach is markedly more robust and efficient than the baseline B&C in Gurobi on larger and more constrained instances. The results on the LEGO plant further demonstrate the applicability of the approach to a complex physical manufacturing system. Overall, the proposed method can synthesize compact, structurally correct, and behaviorally faithful Petri net models of DES. Such models can serve as the discrete-event core of a digital twin for online monitoring, where token flows track parts and buffer occupancies enable early detection of blocking and bottlenecks, and for offline what-if analysis of dispatching rules, routing policies, and resource allocations.

Future work is motivated by both a theoretical view of the synthesis process and the practical requirements of digital twin development. On the theoretical side, our experiments (see Section V-C) showed that presolve can remove a substantial portion of the ILP before the search begins. This suggests that the example-induced formulation contains nontrivial redundancy and points to two related directions for improving computational efficiency. First, we will develop procedures to *choose the horizon* h more systematically. The objective is to ensure that the sample $\mathcal{L}_{\leq h}$ captures sufficient system behavior for exact synthesis, while avoiding unnecessarily large horizons that cause the ILP to grow rapidly. Second, because the underlying system states are typically unobservable, different event sequences in the sample may reach the same state and therefore impose identical system features on the synthesized net. To address this redundancy, we plan to identify *minimal informative examples* that reduce the ILP without changing the synthesized solution.

On the practical side, supporting industrial-scale digital twins requires addressing both scalability and validation. Therefore, a third research direction focuses on *automated system-level composition*. While our current experiments (the LEGO plant) rely on manual decomposition to handle large-scale systems, fully automating the construction of large Petri nets by merging smaller, independently synthesized subnets is a highly promising avenue, building upon existing literature in the field [44], [31]. Finally, a fourth research direction concerns *faithfulness evaluation*. Even when a synthesized net exactly realizes the observed language up to a suitable horizon, its reliability must be continuously validated against the true physical system. We plan to test synthesized nets against new, previously unseen operational traces, for example, traces generated over a longer horizon $h_{\text{test}} > h$. By integrating conformance checking techniques [54], the synthesized model can be assessed more rigorously beyond the primary data used for synthesis.

APPENDIX A
PROOF OF THE THEOREM 1

We first present two propositions used to linearize the enabling and disabling requirements in the ILP construction. Then we give the detailed proof of Theorem 1.

Proposition 1 (Big-M switch)**.** For $i = 1, \ldots, h$, let $a_i \in \mathbb{Z}$ and $d_i \in \{0, 1\}$. Assume $K \gg \max_i |a_i|$ is a given large constant. Then the logical equivalences

$$a_i \leq -1 \iff d_i = 1 \quad \textit{and} \quad a_i \geq 0 \iff d_i = 0, \quad (25)$$

are encoded by the linear constraints

$$\begin{cases} \boldsymbol{a} \le -\mathbf{1}_{h\times 1} + K\left(\mathbf{1}_{h\times 1} - \boldsymbol{d}\right) & \text{(26a)} \\ \boldsymbol{a} \ge -K\boldsymbol{d}, & \text{(26b)} \end{cases}$$

where $\boldsymbol{a} = (a_i) \in \mathbb{Z}^h$ and $\boldsymbol{d} = (d_i) \in \{0,1\}^h$.

Proof: If $d_i = 1$, then (26a) gives $a_i \le -1$, while (26b) is redundant ($a_i \ge -K$). If $d_i = 0$, then (26b) gives $a_i \ge 0$, and (26a) gives $a_i \le K-1$, which is redundant because $K \gg |a_i|$.

Conversely, if $a_i \le -1$ and $d_i = 0$, then (26b) forces $a_i \ge 0$, a contradiction; hence $d_i = 1$. If $a_i \ge 0$ and $d_i = 1$, then (26a) forces $a_i \le -1$, a contradiction; hence $d_i = 0$. ■

Proposition 2 (at least one)**.** If $a, b \in \mathbb{N}$, then the disjunction

$$(a \ge 1) \quad or \quad (b \ge 1), \tag{27}$$

is equivalently expressed by the single linear inequality

$$a + b \ge 1. \tag{28}$$

Proof: If either $a \ge 1$ or $b \ge 1$, then $a + b \ge 1$. Conversely, if $a + b \ge 1$ with $a, b \in \mathbb{N}$, then at least one of a, b is ≥ 1. ■

Proof of Theorem 1: Let $\mathbf{M}_\sigma$ denote the marking reached by firing σ from the initial marking $\mathbf{M}_0$.

(i) Positive examples — enabling rules.) Given a positive example $\xi_k^+ = \left(\sigma_k^+, t_{j_k^+}\right) \in \mathcal{E}^+$. By the inhibitor net system enabling rule (see (1)–(2) in Section III-A), $t_{j_k^+}$ is enabled at $\mathbf{M}_{\sigma_k^+}$ iff

$$\forall p_i \in {}^\bullet t_{j_k^+}, \mathbf{M}_{\sigma_k^+}(p_i) \ge \mathbf{I}\left(p_i, t_{j_k^+}\right) \tag{29}$$

$$\forall p_i \in {}^\circ t_{j_k^+}, \mathbf{M}_{\sigma_k^+}(p_i) < \mathbf{H}\left(p_i, t_{j_k^+}\right). \tag{30}$$

For the inequality (29):

1) If $p_i \in {}^\bullet t_{j_k^+}$, i.e., p_i is an input place of $t_{j_k^+}$, then $\mathbf{M}_{\sigma_k^+}(p_i) - \mathbf{I}\left(p_i, t_{j_k^+}\right) \ge 0$ must hold.
2) If $p_i \notin {}^\bullet t_{j_k^+}$, i.e., p_i is not an input place of $t_{j_k^+}$, then $\mathbf{I}\left(p_i, t_{j_k^+}\right) = 0$, and $\mathbf{M}_{\sigma_k^+}(p_i) - \mathbf{I}\left(p_i, t_{j_k^+}\right) \ge 0$ holds naturally.

The inequality $\mathbf{M}_{\sigma_k^+}(p_i) - \mathbf{I}\left(p_i, t_{j_k^+}\right) \ge 0$ is already linear; substituting $\mathbf{M}_{\sigma_k^+}$ with the equation (4), i.e., $\mathbf{M}_{\sigma_k^+} = \mathbf{M}_0 + \mathbf{C}\boldsymbol{\sigma}_k^+$ yields the constraint (12a).

And for the inequality (30):

1) If $p_i \in {}^\circ t_{j_k^+}$, i.e., p_i is an inhibitor place of $t_{j_k^+}$, then $\mathbf{M}_{\sigma_k^+}(p_i) - \mathbf{H}\left(p_i, t_{j_k^+}\right) < 0$ must hold. Because all variables are integers, this implies $\mathbf{M}_{\sigma_k^+}(p_i) - \mathbf{H}\left(p_i, t_{j_k^+}\right) \le -1$.
2) If $p_i \notin {}^\circ t_{j_k^+}$, i.e., p_i is not an inhibitor place of $t_{j_k^+}$, then $\mathbf{H}\left(p_i, t_{j_k^+}\right) = 0$, and $\mathbf{M}_{\sigma_k^+}(p_i) - \mathbf{H}\left(p_i, t_{j_k^+}\right) \ge 0$ should hold naturally.

To linearize these above two logical conditions, we introduce a binary matrix $\mathbf{X} \in \{0,1\}^{m\times n}$, where $\mathbf{X}\left(p_i, t_{j_k^+}\right) = 1$ holds the former condition, and $\mathbf{X}\left(p_i, t_{j_k^+}\right) = 0$ holds the latter condition. Applying Proposition 1 with $a_i = \mathbf{M}_{\sigma_k^+}(p_i) - \mathbf{H}\left(p_i, t_{j_k^+}\right)$ and $d_i = \mathbf{X}\left(p_i, t_{j_k^+}\right)$, and representing $\mathbf{M}_{\sigma_k^+}$ with initial marking $\mathbf{M}_0$ using the equation (4), yields the linear constraints (12b) and (12c).

(ii) Negative examples — disabling rules.)

Given a negative example $\xi_l^- = \left(\sigma_l^-, t_{j_l^-}\right) \in \mathcal{E}^-$. The transition $t_{j_l^-}$ is disabled at $\mathbf{M}_{\sigma_l^-}$ iff

$$\exists p_i \in {}^\bullet t_{j_l^-}, \mathbf{M}_{\sigma_l^-}(p_i) < \mathbf{I}\left(p_i, t_{j_l^-}\right) \tag{31}$$

or

$$\exists p_i \in {}^\circ t_{j_l^-}, \mathbf{M}_{\sigma_l^-}(p_i) \ge \mathbf{H}\left(p_i, t_{j_l^-}\right). \tag{32}$$

For the inequality (31):

1) If p_i is an input place of $t_{j_l^-}$ and disables it at $\mathbf{M}_{\sigma_l^-}$, then $\mathbf{M}_{\sigma_l^-}(p_i) - \mathbf{I}\left(p_i, t_{j_l^-}\right) \le -1$ must hold.
2) If p_i is an input place of $t_{j_l^-}$ and enables it at $\mathbf{M}_{\sigma_l^-}$ *or* p_i is not an input place of $t_{j_l^-}$, then $\mathbf{M}_{\sigma_l^-}(p_i) - \mathbf{I}\left(p_i, t_{j_l^-}\right) \ge 0$ must hold.

Analogously, to linearize these above two logical conditions, we introduce a binary matrix $\mathbf{Y} \in \{0,1\}^{m\times s}$, where $\mathbf{Y}\left(p_i, \xi_l^-\right) = 1$ holds the first condition, and $\mathbf{Y}\left(p_i, \xi_l^-\right) = 0$ holds the second one. Utilizing Proposition 1 with $a_i = \mathbf{M}_{\sigma_l^-}(p_i) - \mathbf{I}\left(p_i, t_{j_l^-}\right)$ and $d_i = \mathbf{Y}\left(p_i, \xi_l^-\right)$, and substituting $\mathbf{M}_{\sigma_l^-}$ with $\mathbf{M}_0$ using the equation (4), yields the linear constraints (12d) and (12e).

And for the inequality (32):

1) If p_i is an inhibitor place of $t_{j_l^-}$ and disables it at $\mathbf{M}_{\sigma_l^-}$ *or* p_i is not an inhibitor place of $t_{j_l^-}$, $\mathbf{M}_{\sigma_l^-}(p_i) - \mathbf{H}\left(p_i, t_{j_l^-}\right) \ge 0$ must hold.
2) If p_i is an inhibitor place of $t_{j_l^-}$ and enables it, then $\mathbf{M}_{\sigma_l^-}(p_i) - \mathbf{H}\left(p_i, t_{j_l^-}\right) \le -1$ must hold.

To linearize these two logical conditions, we introduce a binary matrix $\mathbf{Z} \in \{0,1\}^{m\times s}$, where $\mathbf{Z}\left(p_i, \xi_l^-\right) = 0$ holds the former one, and $\mathbf{Z}\left(p_i, \xi_l^-\right) = 1$ holds the latter one. Utilizing Proposition 1 with $a_i = \mathbf{M}_{\sigma_l^-}(p_i) - \mathbf{H}\left(p_i, t_{j_l^-}\right)$ and $d_i = \mathbf{Z}\left(p_i, \xi_l^-\right)$, and rewriting $\mathbf{M}_{\sigma_l^-}$ by $\mathbf{M}_0$ using (4), yields the linear constraints (12f) and (12g).

Here we have:

1) $\sum_{i=1}^m \mathbf{X}\left(:, t_{j_l^-}\right)$ represents the total number of inhibitor places of $t_{j_l^-}$.
2) $\sum_{i=1}^m \mathbf{Y}\left(:, \xi_l^-\right)$ represents the total number of input places of $t_{j_l^-}$ that disable it at $\mathbf{M}_{\sigma_l^-}$.
3) $\sum_{i=1}^m \mathbf{Z}\left(:, \xi_l^-\right)$ represents the total number of inhibitor places of $t_{j_l^-}$ that enable it at $\mathbf{M}_{\sigma_l^-}$.
4) $\sum_{i=1}^m \left(\mathbf{X}\left(:, t_{j_l^-}\right) - \mathbf{Z}\left(:, \xi_l^-\right)\right)$ represents the total number of inhibitor places of $t_{j_l^-}$ that disable it at $\mathbf{M}_{\sigma_l^-}$.

At last, the disabling rule requires that *at least one* of the inequalities (31)–(32) holds, i.e., there exists at least one place, either an input place or an inhibitor place of $t_{j_l^-}$ that disables

it at $\mathbf{M}_{\sigma_l^-}$. Using the above column sums and Proposition 2, the disjunction is enforced by the single linear constraint

$$\sum_{i=1}^{m}\left(\mathbf{X}\left(p_i,t_{j_l^-}\right)-\mathbf{Z}\left(p_i,\xi_l^-\right)\right)+\sum_{i=1}^{m}\mathbf{Y}\left(p_i,\xi_l^-\right)\ \geq\ 1,\qquad \forall\,\xi_l^-\in\mathcal{E}^-, \tag{33}$$

which is the constraint (12h).

(iii) Conclusion.) Parts (i)–(ii) show that $\mathcal{G}\left(\mathcal{E}^+,\mathcal{E}^-\right)$ is equivalent to requiring that, for every pair $(\sigma,t)\in\mathcal{L}_{\leq h-1}\times T$, the transition t is enabled at $\mathbf{M}_\sigma$ if $(\sigma,t)\in\mathcal{E}^+$ and disabled at $\mathbf{M}_\sigma$ if $(\sigma,t)\in\mathcal{E}^-$. Since $\mathcal{E}^+$ and $\mathcal{E}^-$ form a partition of $\mathcal{L}_{\leq h-1}\times T$, the enabled transition set of the synthesized system at every prefix marking $\mathbf{M}_\sigma$, with $\sigma\in\mathcal{L}_{\leq h-1}$, coincides exactly with that specified by the sample $\mathcal{L}_{\leq h}$. It follows that every one-step extension σt with $|\sigma t|\leq h$ is admitted by INS iff $\sigma t\in\mathcal{L}_{\leq h}$. Starting from the empty sequence $\epsilon\in\mathcal{L}_{\leq h}$ and applying this equivalence recursively over sequence length, we obtain that the set of all firing sequences of INS up to horizon h is exactly the given sample, i.e., $\mathcal{L}_{\leq h}\left(INS\right)=\mathcal{L}_{\leq h}$. Therefore, INS is a solution to Problem 1. ∎

APPENDIX B
STANDARDIZATION OF THE ORIGINAL INTEGER LINEAR PROGRAM

We standardize the original ILP (10) by constructing the SF (15) quantities $(\boldsymbol{\alpha},\boldsymbol{\gamma},\mathbf{A},\boldsymbol{\beta})$ from the original ILP decision variables $(\mathbf{I},\mathbf{O},\mathbf{H},\mathbf{M}_0,\mathbf{X},\mathbf{Y},\mathbf{Z})$, together with other auxiliary vectors (e.g., all-ones/all-zeros and selector vectors). In the following construction, we use the symbols $\oplus$ and $\ominus$ to denote block concatenation. For column vectors $\boldsymbol{u}\in\mathbb{R}^{r\times 1}$ and $\boldsymbol{v}\in\mathbb{R}^{s\times 1}$,

$$\boldsymbol{u}\oplus\boldsymbol{v}=\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{v}\end{bmatrix}, \tag{34}$$

For row vectors $\boldsymbol{u}'\in\mathbb{R}^{1\times r}$ and $\boldsymbol{v}'\in\mathbb{R}^{1\times s}$,

$$\boldsymbol{u}'\ominus\boldsymbol{v}'=\begin{bmatrix}\boldsymbol{u}' & \boldsymbol{v}'\end{bmatrix}. \tag{35}$$

The decision variable vector $\boldsymbol{\alpha}$ in the SF (15) is a column vector obtained as

$$\boldsymbol{\alpha}\ =\ \boldsymbol{\alpha}_1\oplus\cdots\oplus\boldsymbol{\alpha}_m, \tag{36}$$

where, for each $i\in\{1,\ldots,m\}$,

$$\begin{aligned}\boldsymbol{\alpha}_i=\mathbf{I}\left(p_i,:\right)^\top\oplus\mathbf{O}\left(p_i,:\right)^\top\oplus\mathbf{H}\left(p_i,:\right)^\top\oplus\mathbf{M}_0\left(p_i\right)\\ \oplus\mathbf{X}\left(p_i,:\right)^\top\oplus\mathbf{Y}\left(p_i,:\right)^\top\oplus\mathbf{Z}\left(p_i,:\right)^\top.\end{aligned} \tag{37}$$

Note that $\boldsymbol{\alpha}_i$ consists of all the decision variables associated with the place p_i.

The objective function coefficient $\boldsymbol{\gamma}$ in the SF (15) is a row vector

$$\boldsymbol{\gamma}=\boldsymbol{\gamma}_1\ominus\cdots\ominus\boldsymbol{\gamma}_i\ominus\cdots\ominus\boldsymbol{\gamma}_m, \tag{38}$$

where, for each $i\in\{1,\ldots,m\}$,

$$\boldsymbol{\gamma}_i=\mathbf{1}_{1\times n}\ominus\mathbf{1}_{1\times n}\ominus\mathbf{1}_{1\times n}\ominus 1\ominus\mathbf{0}_{1\times n}\ominus\mathbf{0}_{1\times s}\ominus\mathbf{0}_{1\times s}. \tag{39}$$

Objective function standardization. By (36)–(39), the objective function of the SF in (15a), namely $\boldsymbol{\gamma}\boldsymbol{\alpha}$, is equal to the sum of all entries of $\mathbf{I}$, $\mathbf{O}$, $\mathbf{H}$, and $\mathbf{M}_0$, and thus coincides with the objective function of the original ILP in (11).

The constraint coefficient matrix $\mathbf{A}$ in the SF (15) is a single-bordered block-diagonal matrix

$$\mathbf{A}=\begin{pmatrix}\mathbf{B}\\ \mathbf{D}\end{pmatrix}=\begin{pmatrix}\mathbf{B}_1 & \cdots & \mathbf{B}_i & \cdots & \mathbf{B}_m\\ \mathbf{D}_1 & \cdots & 0 & \cdots & 0\\ \vdots & \ddots & \vdots & & \vdots\\ 0 & \cdots & \mathbf{D}_i & \cdots & 0\\ \vdots & & \vdots & \ddots & \vdots\\ 0 & \cdots & 0 & \cdots & \mathbf{D}_m\end{pmatrix} \tag{40}$$

where $\mathbf{B}$ is a row vector obtained by block concatenation $\mathbf{B}=\mathbf{B}_1\ominus\cdots\ominus\mathbf{B}_m$, and each block $\mathbf{B}_i$ has the row pattern

$$\mathbf{B}_i=\mathbf{0}_{1\times n}\ominus\mathbf{0}_{1\times n}\ominus\mathbf{0}_{1\times n}\ominus 0\ominus\boldsymbol{B}_{i,1}\ominus\boldsymbol{B}_{i,2}\ominus\boldsymbol{B}_{i,3}, \tag{41}$$

and the subvectors:

$$\begin{cases}\boldsymbol{B}_{i,1}=(0,\ldots,-1,\ldots,0)\in\{0,-1\}^{1\times n}, & \text{(42a)}\\ \boldsymbol{B}_{i,2}=(0,\ldots,-1,\ldots,0)\in\{0,-1\}^{1\times s}, & \text{(42b)}\\ \boldsymbol{B}_{i,3}=(0,\ldots,1,\ldots,0)\in\{0,1\}^{1\times s}, & \text{(42c)}\end{cases}$$

whose nonzero entry is at the j-th position for $\boldsymbol{B}_{i,1}$ and at the l-th position for $\boldsymbol{B}_{i,2},\boldsymbol{B}_{i,3}$.

The block diagonal $\mathbf{D}=\operatorname{diag}\left(\mathbf{D}_1,\ldots,\mathbf{D}_m\right)$ consists of m smaller matrices. Let $r=|\mathcal{E}^+|$ and $s=|\mathcal{E}^-|$ denote the number of positive and negative examples, respectively. Each smaller matrix $\mathbf{D}_i$ has $(3r+4s)$ rows and seven block columns as shown in (43).

The sub-vectors $\boldsymbol{D}_{i,d}$ for each $d\in\{1,\ldots,6\}$ of the smaller matrix $\mathbf{D}_i$ are

$$\begin{cases}\boldsymbol{D}_{i,1}=(0,\ldots,\ K,\ldots,0)\in\{0,K\}^{1\times n}, & \text{(44a)}\\ \boldsymbol{D}_{i,2}=(0,\ldots,-K,\ldots,0)\in\{0,-K\}^{1\times n}, & \text{(44b)}\\ \boldsymbol{D}_{i,3}=(0,\ldots,\ K,\ldots,0)\in\{0,K\}^{1\times s}, & \text{(44c)}\\ \boldsymbol{D}_{i,4}=(0,\ldots,-K,\ldots,0)\in\{0,-K\}^{1\times s}, & \text{(44d)}\\ \boldsymbol{D}_{i,5}=(0,\ldots,-K,\ldots,0)\in\{0,-K\}^{1\times s}, & \text{(44e)}\\ \boldsymbol{D}_{i,6}=(0,\ldots,\ K,\ldots,0)\in\{0,K\}^{1\times s}, & \text{(44f)}\end{cases}$$

with the nonzero at the j-th position for $d\in\{1,2\}$ and at the l-th position for $d\in\{3,4,5,6\}$.

The right-hand side $\boldsymbol{\beta}$ in the SF (15) is a column vector

$$\boldsymbol{\beta}=\boldsymbol{\beta}_0\oplus\boldsymbol{\beta}_1\oplus\cdots\oplus\boldsymbol{\beta}_m, \tag{45}$$

where

$$\boldsymbol{\beta}_0=-\mathbf{1}_{s\times 1} \tag{46}$$

and for each $i\in\{1,\ldots,m\}$,

$$\begin{aligned}\boldsymbol{\beta}_i=\mathbf{0}_{r\times 1}\oplus\left((K-1)\ \mathbf{1}_{r\times 1}\right)\oplus\mathbf{0}_{r\times 1}\oplus\left((K-1)\ \mathbf{1}_{s\times 1}\right)\\ \oplus\mathbf{0}_{s\times 1}\oplus\mathbf{0}_{s\times 1}\oplus\left((K-1)\ \mathbf{1}_{s\times 1}\right).\end{aligned} \tag{47}$$

Constraints standardization. By (37), (41)–(42), and (46), the linking constraint $\mathbf{B}\boldsymbol{\alpha}\leq\boldsymbol{\beta}_0$ in (16a) is equivalent to constraint (12h) in the original ILP. Likewise, using (37), (43)–(44), and (47), we obtain that, for each $i\in\{1,\ldots,m\}$, the block inequality $\mathbf{D}_i\boldsymbol{\alpha}^i\leq\boldsymbol{\beta}_i$ in (16b)–(16d) coincides with the i-th row of constraints (12a)–(12g), that is, the

$$\mathbf{D}_i = \begin{pmatrix} \left(\boldsymbol{\sigma}_k^+ + \boldsymbol{e}_{j_k^+}\right)^\top & -\left(\boldsymbol{\sigma}_k^+\right)^\top & \mathbf{0}_{1\times n} & -1 & \mathbf{0}_{1\times n} & \mathbf{0}_{1\times s} & \mathbf{0}_{1\times s} \\ -\left(\boldsymbol{\sigma}_k^+\right)^\top & \left(\boldsymbol{\sigma}_k^+\right)^\top & -\boldsymbol{e}_{j_k^+}^\top & 1 & \boldsymbol{D}_{i,1} & \mathbf{0}_{1\times s} & \mathbf{0}_{1\times s} \\ \left(\boldsymbol{\sigma}_k^+\right)^\top & -\left(\boldsymbol{\sigma}_k^+\right)^\top & \boldsymbol{e}_{j_k^+}^\top & -1 & \boldsymbol{D}_{i,2} & \mathbf{0}_{1\times s} & \mathbf{0}_{1\times s} \\ -\left(\boldsymbol{\sigma}_l^- + \boldsymbol{e}_{j_l^-}\right)^\top & \left(\boldsymbol{\sigma}_l^-\right)^\top & \mathbf{0}_{1\times n} & 1 & \mathbf{0}_{1\times n} & \boldsymbol{D}_{i,3} & \mathbf{0}_{1\times s} \\ \left(\boldsymbol{\sigma}_l^- + \boldsymbol{e}_{j_l^-}\right)^\top & -\left(\boldsymbol{\sigma}_l^-\right)^\top & \mathbf{0}_{1\times n} & -1 & \mathbf{0}_{1\times n} & \boldsymbol{D}_{i,4} & \mathbf{0}_{1\times s} \\ \left(\boldsymbol{\sigma}_l^-\right)^\top & -\left(\boldsymbol{\sigma}_l^-\right)^\top & \boldsymbol{e}_{j_l^-}^\top & -1 & \mathbf{0}_{1\times n} & \mathbf{0}_{1\times s} & \boldsymbol{D}_{i,5} \\ -\left(\boldsymbol{\sigma}_l^-\right)^\top & \left(\boldsymbol{\sigma}_l^-\right)^\top & -\boldsymbol{e}_{j_l^-}^\top & 1 & \mathbf{0}_{1\times n} & \mathbf{0}_{1\times s} & \boldsymbol{D}_{i,6} \end{pmatrix} \tag{43}$$

enabling/disabling requirements induced by the positive and negative examples for place p_i.

Among the optional structural constraints in (13a)–(13d), (13e), and (13g), constraints (13c), (13d), and (13e) act as linking constraints and are therefore represented in the coefficient $\mathbf{B}$, whereas constraints (13a), (13b), and (13g) are block separable and contribute to the block matrices $\mathbf{D}_i$. In the standardization process, these structural constraints are treated in the same way as the enabling/disabling constraints discussed above; for brevity, we do not give their standardized forms.

APPENDIX C
GROUND-TRUTH PETRI NET MODELS OF THE CASE STUDIES

This appendix provides the detailed structural description of the ground-truth Petri net models for the parallel production line (discussed in Section V-A) and the lab-scale LEGO manufacturing plant (discussed in Section V-A).

Ground-truth models of the parallel production line: Fig. 7 shows three Petri net models of the parallel production line: an inhibitor net (Fig. 7(a)), its equivalent P/T net (Fig. 7(b)), and a deadlock-prone inhibitor net (Fig. 7(c)). The first two models correspond to the deadlock-free configuration under normal operation, while the third represents the deadlock-prone configuration under the fault-injection scenario.

For ease of explanation, we refer to the machines of S1, S2, S3, and S4 as M1, M2, M3, and M4, and the buffers of S2 and S3 as B2 and B3. The states of the buffers and machines are described by pairs of places. More precisely, place pairs (p_1, p_2), (p_5, p_6), (p_9, p_{10}), and (p_{11}, p_{12}) characterize the states of machines M1, M2, M3, and M4, while place pairs (p_3, p_4) and (p_7, p_8) characterize the states of buffers B2 and B3. The nominal nets share the same set of regular transitions, each corresponding to a unique event occurring in the production line. For example, transition t_2 denotes the event of moving a part from machine M1 to buffer B2. The BAS policy is captured by input arcs (p_3, t_2), (p_7, t_5), (p_{11}, t_4), and (p_{11}, t_7). Take transition t_2 as an example. Because of the presence of input arc (p_3, t_2), transition t_2 is disabled even if place p_2 contains a token (i.e., machine M1 is busy), but place p_3 is empty (i.e., buffer B2 is full). The ROU policy is captured by transitions t_4 and t_7, which denote the events of moving a part from machines M2 and M3 to machine M4, respectively. When places p_6 and p_{10} are both not empty (i.e., machines M2 and M3 are both busy), and place p_{11} contains a token (i.e., machine M4 is idle), either transition t_4 or t_7 may fire first. Such nondeterministic competition mimics the ROU policy. The difference between the two nominal models lies in how they represent the PBR policy. Note that transition t_5 denotes the event of moving a part from machine M1 to buffer B3. In the inhibitor net (Fig. 7(a)), the priority of buffer B2 over B3 is captured by inhibitor arc (p_3, t_5) in the sense that transition t_5 can be enabled only when place p_3 is empty (i.e., buffer B2 is full). In the P/T net (Fig. 7(b)), inhibitor arc (p_3, t_5) is replaced by an equivalent structure composed of place p_4' and multiple arcs (t_2, p_4'), (p_4', t_3), (p_4', t_5), and (t_5, p_4'). Place p_4' is called the complementary place of p_4 [45].

To evaluate our synthesis algorithm under anomalous conditions, we extend the nominal inhibitor net (Fig. 7(a)) by introducing a fault-injection scenario, as shown in Fig. 7(c). Specifically, we add a fault transition t_9 that consumes the token from p_{11} without returning it to p_{12}. This simulates a sudden catastrophic breakdown at Station S4. Dynamically, the firing of t_9 drives the system into an invalid state where $p_{11}+p_{12}=0$. Because the BAS control policy strictly requires downstream availability to release upstream parts, this single point of failure at S4 triggers a cascading deadlock across the entire line, preventing any further transition firings.

Ground-truth model of the lab-scale LEGO manufacturing plant: Fig. 8 shows the ground-truth Petri net model of the LEGO plant, which was developed and visualized using the GreatSPN [55]. In this model, the states of the system are represented by places, while discrete events (e.g., LOAD, PROCESS, UNLOAD, BLOCK, TRANSFER, and FAIL) are modeled by transitions. For brevity, we refer to the machines of stations S1 through S5 as M1 through M5, and their respective preceding buffers as B1 through B5.

Due to the high complexity of the complete net, we focus on station S1 and its interfaces with the upstream (S5) and downstream (S2) stations as a representative module to explain the modeling rationale. As presented in Fig. 9, the state of buffer B1 is characterized by a pair of places indicating the available slots and the current level, i.e., P_7_S1_buffer_slot and P_8_S1_buffer_level. The states of machine M1 are captured by specific places such as P_1_S1_machine_loading and P_2_S1_machine_idle. State evolutions are characterized by transitions corresponding to those discrete events. For example, the LOAD event is modeled by transition T_1_S1_LOAD;

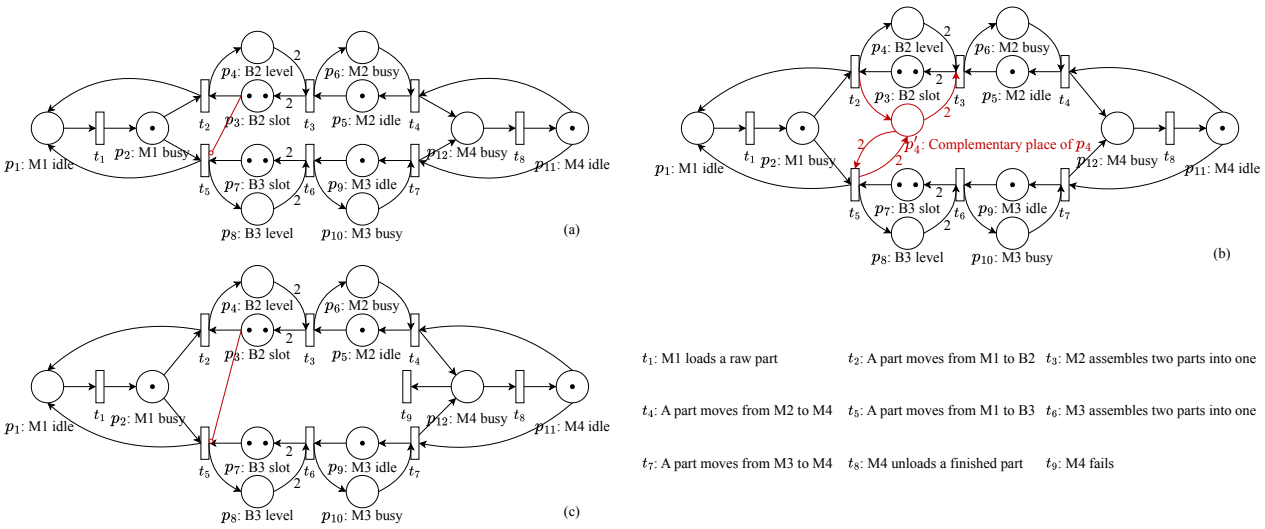

Fig. 7. Detailed Petri net models of the parallel production line: (a) Inhibitor net model. (b) P/T net model. (c) Deadlock-prone inhibitor net model.

upon firing this transition, machine M1 transitions from an idle to a loading state, meaning a token passes from place P_2_S1_machine_idle to P_1_S1_machine_loading. For stations subject to potential breakdowns (i.e., S1 and S5), the FAIL event occurs after the processing phase. Consequently, the UNLOAD event is structurally split into multiple transitions to capture alternative state evolutions. As illustrated in Fig. 9, after processing, M1 may either undergo a normal unload (firing transition T_3_S1_UNLOAD_1 and entering an unloading state, P_5_S1_machine_unloading) or experience a failure (firing T_4_S1_FAIL). In the latter case, the system enters a repairing state (P_4_S1_machine_repairing) before firing the alternative unload transition (T_5_S1_UNLOAD_2).

The part flows in the LEGO plant are strictly governed by the BAS policy. As illustrated in Fig. 9, if buffer B2 is full (i.e., P_9_S2_buffer_slot is empty), the regular transfer transition T_6_S1_TRANSFER_1 is disabled. Instead, the inhibitor arc connecting P_9_S2_buffer_slot to T_7_S1_BLOCK enables the block transition. Firing T_7_S1_BLOCK forces M1 into a blocked state (P_6_S1_machine_blocking). M1 remains blocked until a slot becomes available in B2, which subsequently enables the delayed transfer transition (T_8_S1_TRANSFER_2). Consequently, the TRANSFER event is also modeled using multiple routing transitions to accommodate these varying conditions. Regarding the routing policies, as shown in Fig. 8, there are four transitions modeling the TRANSFER events of machine M2: T_{10}_S2_TRANSFER_1, T_{12}_S2_TRANSFER_2, T_{13}_S2_TRANSFER_3, and T_{14}_S2_TRANSFER_4. The first two transitions model the part transfer from M2 to buffer B3, while the latter two model the transfer from M2 to buffer B4. Taking the unblocked state condition of M2 as an example, both T_{10}_S2_TRANSFER_1 and T_{13}_S2_TRANSFER_3 share the input place P_{12}_S2_machine_unloading. When downstream P_{16}_S3_buffer_slot and P_{23}_S4_buffer_slot are both available, either transition may fire, accurately mimicking the nondeterministic routing policy for the parallel branches. Finally, the ROU policy governing the merging of parts from S3 and S4 into S5 is modeled using a similar nondeterministic transition logic.

REFERENCES

[1] Z. Li and M. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. London: Springer, 2009.

[2] M. Dotoli, M. P. Fanti, A. M. Mangini, and W. Ukovich, "On-line fault detection in discrete event systems by Petri nets and integer linear programming," *Automatica*, vol. 45, no. 11, pp. 2665–2672, 2009.

[3] M. P. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531–1539, 2010.

[4] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[5] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated Discovery of Process Models from Event Logs: Review and Benchmark," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 4, pp. 686–705, 2019.

[6] A. Ehrenfeucht and G. Rozenberg, "Partial (set) 2-structures: Part II," *Acta Informatica*, vol. 27, no. 4, pp. 343–368, 1990.

[7] ——, "Partial (set) 2-structures: Part I," *Acta Informatica*, vol. 27, no. 4, pp. 315–342, 1990.

[8] G. Lugaresi and A. Matta, "Automated manufacturing system discovery and digital twin generation," *Journal of Manufacturing Systems*, vol. 59, pp. 51–66, 2021.

[9] J. Friederich, D. P. Francis, S. Lazarova-Molnar, and N. Mohamed, "A framework for data-driven digital twins of smart manufacturing systems," *Computers in Industry*, vol. 136, p. 103586, 2022.

[10] S. Hu and Z. Li, "A Digital Twin Approach for Enforcing Diagnosability in Petri Nets," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 6068–6080, 2024.

[11] A. Giua and C. Seatzu, "Identification of free-labeled petri nets via integer programming," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 7639–7644.

[12] M. Dotoli, M. P. Fanti, and A. M. Mangini, "Real time identification of discrete event systems using Petri nets," *Automatica*, vol. 44, no. 5, pp. 1209–1219, 2008.

[13] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[14] M. Gribaudo and M. Sereno, "GSPN semantics for queueing networks with blocking," in *Proceedings of the Seventh International Workshop on Petri Nets and Performance Models*, 1997, pp. 26–35.

[15] Y. Chen, Z. Li, K. Barkaoui, and M. Uzam, "New Petri Net Structure and Its Application to Optimal Supervisory Control: Interval Inhibitor Arcs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 10, pp. 1384–1400, 2014.

[16] Y. Chen, Z. Li, K. Barkaoui, N. Wu, and M. Zhou, "Compact Supervisory Control of Discrete Event Systems by Petri Nets With Data Inhibitor Arcs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 364–379, 2017.

[17] E. Badouel, L. Bernardinello, and P. Darondeau, *Petri Net Synthesis*. Berlin, Heidelberg: Springer, 2015.

[18] ——, "Polynomial algorithms for the synthesis of bounded nets," in *TAPSOFT '95: Theory and Practice of Software Development*, P. D. Mosses, M. Nielsen, and M. I. Schwartzbach, Eds. Berlin, Heidelberg: Springer, 1995, pp. 364–378.

[19] T. Tapia-Flores, E. López-Mellado, A. P. Estrada-Vargas, and J.-J. Lesage, "Discovering Petri Net Models of Discrete-Event Processes by Computing T-Invariants," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 992–1003, 2018.

[20] T. Agerwala and M. Flynn, "Comments on capabilities, limitations and "correctness" of Petri nets," in *Proceedings of the 1st Annual Symposium on Computer Architecture*. New York, NY, USA: Association for Computing Machinery, 1973, pp. 81–86.

[21] S. R. Kosaraju, "Limitations of Dijkstra's Semaphore Primitives and Petri nets," *SIGOPS Oper. Syst. Rev.*, vol. 7, no. 4, pp. 122–126, 1973.

[22] M. Mukund, "Petri nets and step transition systems," *International Journal of Foundations of Computer Science*, vol. 03, no. 04, pp. 443–478, 1992.

[23] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev, "Deriving petri nets from finite transition systems," *IEEE Transactions on Computers*, vol. 47, no. 8, pp. 859–882, 1998.

[24] J. Carmona, J. Cortadella, and M. Kishinevsky, "New region-based algorithms for deriving bounded petri nets," *IEEE Transactions on Computers*, vol. 59, no. 3, pp. 371–384, 2010.

[25] N. Busi and G. M. Pinna, "Synthesis of nets with inhibitor arcs," in *CONCUR '97: Concurrency Theory*, A. Mazurkiewicz and J. Winkowski, Eds. Berlin, Heidelberg: Springer, 1997, pp. 151–165.

[26] M. Pietkiewicz-Koutny, "The synthesis problem for elementary net systems with inhibitor arcs," *Fundamenta Informaticae*, vol. 40, no. 2-3, pp. 251–283, 1999.

[27] R. Lorenz, S. Mauser, and R. Bergenthum, "Theory of regions for the synthesis of inhibitor nets from scenarios," in *Petri Nets and Other Models of Concurrency – ICATPN 2007*, J. Kleijn and A. Yakovlev, Eds. Berlin, Heidelberg: Springer, 2007, pp. 342–361.

[28] R. Lorenz and G. Juhás, "Towards Synthesis of Petri Nets from Scenarios," in *Petri Nets and Other Models of Concurrency - ICATPN 2006*, S. Donatelli and P. S. Thiagarajan, Eds. Berlin, Heidelberg: Springer, 2006, pp. 302–321.

[29] P. Darondeau, "Unbounded Petri Net Synthesis," in *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, J. Desel, W. Reisig, and G. Rozenberg, Eds. Berlin, Heidelberg: Springer, 2004, pp. 413–438.

[30] S. Peng and H. Szczerbicka, "Improvements to the Region-Based Petri Nets Synthesis Algorithm for Process Mining," in *Simulation Tools and Techniques*, A. A. Juan, J.-L. Guisado-Lizar, M.-J. Morón-Fernández, and E. Perez-Bernabeu, Eds. Cham: Springer Nature Switzerland, 2025, pp. 186–201.

[31] R. Bergenthum and J. Kovář, "Synthesizing Petri Nets from Labelled Petri Nets," in *Application and Theory of Petri Nets and Concurrency*, E. Amparore and Ł. Mikulski, Eds. Cham: Springer Nature Switzerland, 2025, pp. 63–85.

[32] M. E. Meda-Campaña and E. López-Mellado, "Identification of concurrent discrete event systems using Petri nets," in *Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics*, 2005, pp. 11–15.

[33] A. P. Estrada-Vargas, E. López-Mellado, and J.-J. Lesage, "Input–output identification of controlled discrete manufacturing systems," *International Journal of Systems Science*, vol. 45, no. 3, pp. 456–471, 2014.

[34] ——, "A Black-Box Identification Method for Automated Discrete-Event Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 3, pp. 1321–1336, 2017.

[35] R. Pomares-Angelino and E. López-Mellado, "Discovering petri nets including silent transitions. A repairing approach based on structural patterns," *Discrete Event Dynamic Systems*, vol. 32, no. 2, pp. 291–315, 2022.

[36] E. López-Mellado and Y. Álvarez-Pérez, "Enhancement of Petri nets identified from discrete-event processes by scheduling t-components," *Discrete Event Dynamic Systems*, vol. 35, no. 4, pp. 335–354, 2025.

[37] M. Montes-Partida and E. López-Mellado, "Identification and Assessment of Timed Petri Nets," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 23 745–23 756, 2025.

[38] M. P. Cabasino, A. Giua, and C. Seatzu, "Identification of Petri Nets from Knowledge of Their Language," *Discrete Event Dynamic Systems*, vol. 17, no. 4, pp. 447–474, 2007.

[39] ——, "Identification of unbounded Petri nets from their coverability graph," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 434–440.

[40] G. Zhu, L. Yin, Y. Li, Z. Li, and N. Wu, "Identification of labeled Petri nets from finite automata," *Information Sciences*, vol. 667, p. 120488, 2024.

[41] W. Cheng, L. Zhu, and A. Matta, "Synthesis of Free-Labeled Petri Nets with Inhibitor Arcs," in *IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024, pp. 2819–2824.

[42] F. Basile, P. Chiacchio, and J. Coppola, "Identification of time petri net models," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 9, pp. 2586–2600, 2017.

[43] S. Ould El Mehdi, R. Bekrar, N. Messai, E. Leclercq, D. Lefebvre, and B. Riera, "Design and Identification of Stochastic and Deterministic Stochastic Petri Nets," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, no. 4, pp. 931–946, 2012.

[44] P. Denno, C. Dickerson, and J. A. Harding, "Dynamic production system identification for smart manufacturing systems," *Journal of Manufacturing Systems*, vol. 48, pp. 192–203, 2018.

[45] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[46] R. Janicki and M. Koutny, "Semantics of Inhibitor Nets," *Information and Computation*, vol. 123, no. 1, pp. 1–16, 1995.

[47] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*. ACM New York, NY, USA, 1995, vol. 26.

[48] L. A. Wolsey, *Integer Programming*. John Wiley & Sons, 2020.

[49] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2025.

[50] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, 1st ed. Wiley, 1988.

[51] G. Lugaresi, V. V. Alba, and A. Matta, "Lab-scale Models of Manufacturing Systems for Testing Real-time Simulation and Production Control Technologies," *Journal of Manufacturing Systems*, vol. 58, pp. 93–108, 2021.

[52] G. Gamrath and M. E. Lübbecke, "Experiments with a Generic Dantzig-Wolfe Decomposition for Integer Programs," in *Experimental Algorithms*, P. Festa, Ed. Berlin, Heidelberg: Springer, 2010, pp. 239–252.

[53] IBM, "IBM ILOG CPLEX Optimization Studio Reference Manual," 2022.

[54] W. M. P. Van Der Aalst and J. Carmona, Eds., *Process Mining Handbook*. Cham: Springer International Publishing, 2022, vol. 448.

[55] E. G. Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis, "30 Years of GreatSPN," in *Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on His 70th Birthday*, L. Fiondella and A. Puliafito, Eds. Cham: Springer International Publishing, 2016, pp. 227–254.

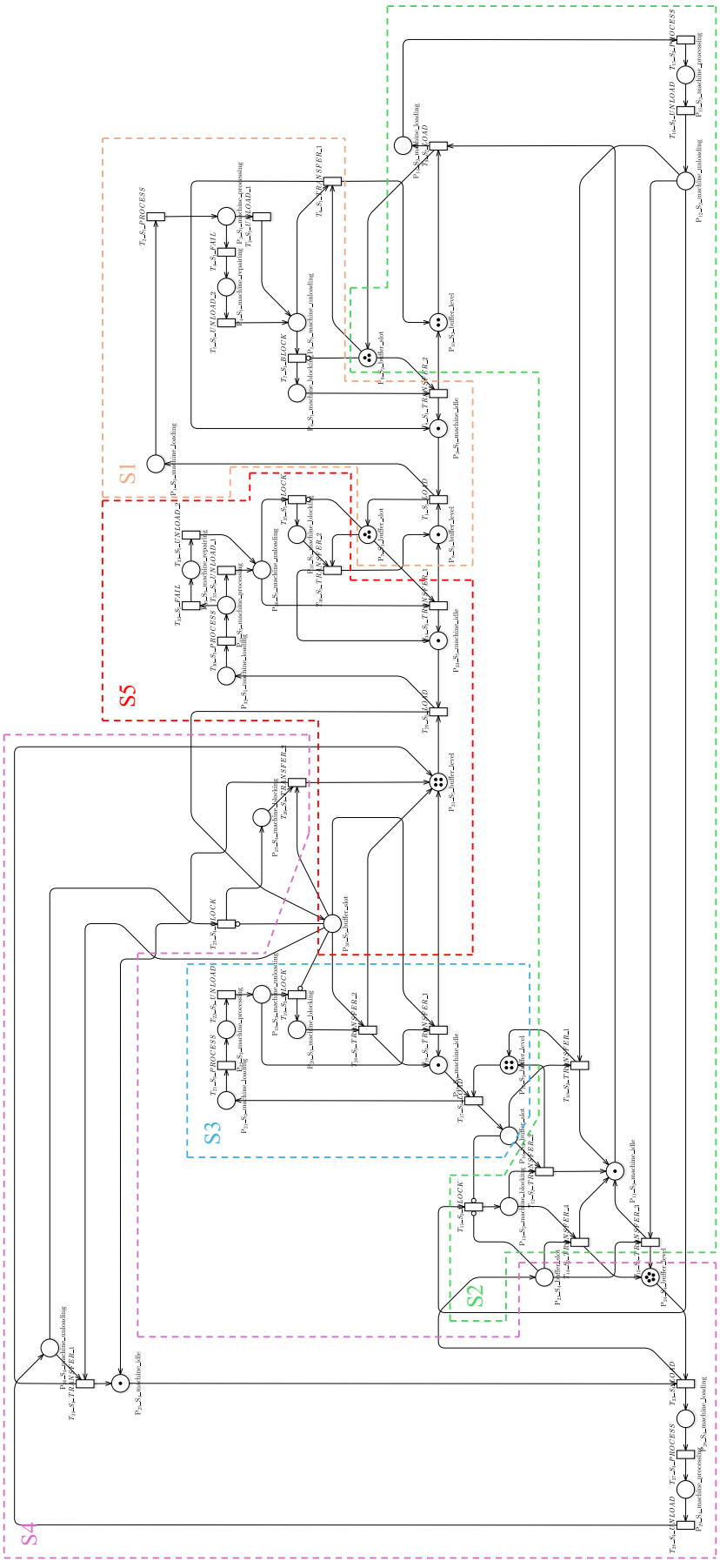

Fig. 8. Detailed Petri net models of the lab-scale LEGO manufacturing plant.

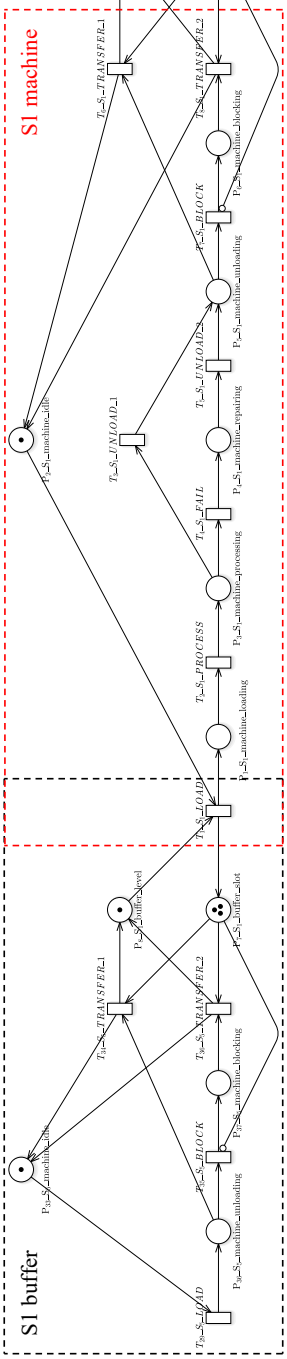

Fig. 9. Detailed subnet of the lab-scale LEGO manufacturing plant, highlighting S1 and its interfaces with S5 and S2.

Wei Cheng (Student Member, IEEE) received the B.S. degree from Northeastern University, Shenyang, China, in 2019, and the M.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering, Politecnico di Milano, Milan, Italy. His research interests include digital twin model discovery, Petri nets, and integer programming.

Lulai Zhu (Member, IEEE) is an Assistant Professor in the Department of Mechanical Engineering at Politecnico di Milano, Italy. He obtained a Ph.D. degree in Computing Research from Imperial College London, UK, in 2022 with a focus on model-driven performance analysis and architecture design of cloud-native applications. His research work currently centers around automated generation and tuning of digital twin models for complex manufacturing systems. He has also been an active contributor to several EU's Horizon/H2020 research projects, including DICE, RADON and AUTO-TWIN.

Andrea Matta (Senior Member, IEEE) is currently a Full Professor of Manufacturing with the Department of Mechanical Engineering, Politecnico di Milano, Milan, Italy. His research interests include analysis, design, and management of manufacturing and health care systems, digital twin discovering and generation, stochastic processes, control systems, process mining, simulation-optimization, and sustainable automation..