Discrete-Event System Model Discovery under Observational Ambiguity: Counterexample Generation via Spectral Learning*

Wei Cheng[1], Andrea Matta[1,*]

[1]Department of Mechanical Engineering, Politecnico di Milano, Via Privata Giuseppe La Masa 1, 20156 Milan, Italy

Abstract

Traditional data-driven discrete-event system model discovery using Petri nets often relies on access to both positive examples that describe admissible behavior and counterexamples that rule out forbidden behavior. In industrial settings, however, event logs provide only observed traces, which constitute a partial observation of system behavior, and counterexamples are typically unavailable. This paper addresses counterexample generation from event logs to support labeled Petri net discovery of discrete-event systems. Observed traces provide positive examples, while unobserved alternative events at each step form candidate counterexamples, since non-occurrence does not imply impossibility. We recover a hidden Markov model (HMM) from event logs via a spectral learning approach and use it only as an auxiliary probabilistic model to validate these candidate counterexamples. Beyond counterexample availability, event logs often exhibit observational ambiguity, where the same event label may correspond to multiple underlying system conditions, necessitating labeled Petri nets in which multiple transitions share the same event label. To handle this ambiguity, we formulate labeled Petri net discovery as an integer program and exploit its structure to develop a tailored branch-and-price algorithm. The proposed approach is validated on event logs from LEGO-based laboratory systems and a CNC machining center. Comparisons with process mining techniques show that our approach derives higher-fidelity Petri nets from real manufacturing data, enabling downstream applications such as supervisory control and digital twin deployment in discrete-event systems.

Keywords

Discrete-event system, Labeled Petri net discovery, Digital twin, Spectral learning, Integer programming

1. Introduction

Manufacturing systems fit naturally into discrete-event system (DES) models, since their state changes are caused by loading, processing, transport, blocking, repair, and similar events [1]. Petri nets are attractive for this setting because places and tokens can describe resource occupancy, buffer states, and concurrency in a form that engineers can inspect [2]. In many plants, however, the Petri net is unavailable or no longer matches the system. The available data are often just event logs from the running equipment. This makes event-log-based DES discovery relevant for simulation, supervisory control, and digital twin construction [3].

A log is only a record of what happened. It does not say what could have happened but did not appear in the finite trajectory. Classical Petri net synthesis and related identification methods can use positive examples of admissible behavior and counterexamples of forbidden behavior [4, 5, 6], whereas industrial logs usually provide only the first type of evidence. Declaring every unobserved continuation forbidden would exclude behavior that may simply be rare. Standard process mining avoids explicit counterexamples, but its models are usually aimed at control-flow description and may miss manufacturing states such as buffer occupancy [7].

We focus on the case where missing counterexamples and observational ambiguity occur together. Observational ambiguity means that the same event label can hide different system changes. For

PNAS'26, International Workshop on Petri Nets for Adaptive Systems, 2026

*The research 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) No. 202206230063.

*Corresponding author.

wei.cheng@polimi.it (W. Cheng); andrea.matta@polimi.it (A. Matta)

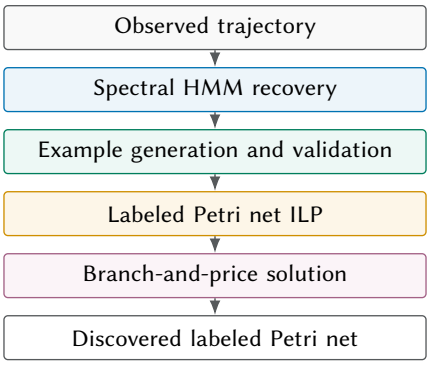


Figure 1: Workflow of the ongoing method. The HMM is an auxiliary probabilistic model for screening candidate counterexamples; the final DES model is a labeled Petri net.

example, an UNLOAD label may refer to unloading after normal processing or after a repair path. A single transition with that label is then too restrictive. We validate counterexamples from event logs and discover a labeled Petri net in which several transitions may share the same observed label.

The resulting pipeline turns a single trajectory into positive examples and HMM-screened counterexamples, then uses them to discover a compact labeled Petri net. This paper reports preliminary LEGO evidence and fixes the evaluation protocol for the full study. The benchmark against process-mining baselines and the full solver study are still in progress.

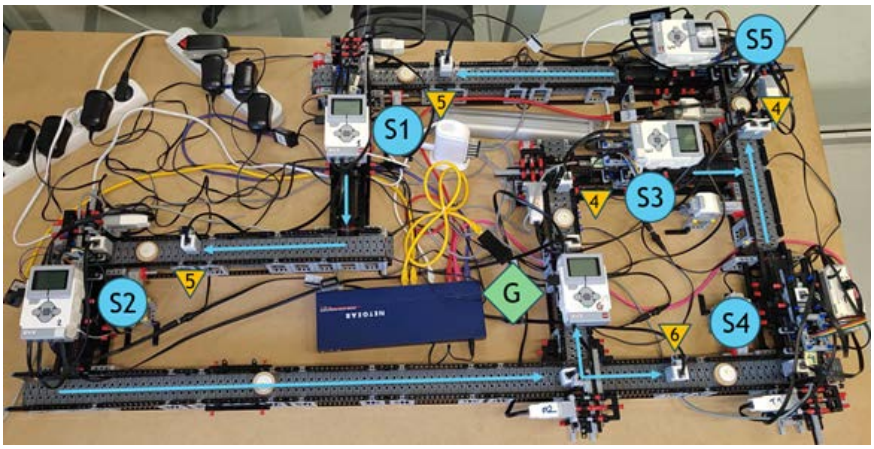
2. Approach

Figure 1 summarizes the pipeline. The method starts from a single long event trajectory. After selecting an anchor and a fixed suffix horizon, each observed next event after a local prefix becomes a positive example. For the same prefix, every label that is not observed next is listed as a candidate counterexample. This list is intentionally broad: it gives the discovery problem possible negative evidence without treating every unobserved event as impossible.

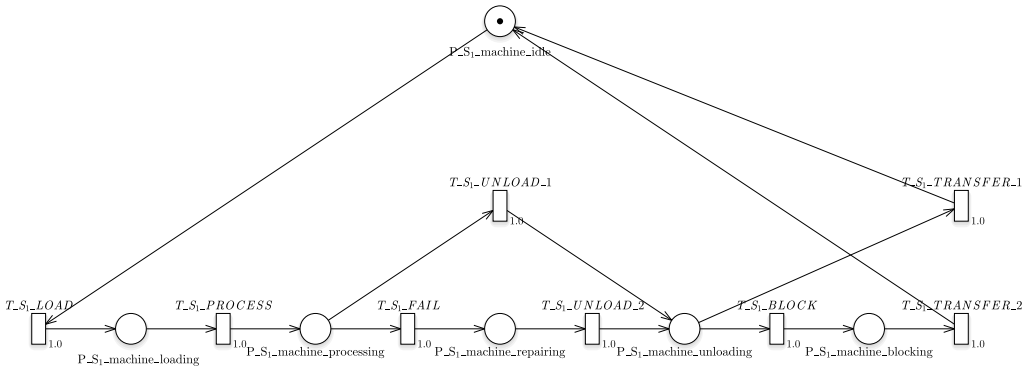
The candidate counterexamples are screened with an HMM recovered from the same event data. We do not use the HMM as the DES model. It only estimates whether an unobserved next event is still plausible after the observed prefix. We recover the HMM with the spectral realization approach of Huang et al. [8]. Because the data are one dependent trajectory, the trajectory is split into separated short traces after estimating dependence decay from the sample path, following the motivation of beta-mixing estimation [9]. We then filter from several initial priors and choose an anchor whose predicted suffix-start state is stable, using the standard fact that HMM filters can forget their initial distribution under suitable conditions [10].

For a candidate pair (s, e), where s is a local prefix and e is an unobserved next label, the recovered HMM estimates the conditional probability of observing e after s. If this probability is below η, the pair is kept as a validated counterexample. Otherwise, the candidate is treated as missing but admissible behavior and is not imposed as a hard exclusion. The validated examples define the labeled Petri net discovery problem. The integer program searches for a small net, measured by arcs and initial tokens, that reproduces all positive examples and excludes all validated counterexamples.

At this stage, the number of transitions allowed to share each event label is fixed before discovery. This keeps structure discovery separate from automatic label splitting, which is a related problem [11]. The compact formulation has a block structure induced by the example constraints. We use this structure in a Dantzig–Wolfe reformulation and a branch-and-price algorithm, so larger event-log horizons need not be solved through the compact ILP directly.



(a) LEGO plant topology.



(b) Discovered single-station labeled Petri net.

Figure 2: Preliminary LEGO evidence: (a) the five-station plant topology with serial stations S1, S2, and S5, parallel stations S3 and S4, preceding buffers, a conveyor, and S2-controlled routing through gate G; (b) the preliminary single-station labeled Petri net discovered from the event log.

3. Preliminary evidence and evaluation plan

Initial tests use the LEGO-based laboratory manufacturing system in Figure 2(a). The plant has five stations: S1, S2, and S5 form a serial route, while S3 and S4 operate in parallel. Each station is preceded by a buffer, products move on a conveyor, and gate G routes products under the control of S2. This layout is enough to create the ambiguity addressed here: the same event label can arise under different routing and buffer conditions.

Figure 2(b) gives the corresponding preliminary single-station result: a labeled Petri net whose repeated labels separate operational contexts that one transition per observed label would collapse. The single-station log contains loading, processing, repair, unloading, blocking, and idle events; station identity is observed, so the split reflects state and buffer context rather than station ambiguity. The preliminary settings are: the single-station case uses 100 log days, trace length 7, HMM rank 6, and discovery length 150; the downstream-buffer case uses 100 log days, trace length 13, HMM rank 20, and discovery length 150; both use the validation threshold $\eta = 0.001$. The downstream-buffer case also includes buffer slots and buffer level.

The full evaluation will use chronological train, validation, and test splits on the LEGO system and a CNC machining-center log. The training segments will be used for HMM recovery, example generation, counterexample validation, and Petri net discovery. The validation segments will choose the extraction horizon, η, and HMM order. The test segments will be kept for behavioral reporting. Comparisons will include representative process-mining discovery methods; ablations will remove the validation module, same-labeled transitions, and the branch-and-price solver.

The reported metrics will include alignment-based fitness, precision, their F-score, and a next-event

enablement rate on held-out logs, together with numbers of places, transitions, arcs, and initial tokens. For the solver study, we will report runtime, memory, optimality status, final gap, generated columns, and branch nodes. The evaluation is aimed at held-out behavioral fidelity and computational scalability, rather than recovery of an unknown ground-truth Petri net.

4. Conclusion

The paper addresses a practical gap in Petri net discovery from manufacturing logs. Logs provide positive evidence but rarely identify which unobserved behaviors are impossible, and a single event label may hide several physical state changes. The proposed pipeline uses a spectral HMM only as a probabilistic screen for candidate counterexamples, then discovers a labeled Petri net from the validated examples. The LEGO cases suggest that same-labeled transitions can separate operational contexts in the resulting net. The remaining work is to complete the LEGO and CNC evaluation, measure the effects of counterexample validation and label multiplicity, and finish the branch-and-price implementation. Label multiplicities are still specified before discovery; inferring them automatically remains open.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] C. G. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, Springer US, Boston, MA, 2008. doi:10.1007/978-0-387-68612-7.

[2] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with Generalized Stochastic Petri Nets, John Wiley & Sons, Chichester, 1995.

[3] G. Lugaresi, A. Matta, Automated manufacturing system discovery and digital twin generation, Journal of Manufacturing Systems 59 (2021) 51–66. doi:10.1016/j.jmsy.2021.01.005.

[4] E. Badouel, L. Bernardinello, P. Darondeau, Petri Net Synthesis, Springer, Berlin, Heidelberg, 2015. doi:10.1007/978-3-662-47967-4.

[5] M. P. Cabasino, A. Giua, C. Seatzu, Identification of petri nets from knowledge of their language, Discrete Event Dynamic Systems 17 (2007) 447–474. doi:10.1007/s10626-007-0025-0.

[6] G. Zhu, L. Yin, Y. Li, Z. Li, N. Wu, Identification of labeled petri nets from finite automata, Information Sciences 667 (2024) 120488. doi:10.1016/j.ins.2024.120488.

[7] W. M. P. Van Der Aalst, J. Carmona (Eds.), Process Mining Handbook, volume 448, Springer International Publishing, Cham, 2022. doi:10.1007/978-3-031-08848-3.

[8] Q. Huang, R. Ge, S. Kakade, M. Dahleh, Minimal realization problems for hidden markov models, IEEE Transactions on Signal Processing 64 (2016) 1896–1904. doi:10.1109/TSP.2015.2510969.

[9] D. J. McDonald, C. R. Shalizi, M. Schervish, Estimating beta-mixing coefficients via histograms, Electronic Journal of Statistics 9 (2015) 2855–2883. doi:10.1214/15-EJS1094.

[10] R. Douc, G. Fort, E. Moulines, P. Priouret, Forgetting the initial distribution for hidden markov models, Stochastic Processes and their Applications 119 (2009) 1235–1256. doi:10.1016/j.spa.2008.05.007.

[11] S. J. Van Zelst, J. Tai, M. Langenberg, X. Lu, Context-based activity label-splitting, in: C. Di Francescomarino, A. Burattin, C. Janiesch, S. Sadiq (Eds.), Business Process Management, volume 14159, Springer Nature Switzerland, Cham, 2023, pp. 232–248. doi:10.1007/978-3-031-41620-0_14.