2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 5-8, 2025. Vienna, Austria

2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 979-8-3315-3358-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/SMC58881.2025.11343563

Multidimensional Stochastic Petri Nets: A Novel Approach to Modeling and Simulation of Stochastic Discrete-Event Systems

Atieh Khodadadi and Sanja Lazarova-Molnar, *Senior Member, IEEE*

***Abstract*— Process Mining (PM) has been proven valuable for extracting process flows from data, also in the form of stochastic Petri net (SPN) models of systems. SPNs are widely recognized for their ability to model complex, stochastic systems and are extensively used in combination with PM. While SPNs provide an intuitive and straightforward way to model complex systems, representing changes across multiple dimensions, such as energy and waste, remains challenging in their standard frameworks. In this paper, we introduce an extension of stochastic Petri nets, termed Multidimensional SPNs (MDSPNs), by extending the SPN framework to capture dynamics along different dimensions. MDSPNs facilitate a comprehensive modeling of systems' behaviors from multiple perspectives, which can correspond to the diverse objectives of systems. To facilitate design and simulation of MDSPNs, we designed and developed MDPySPN, a Python library, which we also introduce in this paper. MDPySPN enables the simulation of MDSPNs by supporting alterations of multiple values at system events. With MDPySPN, we aim to provide researchers, engineers, and simulation professionals with a practical and extensible toolkit to model, simulate, and analyze MDSPNs, thereby supporting multi-objective optimization of stochastic processes in systems. Through a case study, we demonstrate the capabilities of modeling and simulation of MDSPNs using MDPySPN.**

I. Introduction

Complex systems, such as manufacturing, logistics, and healthcare, encompass multiple objectives. Each of these objectives can be linked to a dimension of the system, such as time, energy, cost, and waste [1]. The multi-objective nature of complex systems involves simultaneous management and optimization of multiple, often conflicting objectives [2]. In the context of complex systems management, modeling and simulation emerge as essential methodologies for analyzing systems' behavior and developing strategies to enhance and optimize complex systems [3]. Discrete-event simulation (DES) primarily focuses on changes in the system that occur at discrete points in time and correspondingly updates the simulation time clock [3]. In our recent work [4], we recognized the need for multidimensional models by developing model extraction techniques for energy-oriented models. We concluded that while traditional, time-driven DES models can accurately capture the temporal flow of factory operations, they often have difficulty incorporating other simultaneously varying dimensions such as energy use, waste, and CO_2 generation in the same model. As a result, these models may offer a less holistic view of system performance and potential improvements. Furthermore, the selection of performance indicators, aligned with the simulation objectives, directly influences the selection of relevant dimensions. For instance, adding 'carbon footprint' as a performance indicator would necessitate the inclusion of a corresponding environmental impact dimension. This highlights the need for methodologies capable of observing and modeling systems' behaviors across multiple dimensions, beyond time, such as energy consumption or waste generation [4]. To address this gap, we extend the DES formalism to redefine events as instantaneous occurrences in all relevant systems' dimensions, which are updated analogously to the simulation clock. We aimed for an intuitive way of achieving this redefinition, which became the central motivation for the present work.

With the increasing availability of sensor data in systems, Process Mining (PM) can be employed to automatically extract system models that are data-driven and can be updated automatically when the system changes [5, 6]. PM techniques use data captured by sensors to identify, analyze, and improve complex processes. Among the formalisms used by PM to represent discrete-event system behaviors, stochastic Petri nets (SPNs) are particularly well-suited [7]. Stochastic Petri Nets (SPN) provide a set of components for defining state-transition and event-scheduling mechanisms of stochastic discrete-event systems (SDESs) [8]. SPN's stochastic timing ability enables the representation of concurrent processes and probabilistic events and is ideal for analyzing, designing, and optimizing systems in various fields [8], such as manufacturing [9]. The visual design of SPNs offers an intuitive understanding of system dynamics, making it easier for stakeholders to understand and aiding in the identification of potential issues [10]. SPNs allow for detailed analysis and simulation of systems where temporal dynamics and randomness significantly influence operational dynamics [10]. However, conventional SPN models inherently focus on temporal and probabilistic aspects and typically do not consider the impacts of an event across multiple dimensions beyond time. This limitation can result in models that inadequately capture critical aspects of system behavior, especially in scenarios that require a multidimensional perspective to accurately capture system performance and interactions.

The limitation of considering multidimensional aspects in traditional SPN modeling highlights the need for an enhanced SPN modeling framework that captures both the temporal sequence of events and their multifaceted impacts across multiple system dimensions. To address these limitations, we introduce the concept of multidimensional modeling utilizing Multidimensional SPNs (MDSPNs). To support the implementation of MDSPNs, we developed MDPySPN, an extensible Python library designed specifically for the simulation of multidimensional stochastic Petri nets. We detail the architecture of MDPySPN, its core functionalities, and its real-world applications, illustrating its potential to support multi-objective optimization in complex stochastic environments.

The remainder of this paper is structured as follows: Section II explores the foundational concepts relevant to our study. Section III provides an overview of multidimensional modeling and introduces MDSPNs. Section IV presents a detailed technical description of the MDPySPN library. Section V demonstrates the application of MDPySPN through a case study of a manufacturing system. Section VI provides a summary and discusses potential future developments, and Section VII concludes the paper.

II. Background and Related Work

In the following subsections, we offer an overview of the specifics of modeling stochastic discrete-event systems (SDESs), followed by a definition and description of the types of SPNs that we use, and existing tools for SPNs modeling and simulation. Subsequently, we explore related works in modeling formalisms that integrate different dimensions of systems in simulation models.

A. Modeling of Stochastic Discrete-Event Systems

SDESs are widely encountered in real-world scenarios where systems behave unpredictably and are mainly driven by discrete events. These systems are characterized by a finite or countable set of states and probabilistic transitions, where events occur at discrete points in time, often governed by probability distributions. Common examples include manufacturing systems, computer networks, healthcare processes, and supply chains, where randomness arises from factors such as arrival times, processing durations, and resource availability [11].

SDESs exhibit several key characteristics that distinguish them from other systems [12]. First, an SDES operates within a discrete state space, meaning the system can exist in one of a finite or countable set of possible states. Second, state transitions can exhibit stochastic behaviors, meaning they can occur at points in time that are defined by a stochastic rule or probability distribution. Lastly, SDESs exhibit event-driven behaviors, with changes in system states occurring at distinct points in time, triggered by discrete events such as arrivals, departures, or task completions. SDESs are widely used across various domains, such as manufacturing, telecommunications, healthcare, and logistics, where uncertainty significantly influences system behavior.

Several formal methods have been developed for modeling SDESs, each offering distinct advantages in capturing randomness and discrete dynamics. Discrete-time Markov Chains (DTMCs) are a formalism for specific types of stochastic processes used for modeling systems where changes occur at discrete, separated points in time. Each state in a DTMC depends only on the previous state, following the Markov property, which means that the future state depends only on the current state and not on how that state was reached [13]. Finite automata provide another formalism for SDES modeling, where states represent system configurations and probabilistic transitions capture the uncertainty in state changes [14]. Lastly, SPNs incorporate stochastic timing into transitions, making them suitable for concurrent and distributed systems [15].

B. Stochastic Petri Nets

SPNs serve as a modeling formalism well-suited for representing and analyzing discrete-event systems, particularly considering the dynamics and inherent randomness of complex systems, such as manufacturing systems [16]. SPNs enable modeling and simulation of concurrent processes and probabilistic events, supporting system analysis, design, and optimization across diverse domains. SPNs are able to handle uncertainty, which makes them particularly useful for evaluating system performance, reliability, and other essential quantitative measures [17].

An SPN consists of a finite set of places, a finite set of arcs, and an initial state represented by the initial marking of the Petri Net. Each place can hold a collection of tokens, and a transition is enabled and prepared to fire when its input places possess the requisite number of tokens. Formally, the type of SPNs that our library models are based on is described in [15], according to which, an SPN is defined as a tuple $SPN = (P, T, A, G, m_0)$, where:

- $P = \{P_1, P_2, \dots, P_m\}$ represents the set of places, depicted as circles,
- $T = \{T_1, T_2, \dots, T_n\}$ is the set of transitions with their distribution functions or weights, illustrated as bars,
- $A = A^I \cup A^O \cup A^H$ denotes the set of arcs, with A^O being the output arcs, A^I the input arcs, and A^H the inhibitor arcs, each arc bearing a specific multiplicity,
- $G = \{g_1, g_2, \dots, g_r\}$ encompasses the set of guard functions associated with various transitions, and
- m_0 is the initial marking, defining the token distribution across the places.

Each transition is described as $T_i = (type, F)$, where $type \in \{timed, immediate\}$ indicates the transition's type, and *F* represents either a probability distribution function for timed transitions or a firing weight/probability for immediate transitions. In SPNs, markings are updated through the firing of transitions that are enabled by the current marking, including the possible concurrent firing of multiple enabled transitions. The firing times of the transitions are governed by a stochastic process, with each enabled transition associated with a clock that tracks the remaining time until firing. The countdown speed of these clocks can vary based on the current marking, and a change in marking occurs when one or more clocks reach zero [8].

C. SPN Modeling and Simulation Tools

There are several widely used modeling and simulation (M&S) tools for SPNs, such as TimeNET [18], GreatSPN [19], and SPNP [20], that can handle various types of Petri nets, such as place/transition nets, colored Petri nets, and SPNs. Most SPNs' applications focus primarily on the temporal flow of events. Our library, MDPySPN, enables M&S of MDSPNs (as detailed in Section IV) to better capture multidimensional behaviors in complex systems. MDPySPN, building on our previously published Python-based SPN simulation library PySPN [21], supports several probability distributions, including exponential, normal, and Weibull. Unlike existing tools, MDPySPN enables

simulation of a unified MDSPN model, where each transition can influence multiple dimensions simultaneously.

D. Related Work on Integrating Multiple Dimensions in Simulation Models

In our definition of a multidimensional model, we redefine and observe an event as an instantaneous occurrence across several systems' dimensions, for instance, energy, waste, and time, which occurrence triggers a change in the state variables. This expanded definition allows events to be observed more comprehensively and better reflect complex systems' behaviors in an attempt to target diverse objectives, such as enhanced sustainability and reduced cost. Several alternative modeling methods have been explored for their abilities to represent multidimensional systems, which are usually limited to considering two dimensions of interest. We explore these in the following.

Discrete-Event System Specification (DEVS) [3] is a modular and composable formalism for discrete-event systems modeling. The modular and hierarchical feature of DEVS means constructing models from a collection of 'atomic models,' each defined by its unique states and transitions. These atomic models are then interconnected to form more complex 'coupled models,' enhancing the system's overall complexity. Colored Petri Nets (CPNs) [22] extend traditional Petri nets by incorporating data values, known as "colors," into tokens, enabling the representation of complex system states and facilitating the modeling of concurrent and distributed systems. For instance, Wang et al. [23] focused on developing an energy model of flexible manufacturing systems (FMSs) using Colored Timed Petri Nets (CTPNs) to address the uncertainties in task assignment and operation time in FMSs. Wang et al. applied their method to a case study of an FMS with two jobs. The results demonstrated that CTPNs could effectively model and simulate both energy consumption and timing.

Rule-based modeling [24] is another approach employed in system modeling, where a system's behavior is defined by a set of explicit rules (facts) that dictate interactions and state changes in the system. Each fact is a data record, a mapping from field names to values, representing observed information about the system being monitored. The fact memory, a collection of stored facts, is used to evaluate rules. Matching conditions against the fact memory involves checking whether the required variables and their values are specified in the rules align with the stored facts. Rule-based modeling is useful for systems where interactions are clearly defined by conditional rules, making implementation and understanding more straightforward, but applying it in environments with high uncertainty can be challenging [25].

Another approach for multidimensional modeling is hybrid modeling [26], which combines different modeling approaches, such as discrete event modeling with continuous modeling. For instance, Sobottka and colleagues [27] developed a hybrid simulation-based optimization tool that integrates discrete events and continuous simulations to enhance energy efficiency and productivity in manufacturing systems. The approach of Sobottka et al. captures dynamic interactions between material and energy flows, considering both energy consumption and time dimensions. While hybrid modeling offers powerful capabilities for representing complex systems, it also poses practical challenges, particularly in the modeling and simulation of multidimensional systems. One of the main challenges is the accurate representation of continuous dynamics, which requires formulating mathematical equations that describe system behaviors over time. This task is often complex, as it demands a deep understanding of the underlying physical processes and complex interactions within the system, such as temperature fluctuations or energy flow dynamics.

In summary, while models like CPNs, rule-based, and hybrid approaches can represent multiple dimensions, they generally require extensive manual modeling and are not easily derived from data via process mining (PM), except for SPNs [28]. SPNs are well-suited for data-driven automatic modeling via PM [7] but are limited in capturing multidimensional behaviors of complex systems. To address this gap, we developed MDSPNs, which extend SPNs to support automatic, multidimensional model extraction using standard PM techniques, enabling comprehensive system analysis across dimensions.

E. Related Work on Simulation of Multidimensional Models

Several tools and platforms are available to support the simulation of systems with multidimensional impacts, each offering specific functionalities and varying levels of complexity to accommodate diverse application requirements. These tools enable analysis of system behavior across multiple dimensions, such as time, energy consumption, and waste generation. Several prominent examples are AnyLogic [29], ExtendSim [30], Mercury[31], and MATLAB/Simulink [32]. However, our library, MDPySPN, is currently the only open-source tool supporting detailed multi-dimensional Petri net simulation features, enabling systems analysis in various dimensions.

III. Multidimensional Modeling

Research demonstrates that various modeling approaches can integrate multiple dimensions in a single model. However, our approach is novel in that it uses SPNs as an intuitive framework for multidimensional modeling. Our approach enhances the transparency of system behavior across multiple dimensions in a unified model. Additionally, SPNs enable data-driven model extraction utilizing process mining [33], which further supports the development of Digital Twin models [5], facilitating real-time system analysis and optimization. In the following sections, we will provide a detailed explanation of MDSPNs.

A. Multidimensional SPNs

While traditional SPNs are effective for modeling SDESs, they are limited in representing multiple operational dimensions such as energy or waste. To address this limitation, we introduce MDSPNs as an extension of SPNs by allowing transitions to impact multiple dimensions concurrently. To intuitively describe the different behaviors of the system in each dimension of interest, we introduce the concept of unidimensional SPN models for each dimension of interest (each one is essentially a standard SPN focusing on a single dimension). These unidimensional models can be standalone to facilitate the systems' behavior analysis in a given dimension, identify bottlenecks, and explore opportunities to optimize system performance in that dimension. Furthermore, unidimensional models serve as a

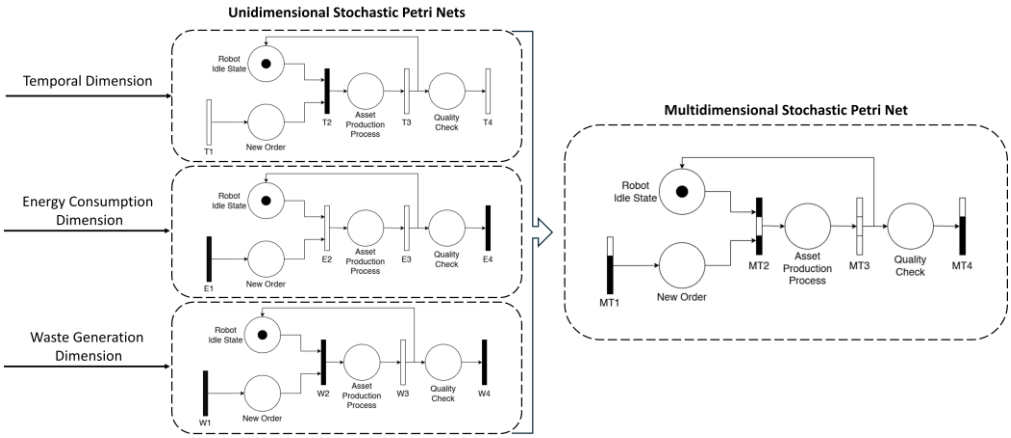

Figure 1. Multidimensional stochastic Petri net model of the example system.

foundation for constructing MDSPN models. In our recent research [34], we developed a methodology for data-driven process discovery of unidimensional models. Our approach facilitates the automatic extraction of unidimensional models from data corresponding to each dimension of interest.

For example, consider a production flow system involving an assembly robot, where the system's behavior influences multiple dimensions, such as waste generation, energy consumption, and time. The assembly robot initiates the production process upon the arrival of raw materials and manufactures a product over a defined duration. During the production process, the robot consumes energy and generates waste. Once the production process is completed, the product is ready to move on to the next phase. Once completed, the product undergoes a quality check, and after a duration of time, the product is sent to subsequent steps. In periods without incoming materials, the robot remains in an idle state, consuming energy until the active state when the next task arrives. This scenario demonstrates how a system exhibits distinct behaviors across different dimensions.

In Figure 1 (left), we illustrated the unidimensional model of the example system across dimensions of time, energy consumption, and waste generation. In that sense, certain transitions may contribute to one dimension while being non-contributing in another (analogous to timed and immediate transitions in temporal flows). For instance, the idle process where a machine waits for a new task is represented differently across dimensions: the related transition, T2, represents an immediate transition in the temporal dimension, indicating no time delay (T2 fires immediately upon the arrival of a new order). In contrast, its corresponding transition in the energy-consumption dimension, E2, consumes energy while W2 (the corresponding transition in the waste dimension) does not contribute to the waste generation dimension. Furthermore, the transition associated with the arrival of new tasks, T1, contributes only to the temporal dimension. The production activity impacts on all dimensions, i.e., time (T3), energy consumption (E3), and waste generation (W3).

The unidimensional models are further integrated into a comprehensive multidimensional model, which captures behavior flows in all dimensions. The integration is intended to be an automated step that will not be a burden on the modelers' side. In the MDSPNs model, each transition can impact from none to all identified dimensions. In MDSPNs, transitions create and destroy tokens and update simulation clocks across multiple dimensions, such as energy consumption and waste generation. These updates across multiple dimensions can be in a positive or negative direction, unlike time, which only progresses positively. This is to denote that some activities can add to or subtract from the corresponding dimension (for example, when the system simultaneously produces and consumes energy). Currently, we assume identical topologies without losing generalizability, as transitions can also be immediate or, as we denote them in our SPNs' extension, noncontributing in different dimensions. To accommodate the multiple dimensions, we extended our original SPN formalism, as described in Section II-B, with the following:

- **Transition definition:** Adjusted the transition definition to reflect the different dimensions that a transition can impact, i.e., $T_i = (dim, type, F)$ where dim is the impacted dimension and $type \in \{contributing, noncontributing\}$.
- **Graphical element:** Introduced a new graphical element for multidimensional transitions that is visually split in segments to map impacts on all relevant dimensions. For instance, in Figure 1 (right), transitions MT1 to MT4 illustrate this concept, where the top segment of the transition represents the behavior of that transition in the time dimension, the middle segment indicates energy consumption, and the bottom segment corresponds to the waste generation dimension
- **Distinct clocks:** Defined distinct simulation clocks for each dimension. The temporal clock tracks time progression (conventional simulation clock), while clocks in other dimensions monitor updates in their corresponding dimensions. These clocks are updated upon transition firing, ensuring dimension-specific representation of system behavior.

In the example of unidimensional models in Figure 1, we next integrate all unidimensional models into a multidimensional model. In this integration, tokens move as they do in the temporal dimension, while transitions that impact other dimensions alter the values of their respective dimensions in their related dimensions' clock. This example demonstrates the capability of MDSPNs to capture both the

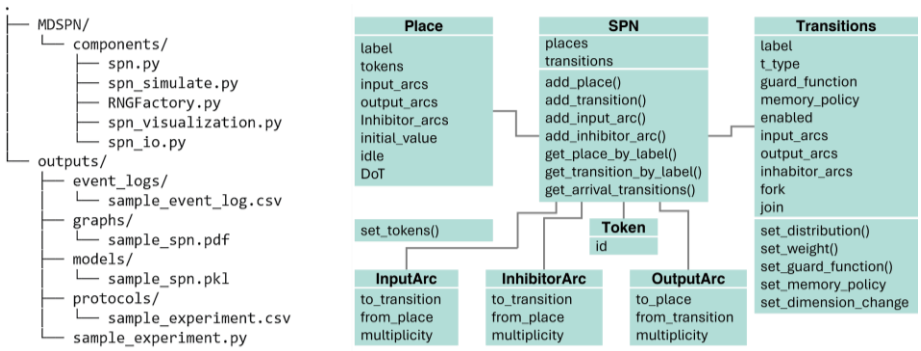

Figure 2. The structure of the directory and SPN class schematic.

progression of process steps over time as well as their effects on other dimensions such as energy and waste. This capability of MDSPNs is fundamental for accurately reflecting the dynamic nature of real-world systems, where actions in a process can lead to the consumption of resources or the generation of waste.

IV. MDPySPN: A Simulation Tool for MDSPNs

MDPySPN is an open-source, user-friendly, and extensible Python library that expands the capabilities of SPNs by supporting traditional simulations while integrating the impact of events across multiple system dimensions. This flexibility enables MDPySPN to adapt to a wide range of multi-objective optimization scenarios, reflecting the complex interdependencies in complex systems. The development of MDPySPN, an extended version of our earlier library, PySPN [21], presents a step forward in the simulation and analysis of multidimensional stochastic processes by supporting the modeling and simulation of complex systems across different dimensions.

MDPySPN supports simulations in which transitions are executed in terms of time and simultaneously affect other dimensions such as energy and waste. This multi-impact of transitions—where they can instantaneously progress the system's state while altering resource usage or waste production—underscores the multidimensional representation and analysis capabilities of MDPySPN. In the following, we detail the organizational framework and key functionalities of the MDPySPN library.

A. MDPySPN Directory Structure

We illustrate the directory structure of MDPySPN and a schematic of the MDSPN class design in Figure 2. The components directory (components/) contains the essential Python files for defining and executing SPNs:

- **spn.py:** Defines core SPN classes (SPN, Place, Transition, etc.) for construction, including properties such as guard functions, dimension tracking, etc.
- **spn_simulate.py:** Executes the MDPySPN simulation, managing tokens in places to execute event logs and protocols.
- **RNGFactory.py:** includes a factory method to generate random samples from probability distributions using Scipy [35].
- **spn_visualization.py:** employs the graphviz library [36] (graph visualization library) to create graphical representations of MDSPNs. The resulting graph includes an information box detailing the system's input and output, along with the total changes in each dimension. Beside each transition, tracking tables display dimension values affected by transitions.
- **spn_io.py:** offers functions for printing SPN descriptions, statistics, simulation protocols, and event logs, and for importing and exporting SPNs in Python's pickle format.

B. MDPySPN Outputs Directory

The outputs directory (outputs/) stores simulation outputs as follows:

- **event_logs/:** CSV files that record each event in a simulation. MDPySPN enables the creation of multidimensional event logs that track changes across various dimensions beyond just time.
- **graphs/:** Graphical representations of SPNs, detailing structure and dimensions.
- **models/:** Holds serialized SPN models that can be reloaded for further analysis.
- **protocols/:** Contains protocol in CSV format, capturing detailed simulation data, such as token movements.

C. MDPySPN Key Functionalities

The key functionalities (KFs) of the MDPySPN Library are described as follows:

KF1. Transition specifications: To manage transition properties effectively, we incorporated specific settings into the simulation adjustments as follows:

- **Input_transition/Output**: traces the token generated/destroyed by a transition.
- **Join/Fork:** enables the merging or splitting of tokens, making it particularly useful in systems such as manufacturing systems, where materials or objects are combined to produce new items or separated into distinct components. In this process, token IDs are reassigned to reflect these transformations.

KF2. Dimension tracking. Special data structures, referred to as "Tracking Tables", are implemented to track the impact of transitions across multiple dimensions. This configuration allows every transition to influence multiple dimensions

and, conversely, each dimension to be affected by any transition. For example, it enables simultaneous tracking of energy efficiency and waste generation caused by a specific machine activity. Each dimension is associated with a distinct simulation clock that progresses independently. Transitions act as synchronization points across dimensions. Key attributes of the tracking tables include:

- **total_dimension:** specifies the total number of dimensions in the model.
- **add_dimension_change**: is used to specify adjustments in the dimension when a transition occurs. For example, in the energy dimension, add_dimension_change defines the amount of energy consumed or generated by a particular transition.

KF3. Handling idle states. To integrate all dimensions in a single simulation model, idle places account for the time tokens remain idle. Consider a scenario in which a machine remains idle until a new task arrives, consuming energy. In SPNs, idle states are not modeled to reflect time-dependent transitions, leading to a disconnect between the energy consumed while idle and the actual time expended in the idle state. This modeling omission means that upon the arrival of a new task, transitions to subsequent states are executed without any temporal delays, failing to account for the energy consumption accrued during idle periods. This oversight in idle state modeling systematically underestimates energy utilization, as the model does not capture the continuous energy consumption during idle phases. In contrast, MDPySPN implements a time-dependent mechanism for utilizing transition dimension values, denoted as DoT=1. This indicates that the duration a token remains in the idle place is quantitatively considered. The transition computes the consumption of resources using (1). The consumption (C_{idle}), expressed in units of the energy dimension (such as kilowatt), is determined by the product of the elapsed time in the idle place (Δt), measured in time units (minutes, seconds), and the specific rate for that dimension (R_{idle}). Here, R_{idle}, denotes the consumption rate per unit of time, quantified in units of the dimension per time unit (such as kilowatt per second).

$$C_{idle} = R_{idle} + \Delta t \quad (1)$$

KF4. Validation of MDPySPN models. MDPySPN facilitates model validation to ensure the simulations accurately represent the multidimensional behavior of the real-world system. MDPySPN provides two types of validation. **Face validation** confirms that the model visually aligns with the real-world system's processes, providing initial confirmation of model fidelity [37]. To aid in the face validation process, MDPySPN generates a graph as an output at the end of the simulation run. **Output Validation,** focuses on quantitative measures and compares simulated results against actual data from the real-world system [38]. Output comparison primarily involves Key Performance Indicators (KPIs) for each dimension, such as system throughput, input, and output rates, alongside energy KPIs, including total energy consumption per asset and across the system. We refer users to the GitHub repository [39] for comprehensive information about the MDPySPN tool and guidance on initial setup. This documentation provides detailed instructions for utilizing the tool.

V. MDPySPN Application Example

To demonstrate the application of MDPySPN, we model and simulate a simplified manufacturing line, considering its progression through time, energy, and waste dimensions. Our case study manufacturing line features a machine with two states: idle and active. When a new order arrives, the machine immediately transitions to the active state, initiating the preprocessing transition. Furthermore, the processing activity runs for a predetermined duration until production is complete, at which point the machine returns to its idle state. The machine operates by manufacturing products, which involves consuming energy and generating waste, and it switches to an idle state between orders

Before initiating the simulation with MDPySPN, it is essential to align the impact of transitions in other dimensions, such as energy, waste, and cost, with the time dimension. For instance, as we noted, the processing activity consumes energy and generates waste; therefore, it is essential to specify the dimensions of its corresponding transition impacts along with the value it adds. Similarly, the energy consumed during idleness must be calculated and added to the energy dimension value upon firing the preprocessing transition for the immediate transition associated with the idle state. Following the modeling of the system, we explored the following key aspects of utilizing the MDPySPN for this system, including:

- **Performance Metrics:** KPIs are quantified for each dimension, such as total energy consumption (overall and per process), total waste generation, and system throughput (input and output rates).
- **Identification of Bottlenecks:** MDPySPN helps pinpoint which components are limiting overall performance. For instance, by examining token accumulations or delays in the model, we can identify bottlenecks in the production line.
- **Optimization Opportunities:** With the multidimensional view, models support multi-objective optimization seamlessly. For example, one could adjust operational schedules or resource allocations to increase throughput, reduce waste, and improve energy efficiency simultaneously.
- **Predictive Insights:** MDPySPN can be employed for predictive analysis. By observing how changes in one dimension affect others, practitioners can forecast systems' behaviors under different scenarios (such as what happens to energy use and waste if production volume increases) and proactively make decisions.
- **Scalability and Flexibility:** MDPySPN reflects the system changes (such as increased demand) and flexibly accommodates new tasks or products, supporting future system growth.

In our recent work [39], we used MDPySPN to simulate MDSPNs extracted from an illustrative case study. In Table 1 and Table 2, we presented an excerpt of the generated simulation protocol alongside the corresponding

Table 1. Excerpts of the generated protocol file.

Place	Time	Marking
New Task	2.28	1
New Task	2.28	2
New Task	2.28	1
Production Process	2.28	1
Idle	2.28	0
Production Process	2.28	1
…	…	…

Table 2. Excerpts of the generated multidimensional event log file.

Time Stamp	ID	Energy Stamp	Waste Stamp	Event
3.12	2	0	0	New Task (begin)
3.12	2	0	0	Preprocessing (begin)
5.12	3	77.92	0	Processing (begin)
5.12	4	77.92	0	Processing (begin)
5.12	3	177.92	1.5	Order Completed
…	…	…	…	…

multidimensional event log. Each entry in multidimensional event logs records a distinct event, such as the start or end of an activity. Additionally, the multidimensional event log tracks changes across predefined dimensions, each noted as "dimension_stamp". The multidimensional logging approach facilitates thorough system analysis and supports multi-flow process mining [37], aiding data-driven simulation and Digital Twin model extraction. Note the role of fork and join conditions, which require updating the event ID to a new number using a standardized method. As observed in the event log, when a fork condition, such as the "Production Process", is triggered, tokens' IDs are reassigned to a new value.

In Figure 4, we illustrate a simulation output MDSPN graph for the case study manufacturing line. The graph includes the simulation's multidimensional clock, positioned in the top left corner. Each transition is divided into the total number of predefined dimensions in order of time, energy, and waste. Additionally, each tracking table besides its corresponding transition represents the state of the transition's impact across all related dimensions. For instance, the preprocessing transition affects energy and time dimensions, while the processing transition influences the dimensions of time, energy, and waste.

In **Error! Reference source not found.**, we present a terminal output excerpt from the simulation run, showing the runtime, final places, and dimension values. The last lines summarize the simulation, including key KPIs for each dimension, such as total energy consumption, input/output, and waste generated.

VI. Summary and Discussion

Collectively, our contributions provide a novel framework that bridges the gap between data-driven multidimensional process analysis and simulation-based multi-objective decision-making support. In summary, the main contributions of the presented work are:

- **Multidimensional Stochastic Petri Nets (MDSPNs):** Introduction of an extended stochastic Petri net formalism that captures the behavior of a single system in multiple dimensions (such as time, cost, resource utilization) intuitively in one framework.
- **MDPySPN Simulation Library:** Development of a Python-based discrete-event simulation tool that implements MDSPNs. MDPySPN can be simulated without requiring specialized simulation languages.
- **Multi-Objective decision making support:** Built-in capabilities for multi-objective analysis in MDPySPN, stakeholders can evaluate and optimize trade-offs between multiple performance indicators (for example, maximizing throughput while minimizing cost and energy consumption).
- **Data-Driven Model Extraction:** It is possible to extract simulation models automatically from event logs using process mining techniques [6]. Consequently, by utilizing process mining to extract MDSPNs, the MDPySPN simulation tool can be employed as a tool for operational decision-making using real-time process data.

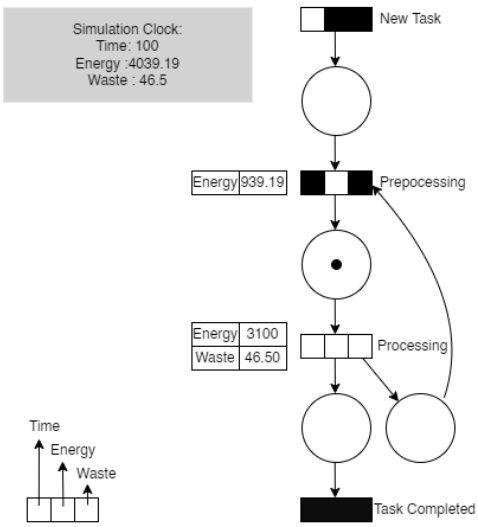

Figure 4. The output MDSPN of the case study manufacturing line.

```
Marking at time 0
Place New Tasks, #tokens: []
Place Idle, #tokens: [Token(1)]
Place Production Process, #tokens: []
Place Production Done, #tokens: []
...
Transition New Task fires at time 3.12
Transition Preprocessing (begin) fires at time 3.12
Transition Processing (begin) fires at time 5.12
...
Input value for New Task: 31
Output value for Task Completed: 31

Summary of Dimensions:
energy: 4039.19
waste: 46.50
```

Figure 3. Excerpt of the terminal output for the simulation run.

VII. Conclusion

In this paper, we introduced Multidimensional Stochastic Petri Nets (MDSPNs) and the supporting Python-based MDPySPN library. MDSPNs extend classical stochastic Petri nets and provide an intuitive representation of processes in multiple dimensions. Through a case study, we demonstrated the use of MDSPNs. Looking ahead, there are several open challenges and avenues for future work:

- **Integration of Unidimensional Models**: Exploring methods to combine independently developed models (each for one dimension) into a coherent

MDSPN. For example, accurately capturing a battery's charging behavior demands integrating its energy-flow model with its time-dependent dynamics so that both aspects operate consistently within one model.

- **Integration of Optimization Tools:** Integration of optimization algorithms and incorporating multi-objective optimization techniques into MDPySPN to simultaneously explore improvements across multiple performance metrics (such as minimize cost and energy while maximizing throughput).
- **ID Tracking for Process Fork and Join Conditions:** Develop methods to enhance ID tracking in process fork and join conditions, enabling detailed event logs, facilitating the extraction of models directly from such systems' event logs.

ACKNOWLEDGMENT

The authors extend their thanks for the funding received from the ONE4ALL project funded by the European Commission, Horizon Europe Programme under Grant Agreement No. 101091877.

REFERENCES

[1] H. Ma *et al.*, "Multi-objective production scheduling optimization and management control system of complex aerospace components: a review," *The International Journal of Advanced Manufacturing Technology,* vol. 127, no. 11-12, pp. 4973-4993, 2023.

[2] S. Sharma and V. Kumar, "A comprehensive review on multi-objective optimization techniques: Past, present and future," *Archives of Computational Methods in Engineering,* vol. 29, no. 7, pp. 5605-5633, 2022.

[3] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of modeling and simulation*. Academic press, 2000.

[4] A. Khodadadi and S. Lazarova-Molnar, "Data-Driven Extraction of Simulation Models for Energy-Oriented Digital Twins of Manufacturing Systems- An Illustrative Case Study," in *2024 Winter Simulation Conference (WSC)*, 2024: IEEE.

[5] 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.

[6] J. J. Ferronato and E. E. Scalabrin, "PM2Sim: the automated creation of a simulation model from process mining," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021: IEEE, pp. 2232-2237.

[7] S. Shahzadi, X. Fang, and D. A. Alilah, "Role of Stochastic Petri Net (SPN) in process discovery for modelling and analysis," *Mathematical Problems in Engineering,* vol. 2021, no. 1, p. 8699164, 2021.

[8] P. J. Haas, "Stochastic Petri nets for modelling and simulation," in *Proceedings of the 2004 Winter Simulation Conference, 2004.*, 2004, vol. 1: IEEE.

[9] J. Luo, S. Yi, Z. Lin, H. Zhang, and J. Zhou, "Petri-net-based deep reinforcement learning for real-time scheduling of automated manufacturing systems," *Journal of Manufacturing Systems,* vol. 74, pp. 995-1008, 2024.

[10] F. Bause and P. S. Kritzinger, *Stochastic petri nets*. Vieweg Wiesbaden, 2002.

[11] A. Zimmermann, *Stochastic discrete event systems*. Springer, 2007.

[12] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.

[13] G. G. Yin and Q. Zhang, *Discrete-time Markov chains: two-time-scale methods and applications*. Springer Science & Business Media, 2005.

[14] P. R. D'Argenio and J.-P. Katoen, "A theory of stochastic systems part I: Stochastic automata," *Information and computation,* vol. 203, no. 1, pp. 1-38, 2005.

[15] S. Lazarova-Molnar, "The proxel-based method: Formalisation, analysis and applications," Otto-von-Guericke-Universität Magdeburg, Universitätsbibliothek, 2005.

[16] G. Balbo, "Introduction to stochastic Petri nets," in *School organized by the European Educational Forum*, 2000: Springer, pp. 84-155.

[17] P. J. Haas, *Stochastic petri nets: Modelling, stability, simulation*. Springer Science & Business Media, 2006.

[18] A. Zimmermann and M. Knoke, "TimeNET 4.0," 2001.

[19] S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis, "The GreatSPN tool: recent enhancements," *ACM SIGMETRICS Performance Evaluation Review,* vol. 36, no. 4, pp. 4-9, 2009.

[20] C. Hirel, B. Tuffin, and K. S. Trivedi, "Spnp: Stochastic petri nets. version 6.0," in *Computer Performance Evaluation. Modelling Techniques and Tools: 11th International Conference, TOOLS 2000 Schaumburg, IL, USA, March 27–31, 2000 Proceedings 11*, 2000: Springer, pp. 354-357.

[21] J. Friederich and S. Lazarova-Molnar, "PySPN: An Extendable Python Library for Modeling & Simulation of Stochastic Petri Nets," in *International Conference on Simulation Tools and Techniques*, 2023: Springer, pp. 70-78.

[22] V. Gehlot and C. Nigro, "An introduction to systems modeling and simulation with colored petri nets," in *Proceedings of the 2010 winter simulation conference*, 2010: IEEE, pp. 104-118.

[23] Q. Wang, X. Wang, and S. Yang, "Energy modeling and simulation of flexible manufacturing systems based on colored timed petri nets," *Journal of Industrial Ecology,* vol. 18, no. 4, pp. 558-566, 2014.

[24] K. Havelund, "Rule-based runtime verification revisited," *International Journal on Software Tools for Technology Transfer,* vol. 17, pp. 143-170, 2015.

[25] D. Heckerman and E. J. Horvitz, "The myth of modularity in rule-based systems," *arXiv preprint arXiv:1304.3090,* 2013.

[26] L. Lättilä, P. Hilletofth, and B. Lin, "Hybrid simulation models–when, why, how?," *Expert systems with applications,* vol. 37, no. 12, pp. 7969-7975, 2010.

[27] T. Sobottka, F. Kamhuber, M. Rössler, and W. Sihn, "Hybrid simulation-based optimization of discrete parts manufacturing to increase energy efficiency and productivity," *Procedia Manufacturing,* vol. 21, pp. 413-420, 2018.

[28] Y. Lee, J. Shin, and W. Lee, "Manufacturing process analysis framework for process mining: case study of fully automated factory applications," *The International Journal of Advanced Manufacturing Technology,* pp. 1-24, 2025.

[29] AnyLogic. "AnyLogic Simulation Software." https://www.anylogic.com/ (accessed 13rd March 2025.

[30] "ExtendSim." https://extendsim.com/ (accessed 2024).

[31] B. Silva *et al.*, "Mercury: An integrated environment for performance and dependability evaluation of general systems," in *Proceedings of industrial track at 45th dependable systems and networks conference, DSN*, 2015, pp. 1-4.

[32] "Simulink." https://www.mathworks.com/products/simulink.html (accessed 2024).

[33] W. Van Der Aalst, "Process mining," *Communications of the ACM,* vol. 55, no. 8, pp. 76-83, 2012.

[34] A. Khodadadi and S. Lazarova-Molnar, "Multi-flow Process Mining for Comprehensive Simulation Model Discovery," presented at the ICICM, 2024.

[35] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature methods,* vol. 17, no. 3, pp. 261-272, 2020.

[36] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz—open source graph drawing tools," in *Graph Drawing: 9th International Symposium, GD 2001 Vienna, Austria, September 23–26, 2001 Revised Papers 9*, 2002: Springer, pp. 483-484.

[37] O. Balci, "Principles and techniques of simulation validation, verification, and testing," in *Proceedings of the 27th conference on Winter simulation*, 1995, pp. 147-154.

[38] J. P. Kleijnen, "Statistical validation of simulation models," *European journal of operational research,* vol. 87, no. 1, pp. 21-34, 1995.

[39] A. Khodadadi. "MDSPN." https://github.com/atikh/MDSPN (accessed 2024).